McGRAW HILL EDUCATION WINNING WE EDGE SERIES

Computer Science and Information Technology

CS & IT

for

GATE

GRADUATE APTITUDE TEST in ENGINEERING

About the Author

Prof. N B Venkateswarlu completed his M.Tech from IIT-Kanpur in 1988 and earlier in 1986, he completed his B.Tech from SV University College of Engineering, Tirupathi. He then joined BITS, Pilani as a Lecturer and actively participated in developing the Computer Science Department. He, along with Prof. Mandke, the then Deputy Director of BITS, Pilani developed many courses in new emerging areas such as Artificial Intelligence, Neural Networks, Robotics, Pattern Recognition, Image Processing, Expert Systems, etc. In 1992, Prof. Venkateswarlu got his Ph.d from BITS, Pilani in "Parallel Image Processing Algorithms". From 1993 Jan. to mid of 1995, he worked at the University of Leeds, UK as a visiting fellow.

He has published many books such as *Advanced Unix Programming, Unix and Windows NT, Linux Programming Tools Unveiled* from BS Publishers Hyderabad, India. His book *C and Data Structures: A Snap Shot Oriented Treatise Using Live Engineering Examples* from S Chand is a very popular title.

He has to his credit more than 25 articles in international journals and is a referee to journals such as *Pattern Recognition, Computer Vision and Graphics*, etc.

At the end of 1995, he started the RITCH Center at Visakhapatnam to impart training in computers for the general masses in and around Visakhapatnam.

Presently, he is a senior professor of Computer Science at AITAM, Tekkali. He also served as member of Board of Studies, JNTU, Kakinada, Andhra University, Visakhapatnam. He worked as a member of National Resource Centre on Free Open Source Software (NRC-FOSS) which is under the Ministry of Information Technology, Government of India.

McGRAW HILL EDUCATION WINNING WE EDGE SERIES

Computer Science and Information Technology

CS & IT *for* GATE

GRADUATE APTITUDE TEST in ENGINEERING

N B Venkateswarlu
Senior Professor
AITAM, Tekkali

McGraw Hill Education (India) Private Limited
NEW DELHI

McGraw Hill Education Offices

New Delhi New York St Louis San Francisco Auckland Bogotá Caracas
Kuala Lumpur Lisbon London Madrid Mexico City Milan Montreal
San Juan Santiago Singapore Sydney Tokyo Toronto

McGraw Hill Education (India) Private Limited

Published by McGraw Hill Education (India) Private Limited,
P-24, Green Park Extension, New Delhi 110 016.

CS & IT for GATE

ISBN (13): 978-1-25-902720-8
ISBN (10): 1-25-902720-1

Vice President and Managing Director—McGraw-Hill Education: *Ajay Shukla*
Deputy General Manager—Test Prep and School: *Tanmoy Roychowdhury*
Publishing Manager—Test Prep: *K N Prakash*
Assistant Sponsoring Editor—*Bhavna Malhotra*
Asst Manager (Developmental Editing): *Anubha Srivastava*
Asst Manager—Production: *Medha Arora*
Senior Production Executive: *Dharmender Sharma*
Product Specialist: *Vikas Sharma*
General Manager—Production: *Rajender P. Ghansela*
Manager—Production: *Reji Kumar*

Typeset at Script Makers, 19, A1-B, DDA Market, Paschim Vihar, New Delhi 110 063, and text and cover printed at Gopsons, A-2&3, Sector-64, Noida, U.P. 201301

Cover Designer: K Anoop

RQQYCRYOLZDQR

Dedicated to
Padmasree B.L. Deekshitulu,
The doyen in IT and
Remote Sensing Development in India

Preface

It is with great pleasure that I place before you this book *CS & IT for GATE*.

I have been actively teaching computer science/IT related courses ever since my teaching career began at BITS, Pilani. When I started RITCH CENTER at Visakhapatnam to impart computer education to the masses in and around Visakhapatnam, it became inevitable for me to equip myself to be competent to teach hard core computer science courses.

During this period, I used to guide many students for the GATE and some of them got good ranks even as early as 2000. Motivated by this, I started spending time and energy in preparing material to ensure that some of my brightest students cleared GATE.

When some executives of McGraw Hill Education approached me and suggested to release my compiled teaching material as a book, I welcomed the idea and that is how this book was conceived. Till recently, only a half handful of good books are available in the market to guide GATE aspirants.

But during the last two years, the number of students who are appearing for GATE has increased tremendously because of introduction of GATE qualification in majority of public sector companies. This has naturally increased the scope for publishing high quality books in this area.

I therefore accepted the proposal from McGraw Hill Education to bring out a comprehensive manual for the aspirants of Computer Science in GATE. The book before you is a judicious blend of theoretical concepts and plenty of numerical problems for practice.

There are hundreds of numerical examples which is based on the concepts explained in detail throughout the book.

The syllabus for GATE Computer Science paper contains computer architecture, data structures, automata theory, operating systems, computer networks, software engineering, compiler construction, principles of programming languages, algorithms, engineering mathematics along with general English and general abilities.

The quality of computer science faculty is not uniform across the country. Thus students are not getting proper exposure to the concepts in their respective colleges. The GATE syllabus tests the fundamental knowledge of the candidates. Moreover, data structures, algorithms, automata theory, operating systems, computer networks are the ones which decides the GATE rank of a student. Thus, I have taken utmost care in supplying theoretical background and numerical examples such that the candidate will be ready to face even the most difficult questions in the above-mentioned areas.

In every chapter, I have added objective questions (with answers) that vary from most easy to most difficult. Also, I have included many descriptive questions along with answers. In addition, in each chapter I have included recent years' GATE questions along with explanatory answers.

Apart from GATE, this book will be also used for advanced GRE in Computer Science and UGC NET examinations. I am hopeful that aspirants would find this book a valuable companion during their preparation.

I wish the aspirants the very best in all their endeavours.

Prof. N B Venkateswarlu

Acknowledgements

I am profoundly indebted to the students who had undergone training at RITCH center, Visakhapatnam. I am especially thankful to my wife Dr Sarada and daughter Apurupa for their patient support to me during the preparation of the manuscript.

Thanks are also due to Prof. B R Gandhi, former professor of BHU, Varanasi and Prof. Roger D Boyle, University of Leeds, UK and also to many of of his fellow teachers and colleagues for their constant inspiration.

I also wish to put on record my thanks to the following:

Dr Someswara Rao: Chairman AITAM
L L Naidu: Secretary AITAM
Prof. VVN Rao: Director, AITAM
Prof. JVR Murthy, Prof Srinivasa Kumar, Mr MHM Krishan Prasad: JNTU, Kakinada

Also, my thanks are due to Dr Sahu, Mr. Dharmajee, Mr. Ch Ramesh, Mr.G. Nageswara Rao, and other staff members of CSE/IT of AITAM College. I would also like to thank Mr. Naga Tirumal Rao, Mr. BG Reddy, Mr. Bharat, Mr Ramu, Mr. Ravi, Mr. Nagendra, Mr. Giridhar, Mr. Raj Kishore, Mrs. Sunitha, Mr. Ramesh, Mr. Satpathy, Mr. B Tirumal Rao, Mr. RVV Muralikrishna, Mr. Subramanyam, Mr. Chandrasekhar (RVRJC), Mr. Hidayatullah. Dr. Surya Rao, Mdm. Jaya Rao, Mr. Achari, Dr. Sambhu Prasad, Mr. Govindarajulu and Mr. Bhaskar.

Finally, I would like to acknowledge the whole hearted support of the staff of McGraw Hill Education India in particular Mr Kannath Prakash (Publishing Manager) and Ms Medha Arora (Assistant Production Manager—Editorial), who spent considerable time and efforts in coordinating with me to bring out the book in time.

Prof. N B Venkateswarlu

Crossing the GATE
(Graduate Aptitude Test in Engineering)

(GATE) is an All India Examination conducted and administered by the Indian Institute of Science and seven Indian Institutes of Technology. It is conducted by the National Coordination Board GATE, Department of Higher Education, Ministry of Human Resource Development, Govt of India.

GATE is conducted through the constitution of eight zones. The zones and the corresponding administrative institutes are:

Zone-1: Indian Institute of Science, Bangalore
Zone-2: Indian Institute of Technology, Bombay
Zone-3: Indian Institute of Technology, Delhi
Zone-4: Indian Institute of Technology, Guwahati
Zone-5: Indian Institute of Technology, Kanpur
Zone-6: Indian Institute of Technology, Kharagpur
Zone-7: Indian Institute of Technology, Madras
Zone-8: Indian Institute of Technology, Roorkee

In the present competitive scenario, where universities and engineering colleges are mushrooming at every nook and corner of the country, the only yardstick to measure and test the actual calibre of engineering students is the GATE.

There is a general misconception among students that GATE exam is only meant for ME./M.Tech courses that finally result in a teaching career.

In this context, the following recent developments and points may be noted about GATE.

1. Many Public Sector Undertakings (PSUs totalling 217 in number) are using the GATE score for selecting candidates for their organizations. Examples are BHEL, Indian Oil Corporation Limited, NTPC, Bhabha Atomic Research Centre, Power Grid Corporation of India Limited, etc..
2. Students who qualify in GATE are entitled to a stipend of ₹ 8000/- per month during their M Tech course.
3. Better remuneration is being offered for students of M.Tech/M.E as compared to those of B.Tech /B.E. A good GATE rank assures a good job. After joining M.Tech. at IITs and IISc, one can look at a salary package ranging from 7 lakh to 30 lakh per annum, depending upon specialization and performance.
4. Clearing GATE is also an eligibility clause for the award of Junior Research Fellowship in CSIR Laboratories. M.Tech. degree is mandatory for those wishing to apply for research positions in R&D centers.
5. GATE qualified are eligible for doing Masters Degree from NUS (National University of Singapore), Singapore.
6. GATE score is valid for 2 years.
7. A GATE score has definitely an edge when it comes to joining reputed companies as well as off-campus recruitments.
8. For those who could not study their BTech in IIT, it provides another opportunity to study in the prestigious IITs.
9. Above all, it certainly gives the aspirant a huge technical edge over others in all his/her interviews and career planning.
10. Availability of GATE application form : Mid of September every year
11. Last date for submitting GATE application form: End of October every year
12. GATE exam date: Generally held on the Second Sunday of February every year[1]

[1]Please check the official website for notification and exact dates and schedule

In recent years the number of aspirants taking the GATE has grown significantly. From 1.66 lakh aspirants in 2008, the number of aspirants touched 5.5 lakh in 2011 and touched 8 lakh in 2013. GATE examination is one of the toughest examinations in our country where competition is very high and one that requires focused study in a planned manner.

It aims at rigorous testing of the students' capability in engineering concepts along with managerial skills. Engineering subjects cover 70% weightage while General aptitude and Engineering Mathematics cover 15% respectively.

To secure a high a percentile one should remember that an individual is being judged relatively and not absolutely. The overall rank achieved depends upon the preparation level of your competitors.

Examination Pattern

The three-hour GATE paper has a total of 65 questions carrying 100 marks. Q.1 to Q.25 (25 questions) carry one mark each (sub-total 25 marks). Q.26 to Q.55 (30 questions) carry two marks each (sub-total 60 marks). Questions Q.56 – Q.65 belong to General Aptitude (GA). Questions Q.56 – Q.60 (5 questions) carry 1 mark each (sub-total 5 marks) and questions Q.61 – Q.65 (5 questions) carry 2-marks each (sub-total 10 marks). Questions Q.48 – Q.51 (2 pairs) are common data questions. Question pairs (Q.52, Q.53) and (Q.54, Q.55) are linked answer questions. The answer to the second question of the linked answer questions depends on the answer to the first question of the pair. If the first question in the linked pair is wrongly answered or is unattempted, then the answer to the second question in the pair will not be evaluated.

NEGATIVE MARKING: For Q.1 – Q.25 and Q.56 – Q.60, 1/3 mark will be deducted for each wrong answer. For Q.26 – Q.51 and Q.61 – Q.65, 2/3 mark will be deducted for each wrong answer. The question pairs (Q.52, Q.53), and (Q.54, Q.55) are questions with linked answers. There will be negative marks only for wrong answer to the first question of the linked answer question pair, i.e., for Q.52 and Q.54, 2/3 mark will be deducted for each wrong answer. There is no negative marking for Q.53 and Q.55.

Questions on Engineering Mathematics will carry about 15% of the total marks (excluding General Aptitude section).

GATE RESULT

GATE scores are valid for two years and one may reappear if he or she is not satisfied with the earlier score and new scores are used for the admission purposes.

Calculation of GATE Score

The GATE Score is calculated as

$$S = S_q + (S_t - S_q)\frac{M - M_q}{\overline{M}_t - M_q},$$

where,

S = GATE Score (normalized) of a candidate,

S_q = GATE Score assigned to M_q (around 300).

S_t = GATE Score assigned to $\bar{M}_t$ (around 900),

M = Marks obtained by a candidate in a paper,

M_q = Qualifying Marks for general category candidates in the paper,

$\bar{M}_t$ = Average Marks of top 0.1% or 10 (which ever is higher) of candidates in the paper,

M_q is usually 25 marks(out of 100) or $\mu + \sigma$, which ever is higher. μ is the mean of marks in a paper and s is standard deviation.

But at the same time as competition grow, the GATE Score along with the rank matters for induction into PSUs' jobs/ M.Tech programmes.

Tips for Success in GATE

1. To start with, go through previous years' question papers for the last 10 years along with solutions.
2. Analyze the subject pattern and focus on those subjects which have maximum weightage. During the last 25 years, it is observed that questions related to algorithms, data structures, compiler construction, automata theory, operating systems were instrumental in the aspirants' success. So I advise you to build a good foundation in these topics.
3. Since this is an academically high level exam, rely on quality books and study material.
4. *Books can further be divided into two categories*
   (a) Books that deal with the fundamentals and focus on conceptual clarity. Here textbooks by reputed publishers are a must.
   (b) Books that provide a great deal of difficult and time consuming questions and are used essentially as practice material
5. Don't rely on just one book for a topic. You may have to consult a couple of books for the same topic.
6. Prepare notes after completing each chapter. These can also be made either during self study or during coaching classes.
7. Practice the maximum number of questions possible on a given topic. This certainly strengthens your preparation.
8. There is no need to cover entire syllabus. Topics which are not in GATE syllabus should be certainly left out.
9. Make a list of topics in which you think you are 'weak' and focus on them.
10. Have all essential formulae on your fingertips.
11. Try to see if there are shortcut methods for a particular problem.
12. Joining a good coaching institute is beneficial as you would be exposed to a regular systematic study. Also, an exposure to peer group would make one more competitive.
13. Online coaching classes/test series are also beneficial in case one cannot join the regular coaching classes.
14. Don't hesitate to consult seniors and professors incase of any doubts or clarification.
15. Do not neglect General English and Aptitude.
16. Theory preparation should be finished one month before the exam and then practice, practice, practice.
17. Scores can be assessed from mock tests which are available online and offline
18. One should keep oneself updated about any changes or developments in the GATE examination for the coming year.
19. In the last one week, go through last 10 years' solutions once again as questions may be repeated and also previous GRE question papers on the subject.

How to tackle the paper

- Start the paper with 1 mark questions (25 in number). Since these are easy to attempt, they will help in building confidence.
- Proceed then to 2 marks questions from Common Data and Linked Answer Questions. Attempt this part with caution. This portion of exam is certainly instrumental in making the final merit. These add upto 8 more questions in addition to the 25 attempted above.
- Then go to General Aptitude section of 10 questions. All these will add upto 43 questions attempted. All these should be done in a time frame of 100–110 mins.
- Now we are left with 22 questions in technical portion and we can allot 50 minutes to this comfortably.
- At the end, you must have 15–20 minutes for a quick revision of the answer sheet to ensure all is in order.

Note:

1. *While attempting the paper, leave questions about which you are not sure. The most deciding factor is negative marking. Avoid making any guesses and try to eliminate choices by analysis and calculations.*
2. *Aspirants should ensure that they should apply for GATE examinations preferably from their respective branch only, i.e. candidate with mechanical engineering should preferably apply for Mechanical Engg only and should avoid moving to other branches like Production and Industrial Engineering, etc. Most of the PSUs induct trainees with their GATE score from their respective B. Tech branch only.*

   For example, please refer to site **www.iocl.com** *for similar details.*

Syllabus

ENGINEERING MATHEMATICS

Mathematical Logic: Propositional Logic; First Order Logic.

Probability: Conditional Probability; Mean, Median, Mode and Standard Deviation; Random Variables; Distributions; uniform, normal, exponential, Poisson, Binomial.

Set Theory & Algebra: Sets; Relations; Functions; Groups; Partial Orders; Lattice; Boolean Algebra.

Combinatory: Permutations; Combinations; Counting; Summation; generating functions; recurrence relations; asymptotics.

Graph Theory: Connectivity; spanning trees; Cut vertices & edges; covering; matching; independent sets; Colouring; Planarity; Isomorphism.

Linear Algebra: Algebra of matrices, determinants, systems of linear equations, Eigen values and Eigen vectors.

Numerical Methods: LU decomposition for systems of linear equations; numerical solutions of non-linear algebraic equations by Secant, Bisection and Newton-Raphson Methods; Numerical integration by trapezoidal and Simpson's rules.

Calculus: Limit, Continuity & differentiability, Mean value Theorems, Theorems of integral calculus, evaluation of definite & improper integrals, Partial derivatives, Total derivatives, maxima & minima.

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

Digital Logic: Logic functions, Minimization, Design and synthesis of combinational and sequential circuits; Number representation and computer arithmetic (fixed and floating point).

Computer Organization and Architecture: Machine instructions and addressing modes, ALU and data-path, CPU control design, Memory interface, I/O interface (Interrupt and DMA mode), Instruction pipelining, Cache and main memory, Secondary storage.

Programming and Data Structures: Programming in C; Functions, Recursion, Parameter passing, Scope, Binding; Abstract data types, Arrays, Stacks, Queues, Linked Lists, Trees, Binary search trees, Binary heaps.

Algorithms: Analysis, Asymptotic notation, Notions of space and time complexity, Worst and average case analysis; Design: Greedy approach, Dynamic programming, Divide-and-conquer; Tree and graph traversals, Connected components, Spanning trees, Shortest paths; Hashing, Sorting, Searching. Asymptotic analysis (best, worst, average cases) of time and space, upper and lower bounds, Basic concepts of complexity classes – P, NP, NP-hard, NP-complete.

Theory of Computation: Regular languages and finite automata, Context free languages and Push-down automata, Recursively enumerable sets and Turing machines, Undecidability.

Compiler Design: Lexical analysis, Parsing, Syntax directed translation, Runtime environments, Intermediate and target code generation, Basics of code optimization.

Operating System: Processes, Threads, Inter-process communication, Concurrency, Synchronization, Deadlock, CPU scheduling, Memory management and virtual memory, File systems, I/O systems, Protection and security.

Databases: ER-model, Relational model (relational algebra, tuple calculus), Database design (integrity constraints, normal forms), Query languages (SQL), File structures (sequential files, indexing, B and B+ trees), Transactions and concurrency control.

Information Systems and Software Engineering: information gathering, requirement and feasibility analysis, data flow diagrams, process specifications, input/output design, process life cycle, planning and managing the project, design, coding, testing, implementation, maintenance.

Computer Networks: ISO/OSI stack, LAN technologies (Ethernet, Token ring), Flow and error control techniques, Routing algorithms, Congestion control, TCP/UDP and sockets, IP(v4), Application layer protocols (icmp, dns, smtp, pop, ftp, http); Basic concepts of hubs, switches, gateways, and routers. Network security – basic concepts of public key and private key cryptography, digital signature, firewalls.

Web technologies: HTML, XML, basic concepts of client-server computing.

Contents

CHAPTER ONE

Introductory Concepts of Digital Logic Design and Computer Architecture

1.1 Analog and Digital Systems

Let us explain the difference between analog and digital systems with the help of some examples which we may find in our daily life. A good example is none other than our fan (rather, its control mechanism). In old fans, fan speed can be specified (controlled) with a knob which contains numbers 0–5. When we keep the knob at 0, fan will not run; at all other places it runs with slow to fast speeds. However, if we keep it in between two numbers, it will not run at all. That is, the fan speed is defined at some predefined numbers (places). If we keep fan knob between any of the two numbers, the fan will automatically get stopped. That is, fan speed is defined at locations 1 to 5; but not in between any of the two numbers. This type of system is called as Discrete (or digital) System. Now, let us consider fans that are available today in which we can control speed by turning a knob (without any numbers). We can change the speed, the way we like. This type of system can be called as Analog or Continuous System. Analog systems process analog signals which can take any value within a range, for example the output from an LDR (light sensor) or a microphone. Digital systems process digital signals which can take only a limited number of values (discrete values), usually just two values are used: the positive voltage (+V) and zero volts (0V).

Advantages of Digital Systems

1. Easier to design. Exact values of voltage or current are not important, only the range (HIGH or LOW) in which they fall.
2. Accuracy and precision are greater. We can predict accuracy of a digital system by a method of reasoning (logic), by understanding the program or by verifying its truth table or otherwise. Sometimes we can also pin-point the sequence of events a digital device will go through in its general functioning. The simplicity of these devices and the principle underlying makes all the difference. So we can most of the times predict how the device will function in a given condition and also we can set the extreme conditions and can also specify how the system will respond in such a condition. So digital systems have nowadays got very much acceptance and reliability and people are switching to digitals.
3. Operation can be programmed. Analog systems can also be programmed, but the variety and complexity of the available operations is severely limited.
4. Digital circuits are less affected by noise. As long as the noise is not large enough to prevent us from distinguishing a High from a Low.
5. More digital circuitry can be fabricated on IC chips.
6. Digital systems interface work well with computer and are easy to control with SW. New features can often be added to a digital system without changing hardware. Often this can be done outside of the factory by updating the product's software. So, the product's design errors can be corrected after the product is in a customer's hands.
7. Information storage can be easier in digital systems than in analog ones. The noise immunity of digital systems permit data to be stored and retrieved without

degradation. In an analog system, noise from aging and wear degrade the information stored. In a digital system, as long as the total noise is below a certain level, the information can be recovered perfectly.

8. One of the primary advantages of digital electronics is its robustness. Digital electronics is robust because if the noise is less than the noise margin then the system performs as if there is no noise at all. Therefore, digital signals can be regenerated to achieve lossless data transmission, within certain limits. Analog signal transmission and processing, by contrast, always introduces noise

Certainly benefits of a system are drawbacks of its opponent. That is all the positive aspects of digital systems are negative points of analog system. However, analog systems will continue to survive because there is speed limitation with digital systems due to their fundamental discrete nature. Analog will still be preferred at very high frequencies and sometimes the only possibility in some cases.

Limitations of Digital Systems

In some specific realisations of digital systems, power consumption is more as compared to their analog counter parts. Also, fabricating a digital system in limited quantities is expensive compared their analog equivalents.

There is really only one major drawback when using digital techniques: The real world is mainly analog. Thus, how nicely we can deal with real world problem very much depends on how accurately we are able represent the real world information in digital form.

Sampling and Quantisation

Because we can not record every point in the original sound wave, we have to sample. We sample at some regular interval called the sampling rate. Faster this rate, more accurate will be the reproduction. What rate do we need to sample at? What are limits of human hearing? Most people can hear sounds up to approximately 20KHz. Based on something called the Nyquist Rule, which says that the sampling rate must be at least twice the highest frequency in the sample, we need to sample at least 40K times per second. For digital audio, the sampling rate is 44,100 Hz.

Analog to Digital Converter (ADC) and Digital to Analog Converter (DAC)

In practice, sound waves are captured by a microphone, which converts the sound pressure to an analog electrical signal which has some equivalency with analog sound wave in terms of its voltage or some other property. An ADC samples this signal and produces a binary number for each sample. To play the sound, we pass these binary numbers to a DAC, which converts each sample back to its analog level. The DAC also interpolates the samples, creating a smooth electrical signal that is sent to a speaker, which converts it to a sound wave.

Number of Bits Per Sample

Let us say we want to use 3 bits to represent the value. The first bit will represent + or –. Then we can use the other two bits to represent the magnitude of the signal. The more bits you use, the more accurately we can reproduce the original sound. CDs use 16 bits.

How much memory is required to record 1 second of music?

16 bits/sample x 44,100 samples/sec x 2 (stereo) = 1,411,200 bits/sec = 176,400 bytes/sec

1.2 Basic Logic Gates

There are several kinds of logic gates, each one of which performs a specific function. These are the: (1) **AND** gate; (2) **OR** gate; (3) **NOT** gate; (4) **NAND** gate; (5) **NOR** gate; and (6) **XOR** gate.

Table 1.1 Logic Gates and their properties

<table>
<tr><th>Gate</th><th>Description</th><th>Truth Table</th></tr>
<tr><td>AND Gate</td><td>The AND gate is a logic gate that gives an output of ‘1’ only when all of its inputs are ‘1’. Thus, its output is ‘0’ whenever at least one of its inputs is ‘0’. Mathematically, Q = A · B.</td><td>
<table>
<tr><th>A</th><th>B</th><th>Output Q</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>OR Gate</td><td>The OR gate is a logic gate that gives an output of ‘0’ only when all of its inputs are ‘0’. Thus, its output is ‘1’ whenever at least one of its inputs is ‘1’. Mathematically, Q = A + B.</td><td>
<table>
<tr><th>A</th><th>B</th><th>Output Q</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>NOT Gate</td><td>The NOT gate is a logic gate that gives an output that is opposite the state of its input. Mathematically, Q = A’ or Q=</td><td>
<table>
<tr><th>A</th><th>Output Q</th></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td></tr>
</table>
</td></tr>
</table>

Gate	Description	Truth Table
NAND Gate	The NAND gate is an AND gate with a NOT gate at its end. Thus, for the same combination of inputs, the output of a NAND gate will be opposite that of an AND gate. Mathematically, Q = A · B.	A B Output Q 0 0 1 0 1 1 1 0 1 1 1 0
NOR Gate	The NOR gate is an OR gate with a NOT gate at its end. Thus, for the same combination of inputs, the output of a NOR gate will be opposite that of an OR gate. Mathematically, Q = A + B.	A B Output Q 0 0 1 0 1 0 1 0 0 1 1 0
XOR Gate	The XOR gate (for 'EXclusive OR' gate) is a logic gate that gives an output of '1' when only one of its inputs is '1'. Rather if both the inputs are same then output is 0; else output is 1.	A B Output Q 0 0 0 0 1 1 1 0 1 1 1 0
X-NOR Gate	The X-NOR gate (for 'EXclusive NOR' gate) is a logic gate that gives an output of '1' when only both of its inputs are same else outputs 0.	A B Output Q 0 0 1 0 1 0 1 0 0 1 1 1

Figure 1.1 shows the general pictorial representation of the above elementary logic gates.

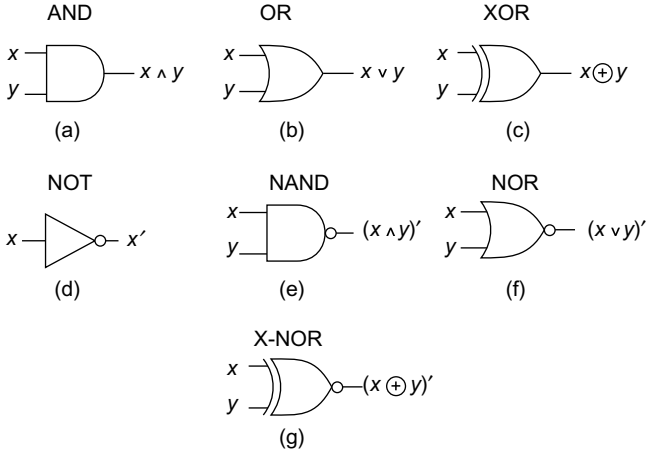
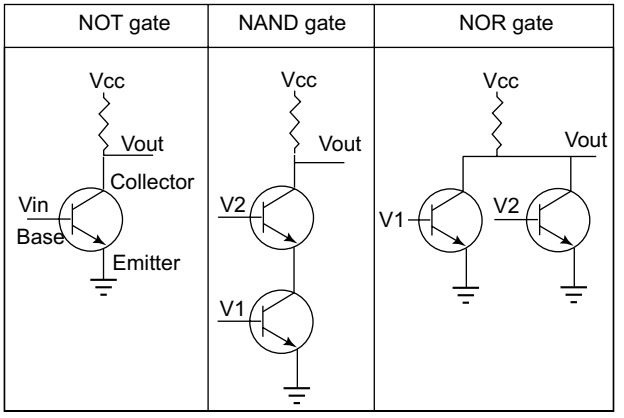


Figure 1.1 Symbols used to Represent Elementary Gates

In fact, logic Gates are made from transistors. For example, the following illustration contains information about NOT, NAND and NOR gates. In a nutshell, NAND and NOR can be realised using two transistors whereas AND, OR needs three. For better details, we advise readers to refer books on basic electronics. However, for the sake of completeness we are presenting those details. Even if you don't understand them, don't worry.



Exclusive OR gate also called XOR: a pair of ANDs with one inverted input into an OR gate.

$$A \times B \text{ or } A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$$

The above truth table and figure demonstrates the equivalence of the above relationships. In a physical sense, output of an XOR gate will be one only when both the inputs are opposite or complimentary. Thus, its output equation is given as $\overline{A} \cdot B + A \cdot \overline{B}$; if one verifies, the Boolean expression will attain true value if A=1, B=0 or A=0, B=1. For all other combinations, this Boolean equation value will be false.

NAND and NOR gates are called as Universal gates as any other gate can be realised using these gates.

1.2.1 Functionally Complete

For gates to be useful we must be able to use them to implement any Boolean function; we call a set of gates that can implement any function a "functionally complete" set of gates. The following are functionally complete sets: {AND, OR, NOT}, {AND, NOT}, {OR, NOT}, {NAND}, {NOR}. The first set is functionally complete because it contains the fundamental operations of Boolean algebra. For each of the other sets, we can prove that they are functionally complete by showing how to implement the fundamental operations (AND, OR, and NOT) using the operations in the set under consideration.

1.3 Boolean Theorems and Postulates

A set of rules formulated by the English mathematician *George Boole* describe certain propositions whose outcome would be either *true or false*. With regard to digital logic, these rules are used to describe circuits whose state can be either, *1 (true) or 0 (false)*. In order to fully understand this, the relation between AND gate, OR gate, and NOT gate operations should be appreciated. A number of rules can be derived from these relations such as:

- **P1: X = 0 or X = 1**
- **P2: 0 . 0 = 0**
- **P3: 1 + 1 = 1**
- **P4: 0 + 0 = 0**
- **P5: 1 . 1 = 1**
- **P6: 1 . 0 = 0 . 1 = 0**
- **P7: 1 + 0 = 0 + 1 = 1**

The following is a list of basic Boolean laws proposed by Bool and other scientists. Note that every law has two expressions, (a) and (b). This is known as *duality*. These are obtained by changing every AND(.) to OR(+), every OR(+) to AND(.) and all 1's to 0's and vice versa. It has become conventional to drop the . (AND symbol), i.e. A.B is written as AB.

Closure

If X and Y are in set (0, 1) then operations X + Y and X · Y are also in set (0, 1)

T1 : Commutative Law

(a) $A + B = B + A$

(b) $A\,B = B\,A$

T2 : Associate Law

(a) $(A + B) + C = A + (B + C)$

(b) $(A\,B)\,C = A\,(B\,C)$

T3 : Distributive Law

(a) $A\,(B + C) = A\,B + A\,C$

(b) $+\,(B\,C) = (A + B)\,(A + C)$

T4 : Identity Law

(a) $A + A = A$

(b) $A\,A = A$

T5 :

(a) $A\,\overline{B} + AB = A$

(b) $(A+B)\,(A+\overline{B}) = A$

T6 : Redundancy Law

(a) $A + A\,B = A$

(b) $A\,(A + B) = A$

T7 :

(a) $0 + A = A$

(b) $0\,A = 0$

T8 :

(a) $1 + A = 1$

(b) $A = A$

T9 :

(a) $\overline{A}\ A = 0$

(b) $\overline{A} + A = 1$

T10 :

(a) $A + \overline{A}\,B = A + B$

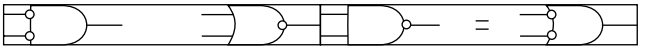
(b) $A(\overline{A} + B) = AB$

T11 : De Morgan's Theorem

(a) $\overline{(A+B)} = \overline{A}\ \overline{B}$

(b) $\overline{(AB)} = \overline{A} + \overline{B}$

That is, the following are equivalent circuits according to De Morgan.



■ **Example** Prove $A + \overline{A}\ B = A + B$

(1) Algebraically:

$A + \overline{A}\ B =$	$A\,1 + \overline{A}\,B$	T7(a)
	$= A\,(1 + B) + \overline{A}\,B$	T7(c)
	$= A\,\lvert AB : \overline{A}\,B$	T3(a)
	$= A + B\,(A + \overline{A})$	T3(a)
	$= A + B$	T(8)

(2) Using the truth table:

A	B	$A+B$	$\overline{A}\,B$	$A + \overline{A}\,B$
0	0	**0**	0	**0**
0	1	**1**	1	**1**
1	0	**1**	0	**1**
1	1	**1**	0	**1**

Using the laws given above, complicated expressions can be simplified as:

$$Z = (A + \overline{B} + \overline{C})\,(A + \overline{B}\,C)$$

$$Z = AA + A\,\overline{B}\,C + A\,\overline{B} + \overline{B}\ \overline{B}\,C + A\,\overline{C} + \overline{B}\,C\,\overline{C}$$

$$Z = A\,(1 + \overline{B}\,C + \overline{B} + \overline{C}) + \overline{B}\,C + \overline{B}\,C\,\overline{C}$$

from laws T8b and Tb

$Z = A + \overline{B}\,C$ from laws T8a, T8b and T9b.

1.3.1 Precedence Rules

In order to reduce the number of parentheses, we have the following set of precedence rules that indicates which sub-expression or operator to evaluate next:

- Evaluate the expression inside a pair of parentheses.
- Evaluate NOT
- Evaluate AND
- Evaluate OR

1.3.2 Evaluating Boolean Expressions

In general, the following rules must always be followed when evaluating a Boolean expressions.

1. First, perform all inversions of single terms; that is, 0 = 1 or 1 = 0, including the ones in parenthesis also.
2. Then perform all operations within parentheses.
3. Perform an AND operation before an OR operation unless parentheses indicate otherwise.
4. If an expression has a bar over it, perform the operations of the expression first and then invert the result.

■ **Example** Precedence Rules Illustration using of expression NOT(1 OR 0) OR 1 AND 0

Step	Expression	Explanation
1	NOT(1 OR 0) OR 1 AND 0	1 OR 0 is inside the brackets, so evaluate it first. The result is 1, so replace (1 OR 0) with 1.
2	= NOT(1) OR 1 AND 0	Evaluate the complement next. NOT(1) = 0. Replace NOT(1) with 0.
3	= 0 OR 1 AND 0	Evaluate the product next. 1 AND 0 = 0. Replace 1 AND 0 with 0.
4	= 0 OR 0	Now, evaluate the sum. 0 OR 0 = 0, so the result of the expression is 0.
5	= 0	We are done.

1.3.3 Special-output Gates

It is sometimes desirable to have a logic gate that provides both inverted and non-inverted outputs. For example, a single-input gate that is both a buffer and an inverter, with a separate output terminal for each function. Or, a two-input gate that provides both the AND and the NAND functions in a single circuit. Such gates do exist and they are referred to as *complementary output gates*.

The general symbolic notation for such a gate is the basic gate figure with a bar and two output lines protruding from it. An array of complementary gate symbols is shown in the following illustration (Figure 1.2).

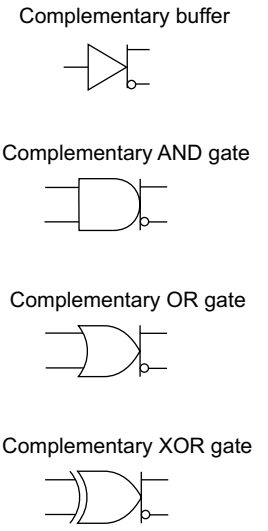


Figure 1.2 Complimentary Gates

Complementary gates are especially useful in "crowded" circuits where there may not be enough physical room to mount the additional integrated circuit chips necessary to provide both inverted and non-inverted outputs using standard gates and additional inverters. They are also useful in applications where a complementary output is necessary from a gate, but the addition of an inverter would introduce an unwanted time lag in the inverted output relative to the non-inverted output. The internal circuitry of complemented gates is such that both inverted and non-inverted outputs change state at almost exactly the same time.

It is possible to implement *any* logic expression using only NAND gates and no other type of gate. This is because NAND gates, in the proper combination, can be used to perform each of the Boolean operations OR, AND, and INVERT.

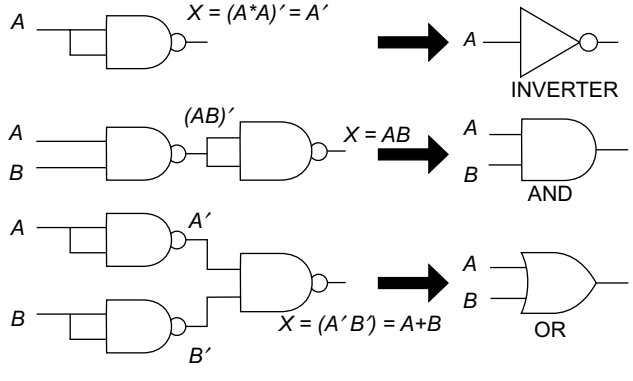


Figure 1.3 Realising basic Gates using NAND Gates

In a similar manner, it can be shown that NOR gates can be arranged to implement any of the Boolean operations.

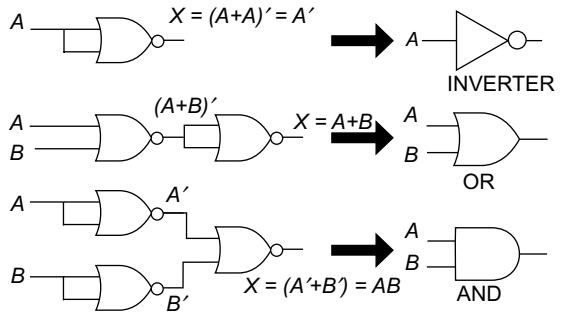


Figure 1.4 Realising basic Gates using NOR Gates

Alternate Logic Gate Representations

The following illustration shows the standard symbol for each logic gate, and the right side shows the **alternate symbol**. The alternate symbol for each gate is obtained from the standard symbol by doing the following:

1. Invert each input and output of the standard symbol. This is done by adding bubbles (small circles) on input and output lines that do not have bubbles, and by removing bubbles that are already there.
2. Change the operation symbol from AND to OR, or from OR to AND. (In the special case of the INVERTER, the operation symbol is not changed.)

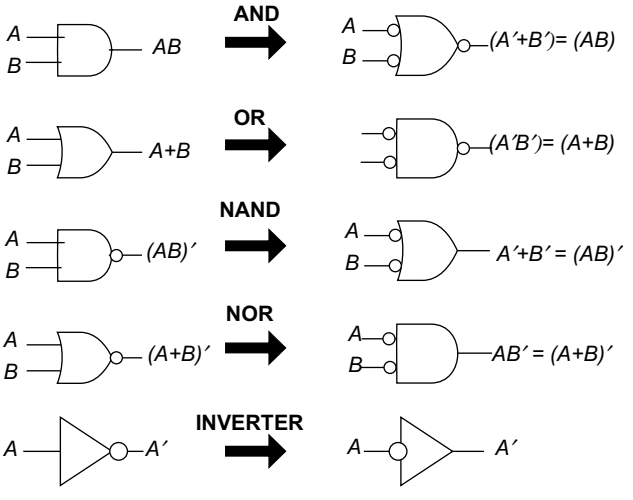


Figure 1.5 Alternate logic gates

Several points should be considered regarding the logic symbol equivalences:

1. The equivalences are valid for gates with any number of inputs.
2. None of the standard symbols have bubbles on their inputs, and all the alternate symbols do.
3. The standard and alternate symbols for each gate represent the same physical circuit: there is no difference in the circuits represented by the two symbols.
4. NAND and NOR gates are inverting gates, and so both the standard and alternate symbols for each will have a bubble on either the input or the output. AND and OR gates are non-inverting gates, and so the alternate symbols for each will have bubbles on both inputs and output.

Note

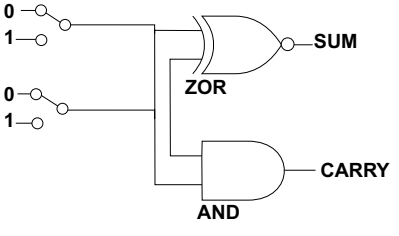
Negative-OR gate function is an OR gate with all its inputs (A, B) inverted. This is equivalent to NAND of A, B. Negative-AND gate function is an AND gate with all its inputs (A, B) inverted. This gate is equivalent to NOR. See above figures.

■ **Example Half Adder Circuit**

The main objective of this circuit is to add two binary bits together to give us a SUM and a CARRY. It is known as a half adder because it only does half the job. That is, it can generate a carry to the next column, but it cannot use (bring in) a carry from the previous column. Thus, its name is half-adder. Truth table for the same can be given as:

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

If one observes the Sum column in the above table, we can find that Sum is same as A Exclusive OR B. Similarly, if we observe the Carry column in the above table, we may find Carry is same as A AND B. Thus, the half-adder circuit can be drawn as:



■ **Example Full Adder Circuit**

Unlike half-adder, a full adder adds two binary numbers (A,B) together and includes provision for a carry in bit (Cin) and a carry out bit (Cout). That is, it can use carry bit previous position and given carry out to the next position in the binary additions. The truth table for a full adder is:

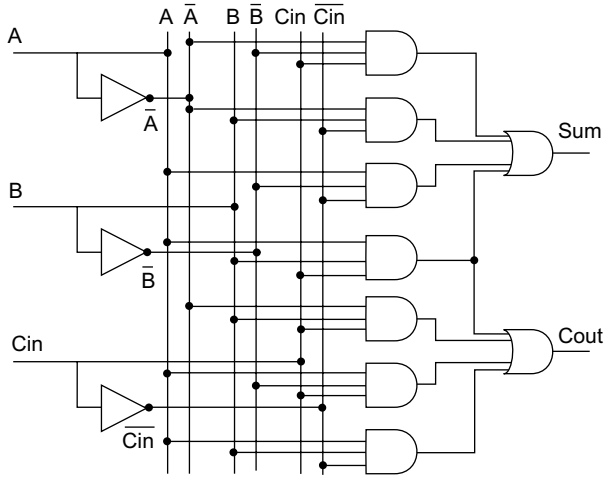
A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

By considering the 1s in the Cout and Sum columns in the above table, we can write Boolean equations for Sum and Cout in Sum of Products form as:

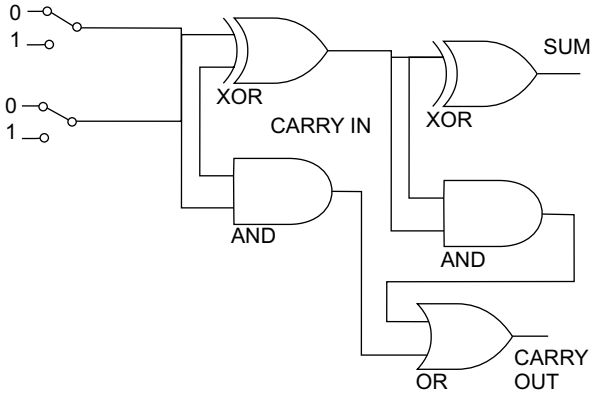
$$\text{Sum} = \overline{A}\,\overline{B}\,Cin + \overline{A}\,B\,\overline{Cin} + A\overline{B}\,\overline{Cin} + ABCin$$

$$\text{Cout} = \overline{A}\,BCin + A\overline{B}Cin + AB\overline{Cin} + ABCin$$

The resulting circuit can be given below.

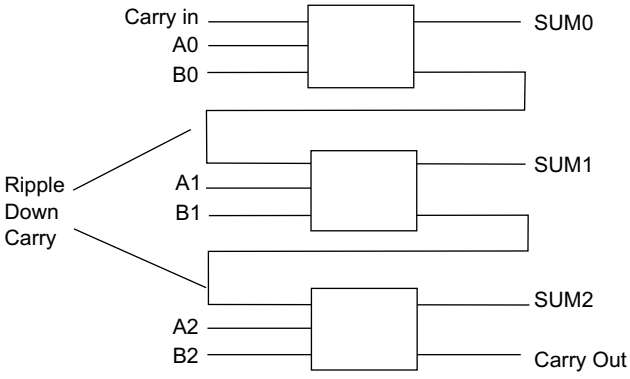


We can also create a full-adder by using two half adders as shown in this figure.



In the above circuit, first half adder works on the two data bits giving Sum and Carry. The resulting Sum and Cin are applied to next half adder circuit. The carry results of both the half adders are ended to get Cout.

We can link together 3-bit full adder circuits to give us the ability to add together two 3-bit numbers.



This is a simplification, modern computers use a technique known as 'look-ahead-carry' in order to predict the carry states, rather than waiting for the carry's to 'ripple down' through the adder.

1.3.4 Substituting one type of gate for another

Logic gates are available on ICs which usually contain several gates of the same type, for example four 2-input NAND gates or three 3-input NAND gates. This can be wasteful if only a few gates are required unless they are all the same type. To avoid using too many ICs you can reduce the number of gate inputs or substitute one type of gate for another.

1.3.5 Reducing the number of inputs

The number of inputs to a gate can be reduced by connecting two (or more) inputs together. Figure 1.6 shows a 3-input AND gate operating as a 2-input AND gate.

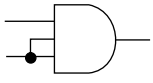


Figure 1.6 Reducing number of inputs

1.3.6 Making a NOT gate from a NAND or NOR gate

Reducing a NAND or NOR gate to just one input creates a NOT gate. Figure 1.7 shows this for a 3-input NAND gate.

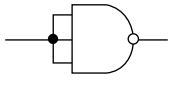


Figure 1.7 Realising a NOT gate using NAND

1.3.7 Building any Gate can be built from NAND or NOR gates

As well as making a NOT gate, NAND or NOR gates can be combined to create any type of gate! This enables a circuit to be built from just one type of gate, either NAND or NOR. For example an AND gate is a NAND gate then a NOT gate (to undo the inverting function). Note that AND and OR gates cannot be used to create other gates because they lack the inverting (NOT) function.

To change the type of gate, such as changing OR to AND, you must do three things:

- Invert (NOT) each input.
- Change the gate type (OR to AND, or AND to OR)
- Invert (NOT) the output.

For example an OR gate can be built from **NOT**ed inputs fed into a NAND (AND + NOT) gate.

■ **Example** Draw logic diagram which uses only NOR gates to realize NAND. Also, explain the Boolean mathematical explanation for the same.

■ **Answer:** We know 2-input NAND is referred to X=(AB)'

$$= A' + B'$$

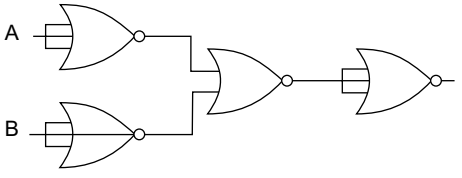
If we use the theorem A+A = A, then the above equation becomes:

$$= (A+A)' + (B+B)'$$

Also, we know A'=(A+A)'. Thus, we take ((X)')'. Thus, the equation becomes:

$$= (((A+A)' + (B+B)')')'$$

This can be shown in the following fashion.



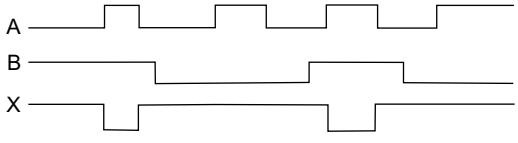
1.4 Positive Logic and Negative Logic

The terms positive logic and negative logic refer to two conventions that dictate the relationship between logical values and the physical voltages used to represent them. In positive logic notation, we consider high voltage is considered as logic 1 (or true) while low voltage as logic 0 (or false). Whereas in the case of negative logic notation, low voltage is considered as logic 1 (or true) while high voltage as logic 0 (or false). In fact, in the positive logic convention, the more positive potential is considered to represent *True* and the more negative potential is considered to represent *False* (hence, positive logic is also known as *positive-true*). By comparison, using the negative logic convention, the more negative potential is considered to represent *True* and the more positive potential is considered to represent *False* (hence, negative logic is also known as *negative-true*). No doubt that positive logic is the more intuitive as it is easy to relate logic 0 to 0V (zero or "no volts") and logic 1 to +ve (the presence of volts or "yes volts"). On this basis, one may wonder why negative logic was ever invented. The answer to this is, as are so many things, rooted in history. When the MOSFET technology was originally developed, PMOS transistors were easier to manufacture and were more reliable than their NMOS counterparts. Thus, the majority of early MOSFET-based logic gates were constructed from combinations of PMOS transistors and resistors. Circuits constructed using the original PMOS transistors typically made use of a negative power supply (that is, a power supply with a 0V terminal and a negative (–ve) terminal). With PMOS transistors, an input connected to the more positive Vss turns that transistor "off", and an input connected to the more negative potential Vdd turns that transistor "on". Once again, the final step is to define the mapping between the physical and abstract worlds; either 0v is mapped to *false* and –ve is mapped to *True*, or vice versa. Thus, negative logic is arrived into picture. Do remember that positive logic systems 1 is called as active high while 0 as active low.

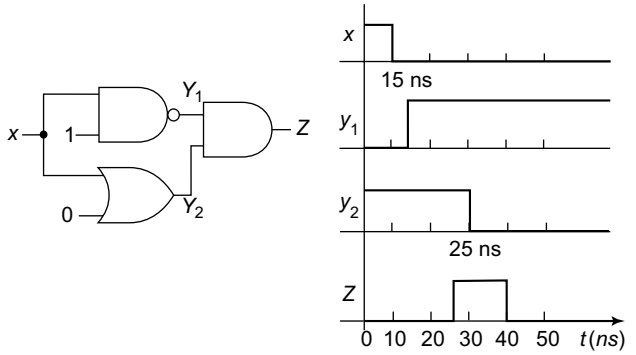
■ **Example** Why one has to minimise Boolean equations? The goal of logic expression minimisation is to find an equivalent of an original logic expression that has fewer variables per term, has fewer terms and needs less logic gates to implement. That is, to reduce number of gates such that the circuit may becomes cheap, consumes less power and increases its speed.

■ **Example** Complete the following timing diagram. Assume that A and B are inputs to a **NAND** gate,
while X is the output.

■ **Answer :** We know NAND gate output is zero if both of its inputs are 1s, otherwise its output is 1. Based on this, X is drawn as shown below.



■ **Example** Complete the timing diagram on the right based on the gate-level schematic given in the left of the following figure. Assume the following gate delays: t_{NAND} = 5 ns, t_{OR} = 20 ns, t_{AND} = 10 ns. Indicate if there are any false outputs.



■ **Answer:** According to the Boolean equation:

$$Z = (X \bullet 1)' \bullet (X + 0)$$
$$= X' \bullet X$$
$$= 0$$

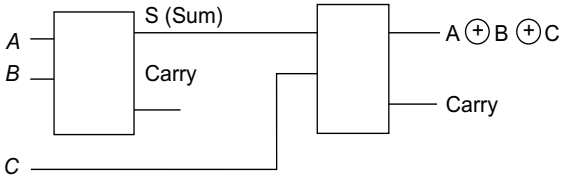
The output should always be 0. However, due to the difference in timing between the two paths from X to Z, we get a false output from $t = 25$ ns to $t = 40$ ns.

■ **Example** Implement the Boolean expression A ⊕ B ⊕ C using two half-adders.

■ **Answer:** We know half adder contains two outputs S (sum) and Cout (Carry out). Truth table for the Boolean expression A ⊕ B ⊕ C is given as:

ABC	A ⊕ B ⊕ C
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

The following circuit with half adders give us the required output as shown in the truth table.

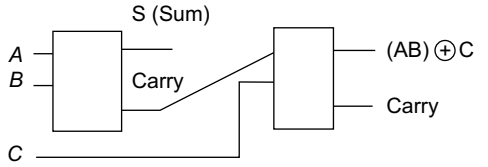


■ **Example** Implement the function (AB) ⊕ C using two half adders.

■ **Answer:** First, we shall prepare truth table for (AB) ⊕ C

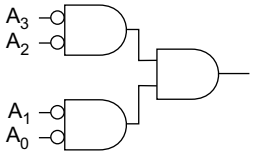
ABC	AB	(AB) ⊕ C
000	0	0
001	0	1
010	0	0
011	0	1
100	0	0
101	0	1
110	1	1
111	1	0

This can be achieved through half adders by adding Carry line first half adder as first input to second half adder as shown in the following figure.



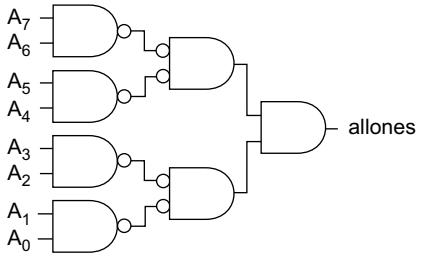
■ **Example** The following circuit is proposed to test whether the given four inputs are 0s or not. Explain its functionality.

■ **Answer:** The circuit is supposed give 1 if all of its four inputs are 0; otherwise it has to give 0. The inputs are complemented and fed to first level AND gates whose outputs are further fed to second level AND gate. If all the four inputs are 0s then outputs of first level AND gates will be 1. Thus, finally second level AND gate gives 1. In all other situations, final output of the circuit will be zero.

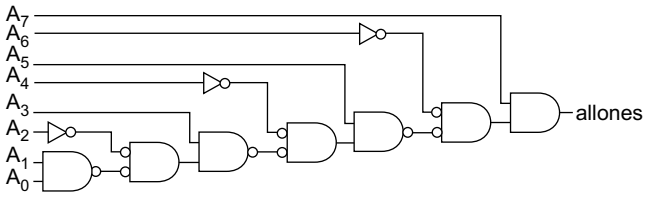


■ **Example** Explain the functionality of the following circuit. It is proposed to check whether all if inputs are 1s or not. Will it satisfy the requirement?.

■ **Answer:** The circuit is organized in three levels. In first level, we have NAND gates while in the second third levels we find AND gates. First level NAND gates gives 0s if both of their inputs are 1s. These outputs are complemented before passing through second level AND gates. Thus, second level AND gates gives 1s only when if all the eight inputs to the circuit are 1s. Thus, the find AND gate gives 1 when all the eight inputs of the circuit are 1s else gives 0.

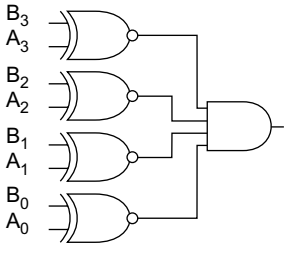


■ **Example** The following circuit is proposed as another alternative circuit for checking for all 1s. Verify its functionality. Which is better?



■ **Answer:** If one verifies, it can also serve the objective. However, this circuit needs more NOT gates and propagation delay is more compared to the previous one. Thus, the previous solution is preferred.

■ **Example** Two four bit binary numbers $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ are fed to the following circuit as shown below to check their equality. Does the given circuit will satisfy the objective? Explain. Also, what happens if exclusive-NOR gates are substitutes with exclusive-OR gates?

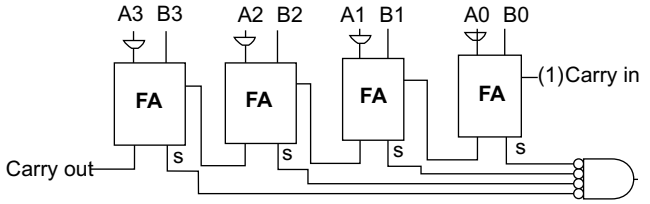


■ **Answer:** If we observe the circuit, we may not find exclusive-NOR gates in the first level for which respective bits of both the numbers are fed. Only if both the bits are same then exclusive-NOR gates gives 1 as their output. The second level AND gates gives 1 if all of its inputs are 1s. That is, if all the respective bits of both the numbers are same, AND gate output will be 1 otherwise 0. That is, if both the numbers are same then the circuit output will be 1, else output will be 0.

If we replace exclusive-NOR gates with XOR, we get output 1 if both the inputs are opposite. That is, if both the given numbers are complements to each other (like 0101 and 1010) we get output as 1, else output will be 0.

■ **Example** The following circuit is proposed to check the equality of two numbers. Assume **carry in** is 1 to the circuit and the first numbers complement is fed into the full adders. Explain its functionality. Also, support the statement that carry out of last adder can be used to test A<B.

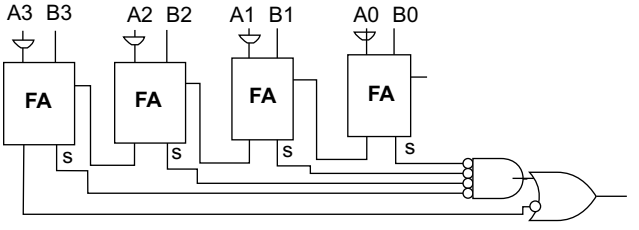
■ **Answer:** To check the equality, we can calculate B – A and if it is 0, we can say that A is same as B, else not. We know in Boolean algebra, B – A = B +(–A). Also, –A can be calculated by complementing A's bits and adding 1 to it. Thus, the circuit takes carry in as 1 while A's bits are complemented using NOT gates as shown in the circuit. If both the numbers are same, sum lines of each of the FA will be 0. Thus, sum lines are complemented before sending to AND gate. If both the numbers are exactly same, output will be 1 else it gives 1.



Carry out of last FA will be 1 if A is less than B, otherwise it will be 0.

■ **Example** The following circuit is proposed to check whether a four bit number A is greater than or equal to another four bit number B. Explore whether it works or not.

■ **Answer:** It will not work. We have replace NOR gate with OR gate to get the required output.



■ **Example** An experimental, single passenger automobile system is to be designed that will sound an alarm under certain conditions. The alarm is to sound if the seat belt is not fastened and the engine is running, or if the lights are left on when the key is not in the ignition, or if the engine is running, and the door is open.

Determine the number of inputs and outputs, and assign meaningful names to them. For example, one can chose an input variable called **seatbelt** to indicate whether the seat belt is fastened or not. We can then make the arbitrary decision to let a (logic) 1 mean that the seatbelt is fastened; a (logic) 0 would obviously mean that the seatbelt is not fastened. List all of the input/output combinations in a single truth table.

■ **Answer:** We propose to take the following Boolean variables and their possible values.

seatbelt: To indicate whether seatbelt is fastened (1) or not (0).

Ignition: To indicate whether engine is ON(1) or Off (0).

Lights: To indicate whether lights are ON(1) or Off (0).

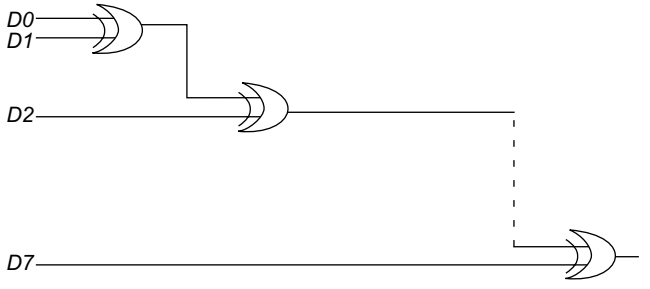
Door: To indicate whether door is closed (1) or not(0).

Alarm: To give sound (1) or not (0).

Except Alarm, all other variables are input variables. Thus, we will be having 16 possibilities.

Seatbelt	1	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1
Ignition	1	0	1	1	1	1	0	0	1	1	0	0	1	0	0	0
Lights	0	1	0	0	1	1	1	1	1	0	0	0	1	0	1	0
Door	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	0
Alarm	1	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0

■ **Example** Parity bit is used to detect single bit error. A typical example is character transmission, in which a parity bit is attached to a 7 bit character, the value of this bit is selected so that the character has an even number of 1's (even parity) or an odd number of 1's (odd parity). That is, in even parity approach parity bit is taken as 1 if the data contains odd number of 1s, otherwise parity bit value is taken as 0. Where as in the case of odd parity approach, parity bit value is taken as 1 if the data contains even number of 1s, otherwise parity bit value is taken as 0. The following circuit is proposed to such that if there is any error in transmission, i.e., any data bits gets spoiled the circuit will generate 1, otherwise it generates 0. We assume data bits are D1 to D6 and parity bit is D7. Validate its suitability for this application.

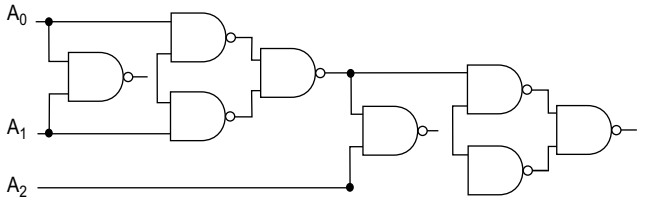


■ **Answer:**

Data	Data + Parity bit	Circuit output assuming column 2 data is fed into the circuit.
1001111	1001111 **1** (Even Parity)	0
1001011	1001011 **0** (Even Parity)	0
1001111	1001111 **0** (Odd Parity)	1
1001011	1001011 **1** (Odd Parity)	1
	Assume a single data bit is spoiled as shown below.	Circuit output assuming column 2 data is fed into the circuit.
	10 **1** 11111-(Even Parity)	1
	10 **1** 10110-(Even Parity)	1
	10 **1** 11110-(Odd Parity)	0
	10 **1** 10111-(Odd Parity)	0

From the above table, we can conclude that the circuit cannot be useful for error detection for both even and odd parity approaches. If the approach is even parity and then single data bit error gives raise 1 as output. If the approach is odd, single data bit error raises 0.

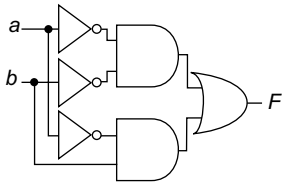
■ **Example** Prepare truth table for the following circuit and give a statement about its functionality.



■ **Answer:** By observing the following truth table, we can say that it gives one if the number ones in the given input are 1 or 3. That is, it works like odd function.

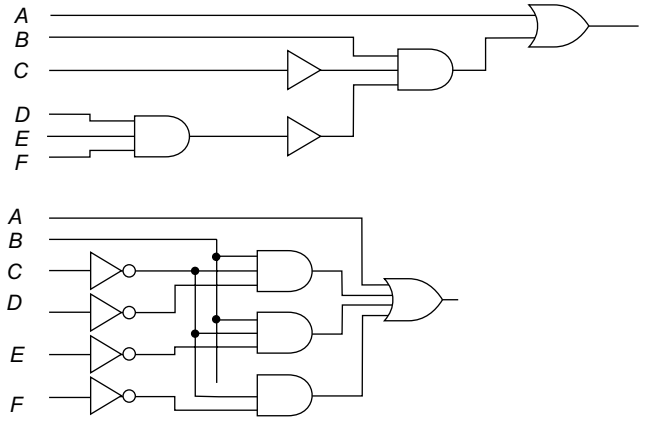
A0A1A2	1st gate	2nd level top gate	2nd level bottom gate	3rd level gate	4th level gate	5th level top gate	5th level bottom gate	Output (final gate)
000	1	1	1	0	1	1	1	0
001	1	1	1	0	1	1	0	1
010	1	1	0	1	1	0	1	1
011	1	1	0	1	0	1	1	0
100	1	0	1	1	1	0	1	1
101	1	0	1	1	0	1	1	0
110	0	0	0	1	1	0	1	0
111	0	0	0	1	0	1	1	1

■ **Example** Prove that the following circuit is same as NOT of a (negation of Boolean variable a), that is F(a,b)=a'.



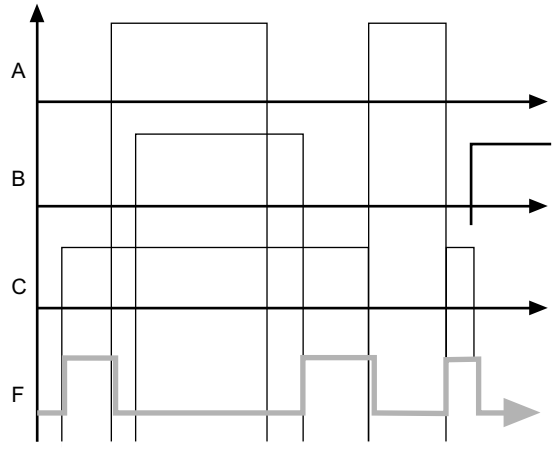
■ **Answer:** If we write F in Boolean equation, F = a'b' + a'b. If we take a' common, F becomes a'(b + b'). We know (b + b') is one. Therefore, F(a,b)= a'.

■ **Example** Compare whether the following two circuits are equivalent or not.



■ **Example** Draw timing diagram for function F given F=(A+B)'C. Neglect gate delays. Timing information of A, B, and C are given in the figure. Draw F in the same diagram.

■ **Answer:** The following figure contains timing information about F.



■ **Example** Prove A[AB + C(D' + A) + B] is equal to AB+AC.

■ **Answer:**

= A[AB + CD' + AC + B]

= AB + ACD' + AC + AB

= AB + AC(D'+1)+AB

As, AB is repeated and D'+1 is 1, the final equation becomes:

AB + AC

Therefore, the relation is proved.

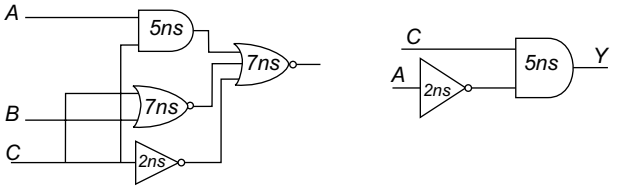
1.5 Simplification of Digital Circuits

Boolean algebra finds its most practical use in the simplification of logic circuits. If we translate a logic circuit's function into symbolic (Boolean) form, and apply certain algebraic rules to the resulting equation to reduce the number of terms and/or arithmetic operations, the simplified equation may be translated back into circuit form for a logic circuit performing the same function with fewer components. If equivalent function may be achieved with fewer components, the result will be increased reliability and decreased cost of manufacture. Also, response time of the circuits becomes attractive in addition to other benefits such as ease of packaging, lower power consumption etc. Circuit minimisation is often called as circuit optimisation also. However, we believe circuit optimisation is a little broader term. However, minimising the number gates is eternal interests for circuit designers and considered as integral part of circuit minimisation.

Circuit minimisation in a nutshell may lead to

- Reduction in number of literals (gate inputs)
- Reduction in number of gates
- Reduction number of levels of gates

Fewer inputs imply faster gates in some technologies. Also, fan-ins (number of gate inputs) are limited in some technologies. Fewer levels of gates imply reduced signal propagation delays. Consider the following two alternate circuits for a function Y.



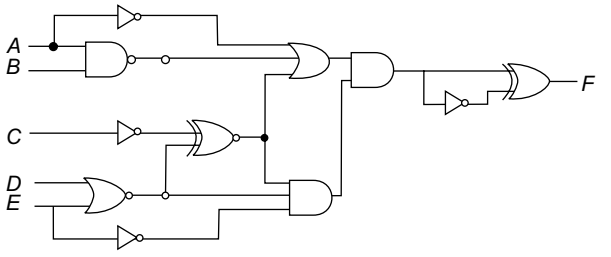
Truth table for both the circuits is given below. We may find that both are equivalents.

A	B	C	C'	AC	(B + C)'	Y (first circuit)	Y (second circuit)
0	0	0	1	0	1	0	0
0	0	1	0	0	0	1	1
0	1	0	1	0	0	0	0
0	1	1	0	0	0	1	1
1	0	0	1	0	1	0	0
1	0	1	0	0	0	1	1
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0

In both the circuit figures, we have given possible delays with each of the gates. Thus, worst possible delay in the first one (via B-Y) is 14ns whereas in the second circuit it is 7ns.

Thus, second circuit is preferred over first one. Also, second one contains less number of gates compared to the first one. As mentioned earlier, one of the objective of the circuit minimisation includes reducing the delays in the circuit such that the response times of circuits becomes better.

■ **Example** Look at the following circuit. Assume each gate delay including NOT gate also t units. Find out the possible delay of the circuit.



■ **Answer:** 6t

1.5.1 Minterms and maxterms

In Boolean algebra, any Boolean function can be expressed in a canonical form using the dual concepts of *minterms and maxterms. Minterms* are called products because they are the logical AND of a set of variables, and *maxterms* are called sums because they are the logical OR of a set of variables. These concepts are called *duals* because of their complementary-symmetry relationship as expressed by De Morgan›s laws. The dual canonical forms of *any* Boolean function are a "sum of minterms" and a "product of maxterms." The term "Sum of Products" or "SoP" is widely used for the canonical form that is a disjunction (OR) of minterms. Its De Morgan dual is a "Product of Sums" or "PoS" for the canonical form that is a conjunction (AND) of maxterms. These forms allow us for greater analysis into the simplification of these functions, which is of great importance in the minimisation or other optimisation of digital circuits.

For a Boolean function of n variables, a product term in which each of the n variables appears once (in either its complemented or un-complemented form) is called a minterm. Thus, a minterm is a logical expression of n variables that employs only the complement operator and the conjunction operator. There are 2^n minterms of n variables, since a variable in the minterm expression can be in either its direct (prime) or its complemented form(non-prime)-two choices per n variables.

Indexing minterms

In general, one assigns each minterm an index based on a conventional binary encoding of the complementation pattern of the variables (where the variables in all the minterms are written in the same order, usually alphabetical). This convention assigns the value 1 to the direct form (x_i) and 0 to the complemented form (x'_i). For example, we assign the index 6 to the minterm *ABC*' (110) and denote that minterm as m_6. Similarly, m_0 of the same three variables is *A*'B'*C*' (000), and m_7 is *ABC* (111). Given minterm number in decimal, we can find the minterm. For example, in a 4 variable system minterm m14 can be said as ABCD' (14s binary code is 1110, thus A, B, C are taken in direct form while D is taken as in complement form).

Maxterms

Similar to minterms, in a Boolean function of n variables, a sum term in which each of the n variables appears once (in either its complemented or un-complemented form) is called a maxterm. Thus, a maxterm is a logical expression of n variables that employs only the complement operator and the disjunction operator. Maxterms are a dual of the minterm idea (i.e., exhibiting a complementary symmetry in all respects). Instead of using ANDs and complements, we use ORs and complements and proceed similarly.

There are again 2^n maxterms of n variables, since a variable in the maxterm expression can also be in either its direct or its complemented form-two choices per n variables.

Indexing maxterms

Each maxterm is assigned an index based on the opposite conventional binary encoding used for minterms. The maxterm convention assigns the value 0 to the direct form (A) and 1 to the complemented form (A'). For example, we assign the index 6 to the maxterm A' + B' + C (110) and denote that maxterm as M6. Similarly M0 of these three variables is A + B + C (000) and M7 is A' + B' + C' (111).

Tables 1.2 (a) and (b) contain minterms and maxterms of a three variable system with Boolean variables being A, B and C.

Tables 1.2 (a)

A	B	C	Minterm	Maxterm
0	0	0	A′B′C′	A+B+C
0	0	1	A′B′C	A+B+C'
0	1	0	A′B C''	A+B'+C
0	1	1	A′BC	A+B'+C'

Tables 1.2 (b)

1	0	0	AB′C′	A'+B+C
1	0	1	AB′C	A'+B+C'
1	1	0	ABC′	A'+B'+C
1	1	1	ABC	A'+B'+C'

■ **Example** Consider the following two representations of a Boolean equation. Discuss which is better in terms of propagation delay and number of levels.

$$Z = (A+B')C+A'BC' \quad (1)$$

$$Z = AC+B'C+A'BC' \quad (2)$$

■ **Answer:** If we observe the first version, we may find that the circuit will be in four levels with:

1. NOT gates of A,B, and C (First Level)
2. A+B' in second level.
3. Product terms in the third level.
4. Final level contains sum term (OR gate).

Whereas the second version may need only three levels given as:

1. NOT gates of A,B, and C (First Level)
2. Product terms in the third level.
3. Final level contains sum term (OR gate).

Thus, the second version may give less propagation delay.

1.5.2 Karnaugh maps

In the previous sections, we have seen that applying Boolean algebra postulates can be awkward in order to simplify Boolean expressions. Apart from being laborious (and requiring the remembering of all the laws) the method can lead to solutions which, though they appear minimal, are not. Also, there is no guaranty that everyone gets the same final simplified form as there is no any strict procedure that has to be employed by all. Karnaugh Map builds this gap by giving a uniform procedure for simplifying the Boolean equations or logic circuits. In essence, in Karnaugh Map based Boolean equation simplification we use a systematic graphic oriented solution.

The Karnaugh map provides a simple and straight-forward method of minimising Boolean expressions. With the Karnaugh map, Boolean expressions having up to four and even six variables can be simplified. Instead of using Boolean algebra simplification techniques, we can transfer logic values from a Boolean statement or a truth table into a Karnaugh map. The arrangement of 0's and 1's within the map helps us to visualise the logic relationships between the variables and leads directly to a simplified Boolean statement. Karnaugh maps, or K-maps, are often used to simplify logic problems with 2, 3, 4 or 5 variables.

A Karnaugh map provides a pictorial method of grouping together expressions with common factors and therefore eliminating unwanted variables. The Karnaugh map can also be described as a special arrangement of a truth table. Fig. 1.8 illustrates the correspondence between the Karnaugh map and the truth table for the general case of a two variable problem. In the following figure, left side is truth table while the right side part is its Karnaugh map. Here, the output variable F is function of input variable A and B. In practice, we will be drawing Karnaugh maps for each of the output variables. In our example, only F is the output variable, thus Karnaugh map for F is shown in Fig. 1.8.

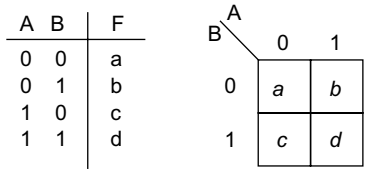


Figure 1.8 Karnaugh Map for F

The values inside the squares are copied from the output column (F) of the truth table, therefore there is one square in the map for every row in the truth table. Around the edge of the Karnaugh map are the values of the two input variable. A is along the top and B is down the left hand side. We can reverse this order also. That is, we can make A as along column while B along row.

Two variable Karnaugh maps are trivial but can be used to introduce the concept behind Karnaugh map based Boolean circuit simplification. For example, the Karnaugh map for a 2-input OR gate looks as shown in Fig. 1.9. We may find truth table and respective Karnaugh Map for OR gate.

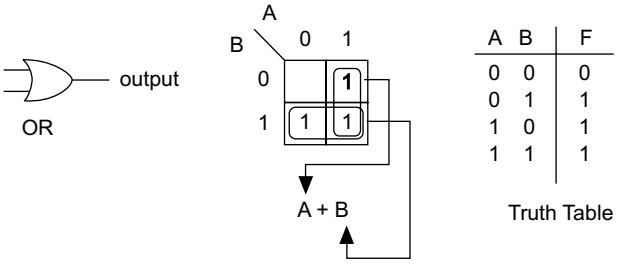


Figure 1.9 Karnaugh Map for a 2-input OR gate

As explained earlier, the values of one Boolean variable (A) appear across the top of the map, defining the column values of K-Map, while the values of the other Boolean variable (B) appear at the side, defining the values of the variable in each row of Karnaugh map. The Karnaugh map for the OR gate is completed by entering a '1' in each of the appropriate cells. Usually, we don't write in the '0's'.

Most important step in K-Map approach is identifying groups of 1s in the K-Map. That is, within the map, adjacent cells containing 1's are grouped together in twos, fours, or eights. In this case, there is one horizontal and on vertical group of two. We indicate these groupings by drawing a circle round each group of 1s. We may allow overlapped groups also. To extent possible, we will try to see every cell having 1 will be in at least in one group. In the final step, for

each group, we derive (straightly write from the graphical arrangements of 1s in that group) a Boolean equation in its minimal form.

For example, in the above K-Map, the horizontal group can be made corresponds to a B value of 1. That is, we take B into our final simplified Boolean equation. How do we achieve this and what is the background behind this is the lacuna here.

In the left hand cell of this horizontal group of 1s, A=0 and in the right hand cell, A=1. In other words, the value of A does not affect the outcome of the Boolean expression for these cells. Graphically, we can even say that half (one) of this group of cells are in A and remaining are in A'. Thus, A will not appear in final Boolean equation. Whereas all the elements of this group are in B (see K-Map). Thus, B appears in the final simplified Boolean equation. As this is a two variable system, minterm is supposed to have at most two variables or literals. However, the simplified Boolean term related to this group can be taken as B from the above discussion. This can be explained or proved algebraically as follows. Consider Boolean expression for these two cells of this group: A'B+AB

By taking B as a common term from each term, this reduces to:

$$= B\,(A'+A)$$

We know A'+A is 1. Thus, final equation becomes: B
In a similar way, the vertical group could have been written as: AB'+AB.
From the map, we can see that the value of B does not affect the value written in the cells for this group. In other words, the vertical group reduces to:
A

In this way, the Karnaugh map above leads to the overall expression for F as A + B, which is Boolean OR of A and B.

Of course, this is not very exciting but if we apply the same method to a more complex logic problem, we will begin to understand and appreciate how Karnaugh maps lead to simpler Boolean equations. We shall further explore about this in the next sections.

1.5.2.1 Exploiting Graphically Logical Adjacency: The basis for Karnaugh Map

In the previous section, we have used Logical adjacency as the basis for all Boolean simplification. In a nutshell, the facility of the K-Map approach is that it transforms logical adjacency into physical adjacency so that simplifications can be done by inspection of K-Map graph which we can call as graphical simplification.

To further understand the idea of logical adjacency, we would like to recollect two simplifications based on the fundamental properties of Boolean algebra. For any Boolean variables X and Y:

$$X \cdot Y + X \cdot Y' = X \cdot (Y + Y') = X \cdot 1 = X$$

$$(X + Y) \cdot (X + Y') = X \cdot X + X \cdot Y' + Y \cdot X + Y \cdot Y'$$
$$= X \cdot X + X \cdot Y' + X \cdot Y + 0$$
$$= X + X \cdot (Y' + Y) = X + X = X$$

Two Boolean terms are said to be logically adjacent when they contain the same Boolean variables and differ in the form of exactly one variable i.e., one variable will appear negated in one term and in true form in the other term and all other variables have the same appearance in both terms. Consider the following lists of terms, the first in 1 variable and the others are having two variables.

X	X'		
X • Y	X • Y'	X' • Y'	X' • Y
(X + Y)	(X + Y')	(X' + Y')	(X' + Y)

The terms in the first list are easily seen to be logically adjacent (also physically when represented in K-Map). The first term has a single variable in the true form and the next has the same variable in the negated form.

We now examine the second list, which is a list of product terms each with two variables. Note that each of the terms differs from the term following it in exactly one variable and thus is logically adjacent to it: X • Y is logically adjacent to X • Y', X • Y' is logically adjacent to X' • Y', X' • Y' is logically adjacent to X' • Y, and X' • Y is logically adjacent to X • Y. Note that logical adjacency is a commutative relation thus X • Y' is logically adjacent to both X • Y and X' • Y'. Using the SOP notation, we represent this list as 11, 10, 00, 01.

The third list also displays logical adjacencies in its sequence: (X + Y) is logically adjacent to (X + Y'), which is logically adjacent to (X' + Y'), which is logically adjacent to (X' + Y). Using POS (product of sums) notation, we represent this list as 00, 01, 11, 10.

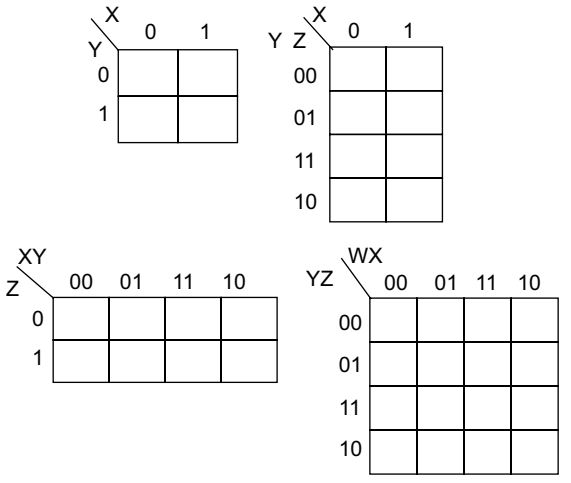
Consider the list of product terms when written in the more usual sequence
X' • Y' X' • Y X • Y' X • Y, or 00, 01, 10, 11 in the SOP (sum of products) notation.

In viewing this list, we can see that the first term is logically adjacent to the second term, but that the second term is not logically adjacent to the third term: X' • Y and X • Y' differ in two variables. This type of conclusions can be also arrived at by viewing the numeric list 00, 01, 10, and 11 which are binary equivalents of Boolean terms. Note that each of the digits in 01 and 10 is different, so that 01 and 10 can't represent logically adjacent terms.

■ **Example** Is it possible to say that two terms are logically adjacent if their Hamming distance is 1?

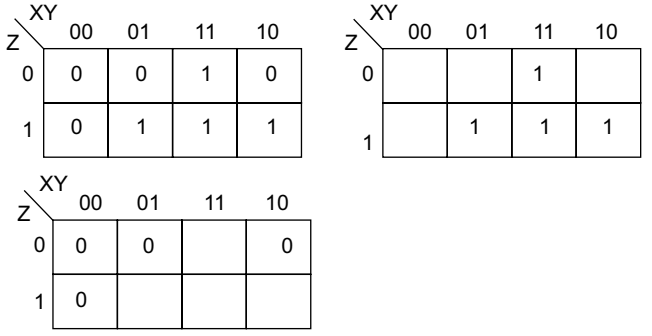
1.5.2.2 Karnaugh Maps for 2, 3, and 4 variables

K-Maps with 5 or more variables are hopelessly complex to display graphically. The following figure shows the basic K-Maps for 2, 3, and 4 variables. Note that there are two equivalent forms of the 3-variable K-Map; the student should pick one style and use it. We have already explained how K-Map is formed from the truth table of a Boolean equation.



One way to view a K-Map is as a truth-table with the main exception of the ordering 00, 01, 11, 10. For those interested, this ordering is called a Gray code. This ordering is used for ease of graphical simplification.

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



In our previous explanation, we have mentioned that the cells with 1s are usually displayed in K-Map. However, in practice we can represent the same in other forms also. For example, the above three K-Map forms are equivalent. In the first one, we have displayed both 1s and 0s of the truth table, second one displays only 1s of the truth table, while the third one displays 0s of the truth table.

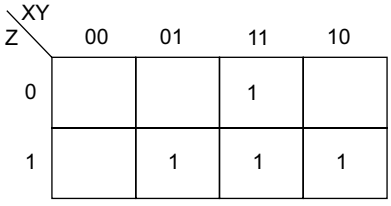
The K-Map form omitting the 0's is used when simplifying SOP expressions, while simplifying POS expression, we use K-Map for that omits the 1's.

Note

K-Maps are used to simplify Boolean expressions written in canonical form.

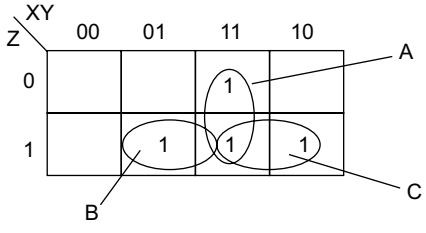
1.5.3 K-Maps for Sum of Products (SOP)

Consider the Canonical SOP expression F(X,Y,Z) = X' • Y • Z + X • Y' • Z + X • Y • Z' + X • Y • Z.
The first step in using K-Maps to simplify this expression is to use the SOP numbering to represent these as 0's and 1's. The negated variable is written as a 0, the plain as a 1. Thus, this function is represented as 011, 101, 110, and 111.



Place a 1 in each of the squares with the "coordinates" given in the list above. In the K-Map at left, the entry in the top row corresponds to 110 and the entries in the bottom row correspond to 011, 111, and 101 respectively. Remember that we do not write the 0's when we are simplifying expressions in SOP form.

The next step is to notice the physical adjacencies. We group adjacent 1's into "rectangular" groupings of 2, 4, or 8 boxes. Here there are no groupings of 4 boxes in the form or a rectangle, so we group by two's. There are three such groupings, labeled A, B, and C (see figure below).



The grouping labeled A represents the product term XY as all the cells of this group are in X and Y. We may find Z term is missing as half of the cells are in Z and remaining half are in Z'. Similarly, the B group can be said as representing the product term YZ while the C group cells representing the

product term XZ. Thus, final simplified Boolean function can be said as: X • Y + X • Z + Y • Z.

Summary of points while simplifying Boolean equations using K Map

1. Prepare K-Map table from the Boolean equation or truth table.
2. Only groups of 1's are selected during simplification process.
3. Group (block) of 1s (diagonals are not selected).
4. Only groups of cells with number of cells that are integer power of 2 are selected. That is, a group of 1, 2, 4, 8, 16, etc., are selected.
5. Groups (blocks) should be as large as possible.
6. Every 1 must be in at least one group (block).
7. Overlapping of groups is allowed.
8. Wrap around allowed while groups 1s.
9. Fewest number of groups (blocks) that embodies all 1's in the K-Map is selected. This selected group's decides the minimal Boolean equation.
10. If we have m variables then we will be having 2^m cells in K-Map. Thus, largest possible group contains 2^m cells only.
11. In a K-Map with 2^m cells, if selected group contains 2^d cells then the corresponding simplified Boolean product term related to this group contains m-d variables. While finding which variables will be there in the minimised Boolean term we follow the following rules. If all 1s of a group are in column (or row) of input variable A (or A'), we take A into minimised Boolean expression. This, we apply for each of the input variables and thus final simplified Boolean equation for a group if framed graphically. This is the basis for K-Map method. For example, consider Fig. 1.10 in which groups of 1s are marked with circles. First, consider bottom left most circle whose 1s are completely in a and c. Thus, we take Boolean equation ac. Similarly, the bottom right most 1s are completely in a and b. Thus, we consider ab against this group. In the same fashion, we can take term bc against another group of 1s of the figure. Thus, final simplified equation for this K-Map table is: ac + ab +bc.

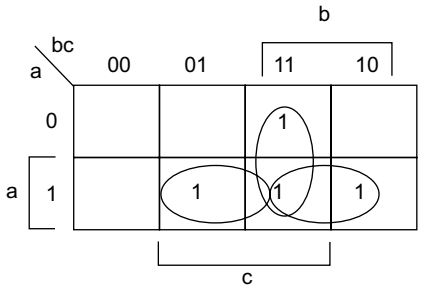
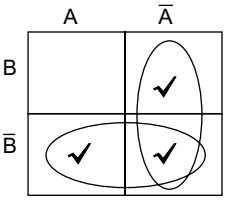


Figure 1.10 Grouping is in 3-input Karnaugh Map

■ **Example** Simplify $A\overline{B}+\overline{A}B+\overline{A}\,\overline{B}$



■ **Answer:**

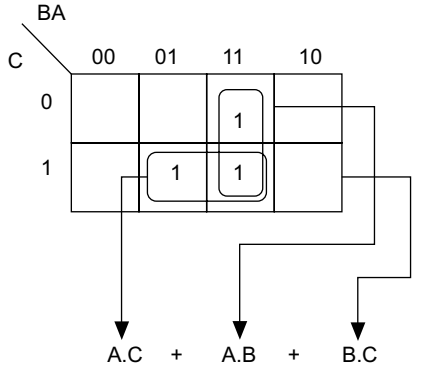
- Place a check in the $\overline{A}B$ area.
- Place a check in the $\overline{A}B$ area.
- Place a check in the $\overline{A}\,\overline{B}$ area.
- Draw loops around pairs of adjacent checks.

Because there are two blocks of 1s, there will be two terms in the simplified expression. The vertical loop contains $\overline{A}$, B, $\overline{A}$, and $\overline{B}$. We remove one $\overline{A}$ to make an unduplicated list. The B and $\overline{B}$ cancel, leaving the remaining $\overline{A}$. From the horizontal loop we remove the duplicate $\overline{B}$, then remove A and $\overline{A}$ leaving only $\overline{B}$ in the second term. We write the Boolean sum of these, and the result is $A+\overline{B}$. Thus, $A\overline{B}+\overline{A}B+\overline{A}\overline{B}=A+\overline{B}$

■ **Example** Here is the truth table for a 3-person majority voting system:

C	B	A	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

This is converted into a Karnaugh map, as follows:



Within the K-map, we can identify three groups of two, as indicated. The left hand horizontal group combines the

cells AB'C and A.B.C. Within this group, the value of B does not affect the cell values. This means that B can be eliminated from the expression, leaving AC. Also, we can infer Boolean equations for the other two groups as AB and BC. Thus, final simplified Boolean equation which satisfies the majority voting is: AC + AB + BC.

What are the steps used while designing a practical digital system?

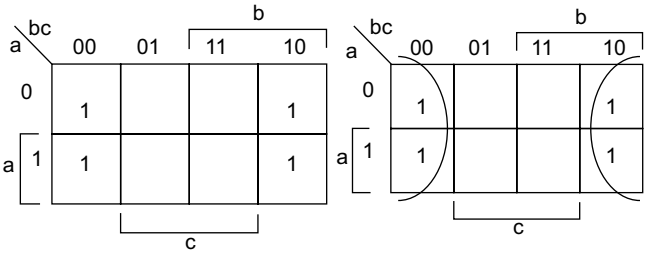
Step 1: represent input and output signals with Boolean variables.

Step 2: construct truth table to carry out computation.

Step 3: derive (simplified) Boolean expression using sum-of products.

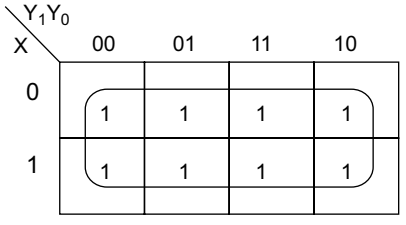
Step 4: transform Boolean expression into circuit.

■ **Example** Let's try another 3 variable map.



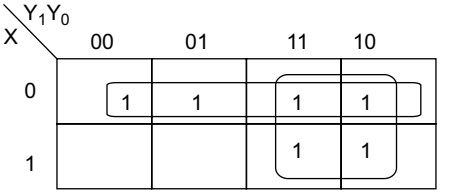
At first it may seem that we have two sets, one on the left most column of the map and the other on the right most column. If we observe, 000 and 010 are logically adjacent as they differ at one variable. Also, 100 and 110 differs by one variable. Thus, all the four 1s should be considered as one set because the left and right are adjacent as are the top and bottom. The expression for all 4 1s is c'. Notice that the 4 1s span both values of a (0 and 1) and both values of b (0 and 1). Thus, only the c value is left. The variable c is 0 for all the 1s, thus we have c'. The other way to look at it is that the 1's overlap the horizontal b line and the short vertical a line, but they all lay outside the horizontal c line, so they correspond to c'. (The horizontal c line delimits the c set. The c' set consists of all squares outside the c set. Since the circle includes all the squares in c', they are defined by c'. Again, notice that both values of a and b are spanned, thus eliminating those terms.)

■ **Example**



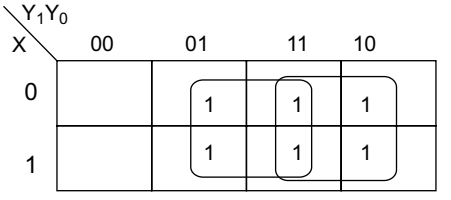
The sample below is based on an earlier design shows a particularly simple problem. We find that all the entries in the K-Map are covered with a single grouping, thus removing all three variables. Since the entire K-Map is covered, the simplification is F = 1. That is for any input variables, the output is 1.

■ **Example**

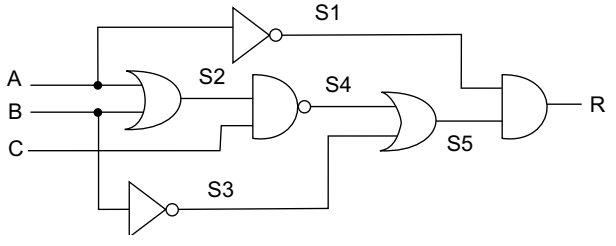


The K-Map given above shows an example with overlap of two groupings of 1's. All 1's in the map must be covered and some should be covered twice. The top row corresponds to X'. We then form the 2-by-2 grouping at the right to obtain the term Y_1. Thus F = X' + Y_1.

■ **Example** The 2-by-2 grouping in the middle gives rise to Y_0 and left to Y_1. So we get simplified equation as = Y_0 + Y_1.



■ **Example** A student has shown his assignment for a problem as the following circuit. Professor shouted at him saying you did not minimize correctly. Prepare your version of minimised circuit which satisfies the function requirement and efficient.



■ **Answer:** We first prepare truth table for the above circuit to find out function requirement.

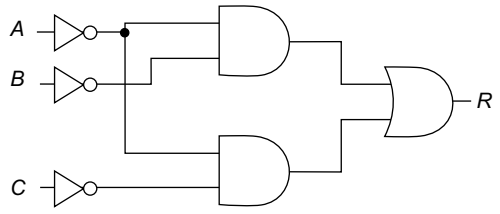
ABC	S1	S2	S3	S4	S5	R
000	1	0	1	1	1	1
001	1	0	0	1	1	1
010	1	1	1	1	1	1
011	1	1	0	0	0	0
100	0	1	1	1	1	0
101	0	1	0	0	0	0
110	0	1	1	1	1	0
111	0	1	0	0	0	0

Karnaugh map for the above system is given as:

A/BC	00	01	11	10
0	1	1		1
1				

Therefore, R= A'B' + A'C'

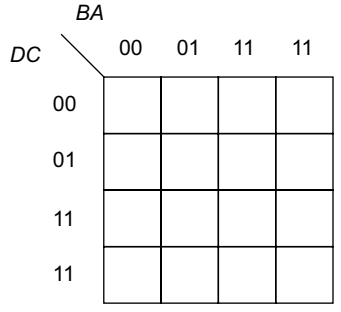
If we observe, we may find that in reality the above function can be realised with two 2-input AND gates, one OR gate and three NOT gates as shown in the following figure.



This is simpler than the circuit given by the student. Also, it is in three level while our simplified version suggests two level circuit only. Thus, Professor is correct.

Simplifying four variable Boolean equations

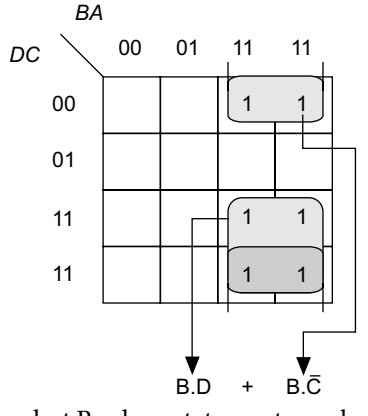
A 4-variable map will contain $2^4 = 16$ cells. It is important to write the variable values along the columns and rows in Grey code:



To simplify the equation:

$$x = B.\overline{C}.\overline{D} + \overline{A}.B.\overline{C}.D + A.B.\overline{C}.D + \overline{A}.BCD + A\,BCD$$

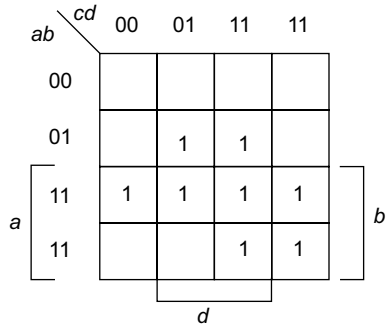
The Karnaugh map becomes:



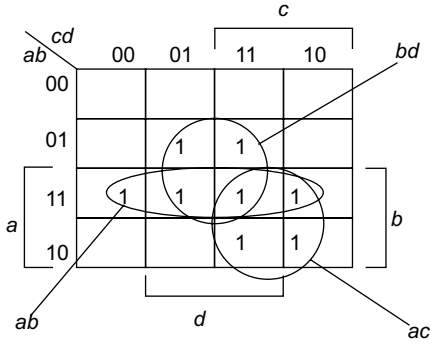
To give the simplest Boolean statement, we should put a circle round the maximum number of terms. In this case, we can make two groups of four, one of which wraps around from top to bottom. We identify the two variables which remain constant in each group and eliminate the other two:

$$x = B.D + B.\overline{C}$$

■ **Example** Simplify q = a'bc'd + a'bcd + abc'd' + abc'd + abcd' + ab'cd + ab'cd'



Grouping the 1s together results in the following.

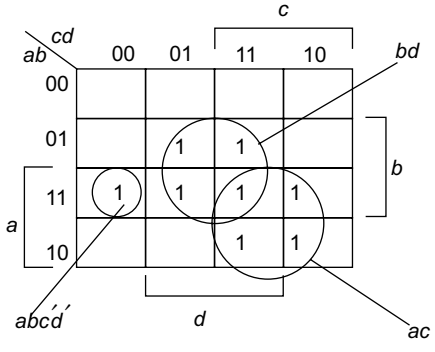


The simplified expression for the groupings above is given as:

$$q = bd + ac + ab$$

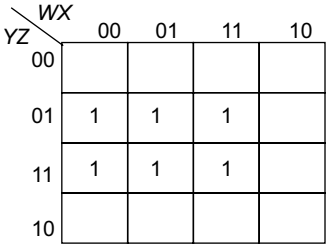
This expression requires three 2-input and gates and one 3-input or gate.

We could have accounted for all the 1s in the map as shown below, but that results in a more complex expression requiring a more complex gate.

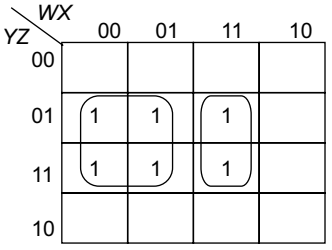


The expression for the above is bd + ac + abc'd'. This requires two 2-input and gates, a 4-input and gate, and a 3 input or gate. Thus, one of the AND gate is more complex (has two additional inputs) than required above. Two inverters are also needed.

■ **Example** Another example with overlapping groups
Consider the following K-Map.

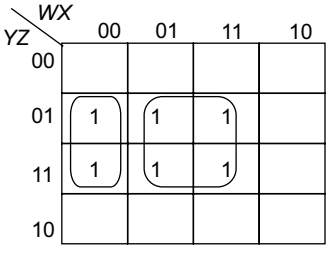


The six ones can be grouped in a number of ways. Consider the following.

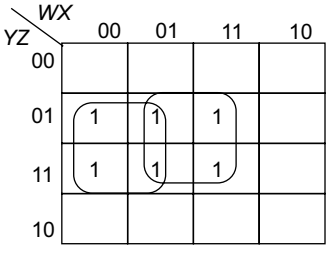


This grouping of four and two covers the six one's in the K-Map.

The four ones in the square form the term W'•Z. The two ones in the rectangle form the term W•X•Z.
The K-Map simplifies to W'•Z + W•X•Z.



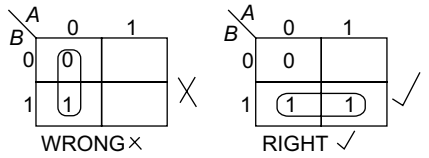
Another way to consider the simplification of the K-Map is to group the rectangle and the square as in the figure.
The rectangle corresponds to the term W'•X'•Z.
The square corresponds to the term X•Z.
This simplification yields W'•X'•Z + X•Z.
It is important to note that the groupings can overlap if this yields a simpler reduction.



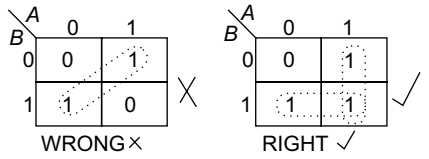
Here we show two overlapping squares.
The square at left corresponds to the term W'•Z.
The square at right corresponds to the term X•Z.
This simplification yields W'•Z + X•Z, which is simpler than either of the other two forms validly produced by the K-Map method.

The Karnaugh map uses the following rules for the simplification of expressions by *grouping* together adjacent cells containing *ones*

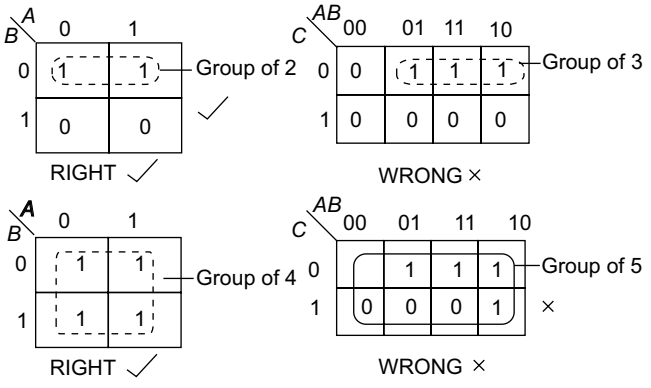
- Groups may not include any cell containing a zero



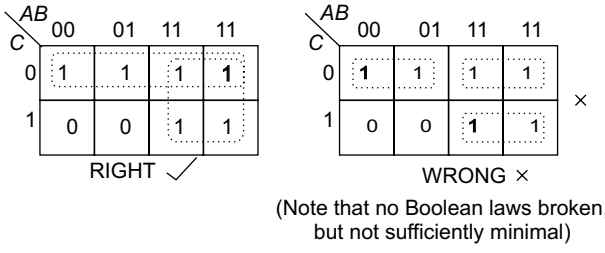
- Groups may be horizontal or vertical, but not diagonal.



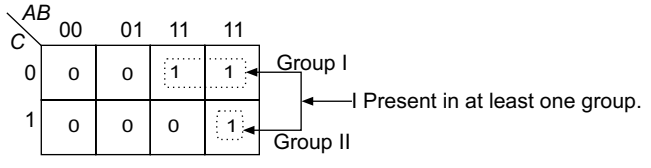
- Groups must contain 1, 2, 4, 8, or in general 2^n cells. That is if n = 1, a group will contain two 1's since 2^1 = 2. If n = 2, a group will contain four 1's since 2^2 = 4.



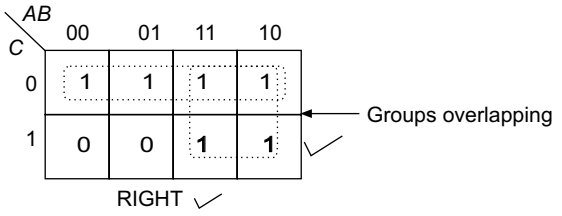
- Each group should be as large as possible.

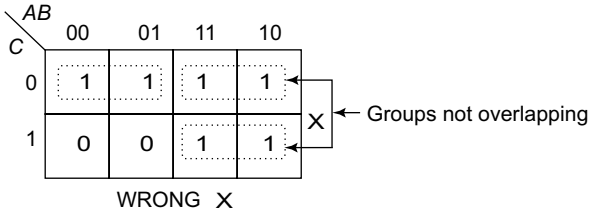


- Each cell containing a *one* must be in at least one group.

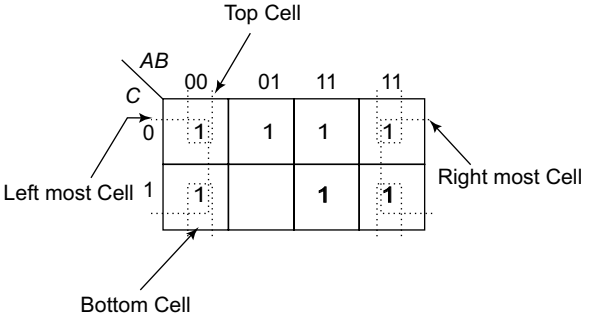


- Groups may overlap

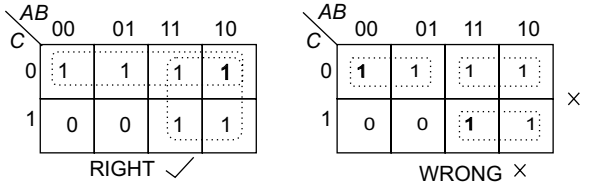




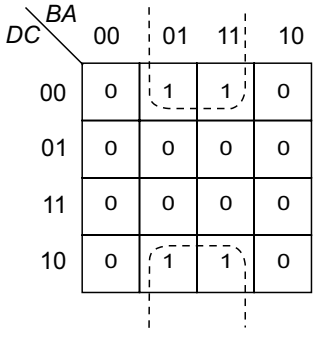
- Groups may wrap around the table. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.



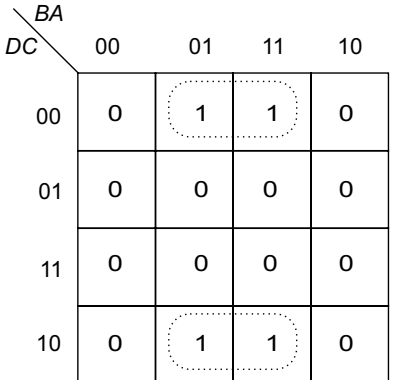
- There should be as few groups as possible, as long as this does not contradict any of the previous rules.

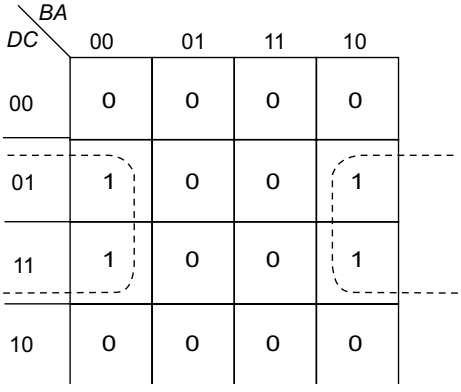


In a four variable Karnaugh map also, we can group 1s in wraparound fashion.

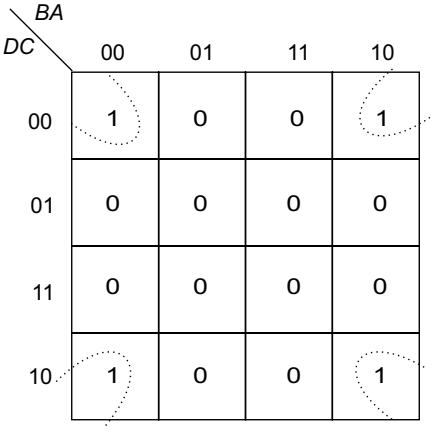


Correct





Correct wraparound grouping.



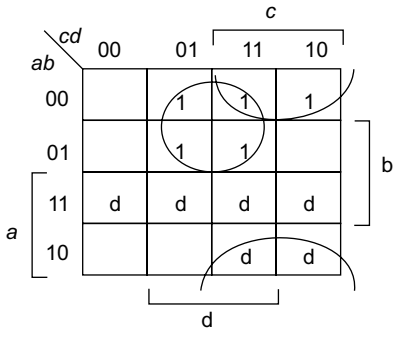
Incorrect

1.5.3.1 Don't Cares

Sometimes we do not care whether a 1 or 0 occurs for a certain set of inputs. It may be that those inputs will never occur so it makes no difference what the output is. For example, we might have a bcd (binary coded decimal) code which consists of 4 bits to encode the digits 0 (0000) through 9 (1001). The remaining codes (1010 through 1111) are not used. If we had a truth table for the prime numbers 0 through 9, it would be

abcd	p
0000	0
0001	1
0010	1
0011	1
0100	0
0101	1
0110	0
0111	1
1000	0
1001	0
1010	d
1011	d
1100	d
1101	d
1110	d
1111	d

The d's in the above stand for "don't care", we don't care whether a 1 or 0 is the value for that combination of inputs because (in this case) the inputs will never occur in practice.



The circle made entirely of 1s corresponds to the expression a'd and the combined 1 and d circle (actually a combination of arcs) is b'c. Thus, if the disallowed input 1011 did occur, the output would be 1 but if the disallowed input 1100 occurs, its output would be 0. The minimised expression is

$$p = a'd + b'c$$

Notice that if we had ignored the ds and only made a circle around the 2 1s, the resulting expression would have been more complex, a'b'c instead of b'c.

Table 1.2

	X = 0	X = 1
Y_1Y_0	J_1	J_1
0 0	0	1
0 1	1	0
1 0	d	d
1 1	d	d

The general rule in considering a simplification with the Don't-Care conditions is to count the number of 0's and number of 1's in Table 1.2 and to use SOP simplification when the number of 1's is greater and POS simplification when the number of 0's is greater. Again we admit that most students prefer the SOP simplification. With a two-two split, we try SOP simplification.

Table 1.3

Y_1	Y_0	X	J_1
0	0	0	0
0	0	1	1
0	1	0	d
0	1	1	d
1	0	0	1
1	0	1	0
1	1	0	d
1	1	1	d

First we should explain Table 1.3 in detail. The first thing to say about it is that we shall see similar tables again when we study flip-flops. For the moment, we call it a "folded over" truth table, equivalent to the full truth table. The function to be represented is J_1. Lines 0, 1, 4, and 5 of the truth table seem to be standard, but what of the other rows in which J_1 has a value of "d". This indicates that in these rows it is equally acceptable to have $J_1 = 0$ or $J_1 = 1$. We have four "Don't-Cares" or "d" in this table; each can be a 0 or 1 independently of the others – in other words we are not setting the value of *d* as a variable.

Design with flip-flops is the subject of another course.

Observability Don't Care

For some input patterns, the outputs of the system won't change if one output bit of some subsystem changes. So that bit of that subsystem is a don't care for such input patterns.

Satisfiability Don't Care

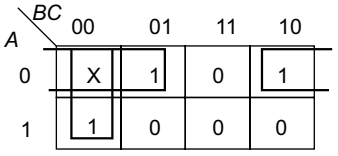
Some input patterns never appear. The output of the system are don't care for such input patterns.

■ **Example** Suppose We have a system with three buttons. Each button sends a logic 1 when the button is being pressed and a zero once it is released. The system should light an LED (by sending it a logic 1) whenever only one is button pressed at a time, and should turn off the LED (by sending it a logic 0) when more than one button is pressed. At least one button will always be ressed so we do not care what the circuit does when no buttons are pressed. Design a minimized circuit to control the LED.

■ **Answer:** Call the buttons A, B, and C. The desired output is shown in the truth table below, where L is the LED signal. Do observe the first row, if all the buttons are not pressed the state of the LED can be anything.

A	B	C	L
0	0	0	X
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Karnaugh map and maxterm groupings are given below.



From the Karnaugh map, we can find L=A′B′+A′C′+B′C′

Simplification of Boolean equations given minterms or maxterms.

Sometimes, we will be given Boolean equation in terms of POS or SOP. We need to simplify the equation. At this junction, we need to know minterms or maxterms indexes. For example, Fig. 1.11 displays the minterm numbers for 2, 3 and 4 variable systems.

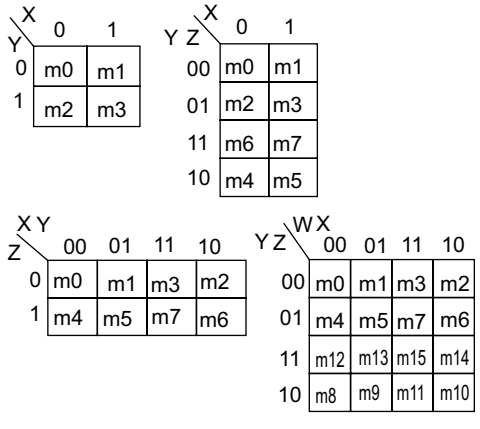
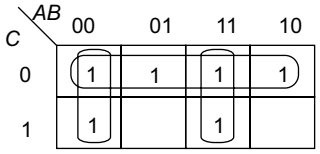


Figure 1.11 K-Map with minterm indexes

■ **Example** The next example is to simplify F(A, B, C) = Π(3, 5). We shall consider use of K-Maps to simplify POS expressions, but for now the solution is to convert the expression to the SOP form F(A, B, C) = Σ(0, 1, 2, 4, 6, 7). We could write each of the six product terms, but the easiest solution is to write the numbers as binary: 000, 001, 010, 100, 110, and 111.



The top row of the K-Map corresponds to the entries 000, 010, 100, and 110, arranged in the order 000, 010, 110, and 100 to preserver logical adjacency. The bottom row corresponds to the entries 001 and 111. The top row simplifies to C'. The first column simplifies to A'B' and the third column to AB. Thus we have F(A, B, C) = A' • B' + A • B + C'.

We next consider a somewhat offbeat example not in a canonical form.

$$F(W, X, Y, Z) = W' \bullet X' \bullet Y' \bullet Z' + W' \bullet X' \bullet Y' \bullet Z + W \bullet X' \bullet Y'.$$

The trouble with K-Maps is that the technique is designed to be used only with expressions in canonical form. In order to use the K-Map method we need to convert the term W • X' • Y' to its equivalent W • X' • Y' • Z' + W • X' • Y' • Z, thus obtaining a four-term canonical SOP.

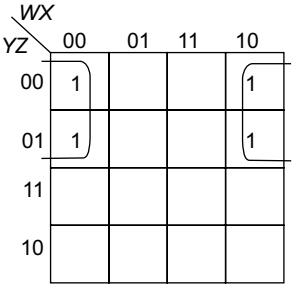
Before actually doing the K-Map, we first apply simple algebraic simplification to F.

$$\begin{aligned} F(W, X, Y, Z) &= W' \bullet X' \bullet Y' \bullet Z' + W' \bullet X' \bullet Y' \bullet Z + W \bullet X' \bullet Y' \\ &= W' \bullet X' \bullet Y' \cdot (Z' + Z) + W \bullet X' \bullet Y' \\ &= W' \bullet X' \bullet Y' + W \bullet X' \bullet Y' \\ &= (W' + W) \bullet X' \bullet Y' = X' \bullet Y' \end{aligned}$$

Now that we see where we need to go with the tool, we draw the four-variable K-Map.

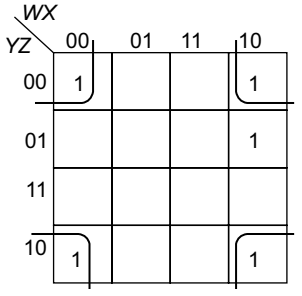
$$F(W, X, Y, Z) = W' \bullet X' \bullet Y' \bullet Z' + W' \bullet X' \bullet Y' \bullet Z + W \bullet X' \bullet Y' \bullet Z' + W \bullet X' \bullet Y' \bullet Z.$$

Using the SOP encoding method, these are terms 0000, 0001, 1000, and 1001. The K-Map is



The first row in the K-Map represents the entries 0000 and 1000. The second row in the K-Map represents the entries 0001 and 1001. The trick here is to see that the last column is adjacent to the first column. The four cells in the K-Map are thus adjacent and can be grouped into a square.

We simplify by noting the values that are constant in the square: X = 0 and Y = 0. Thus, the expression simplifies to X' • Y', as required.

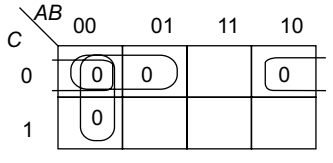


We close the discussion of SOP K-Maps with the example at right, which shows that the four corners of the square are adjacent and can be grouped into a 2 by 2 square. This K-Map represents the terms 0000, 0010, 1000, 1010 or W′ • X′ • Y′ • Z′ + W′ • X′ • Y • Z′ + W • X′ • Y′ • Z′ + W • X′ • Y • Z'. The values in the square that are constant are X = 0 and Z = 0, thus the expression simplifies to X′ • Z′.

1.5.4 K-Maps for POS

K-Maps for Product of Sums simplification are constructed similarly to those for Sum of Products simplification, except that the POS copy rule must be enforced: 1 for a negated variable and 0 for a non-negated (plain) variable. Place a 0 at each location, rather than the 1 placed for SOP.

Consider F2 = (A + B + C) • (A + B + C') • (A + B' + C) • (A' + B + C). Using the POS copy rule, we translate this to 000, 001, 010, and 100.



We begin the K-Map for POS simplification by placing a 0 in each of the four positions 000, 001, 010, 100. Noting that 000 is adjacent to 001, just below it, we combine to get 00– or (A + B). The term 000 is adjacent to 010 to its right to get 0–0 or (A + C). The term 000 is adjacent to 100 to its "left" to get –00 or (B + C). As a result, we get the simplified form. F2 = (A + B) • (A + C) • (B + C)

How to Simplify Boolean equation which does not contain canonical terms ?

All the examples which we have discussed till now contain Boolean equations with canonical terms. How do we tackle the situation in which the Boolean equation terms are not canonical? May be, we can prepare truth table from the given Boolean equation and then prepare Karnaugh map for simplification. This process is little laborious.

We know the following facts about Karnaugh maps.

For a 2 input function,

i. any term that contains 2 variables = 1 one cell in the Karnaugh map.
ii. any term that contains 1 variable = 2 cells in the Karnaugh map.

For a 3 input function,

i. any term that contains 3 variables = 1 cell in the Karnaugh map.
ii. any term that contains 2 variables = 2 cells in the Karnaugh map.
iii. any term that contains 1 variable = 4 cells in the Karnaugh map.

For a 4 input function,

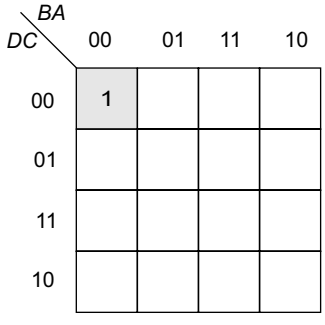
i. any term that contains 4 variables = 1 cell in the Karnaugh map.
ii. any term that contains 3 variables = 2 cells in the Karnaugh map.
iii. any term that contains 2 variables = 4 cells in the Karnaugh map.
iv. any term that contains 1 variable = 8 cells in the Karnaugh map.

If we recapitulate what we have done Karnaugh map simplification, we know that we have exploited logical adjacency from Karnaugh map. That is, a group of 1s of size 2, 4 , 8 are represented by a minterm with minimum number of variables. Thus, if we are given a Boolean equation without canonical terms, we replace the terms with a group of canonical terms and prepare Karnaugh map for simplification. That is, we may be working in exactly opposite way of grouping terms. The following examples explains our idea.

■ **Example** Simplify the following Boolean Expression which contains some non canonical terms. The system is four variable system.

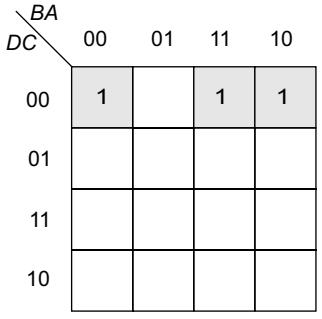
$$\overline{A}.\overline{B}.\overline{C}.\overline{D} + B.\overline{C}.\overline{D} + \overline{B}.\overline{C}.D + A.\overline{B}.C.\overline{D} + A.\overline{B}.D + B.\overline{C}.D$$

We take each term individually and put a '1' in the cell of the Karnaugh map that corresponds to the logic expression if it is in fully canonical form.

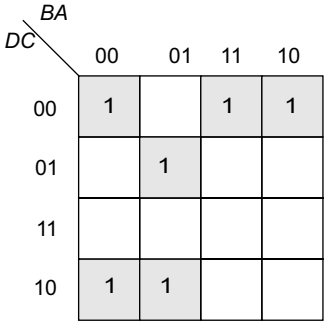


First term $\overline{A}.\overline{B}.\overline{C}.\overline{D}$ (4 variables = 1 cell in map)

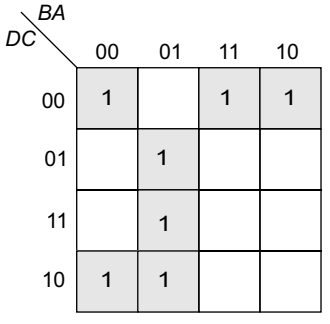
Second term $\mathbf{B.\overline{C}.\overline{D}}$ is having only 3 variabes while the system is four variable system. Missing variable is A. Thus, we can consider this term as two terms ABC'D' and A'BC'D'. That is, in the Karnaugh map this term occupies two cells as shown in this figure.



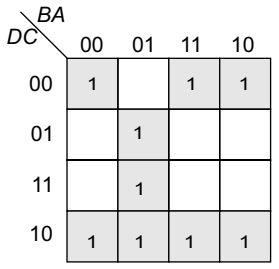
Similarly third term $\mathbf{\overline{B}.\overline{C}.D}$ can be decomposed into AB'C'D and A'B'C'D. Thus, in the Karnaugh map this term contributes 1s to two cells as shown below. Also, fourth term $\mathbf{A.\overline{B}.C.\overline{D}}$ contributes 1 to a cell in the Karnaugh map as shown in this figure.



Fifth term $\mathbf{A.\overline{B}.D}$ can be considered some of two canonical terms AB'CD and AB'C'D. Thus, it contributes 1 to two cells as shown in this figure. However, AB'C'D is already one from previous terms.

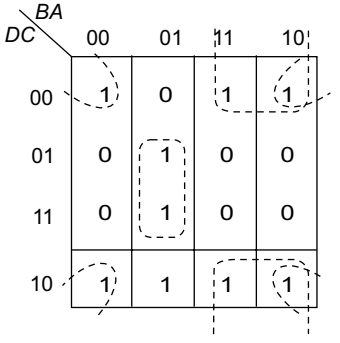


Similarly sixth term $B.\overline{C}.D$ is having only 3 variables, i.e., A component is missing. Thus, we consider it as ABC'D and A'BC'D. Thus, this term contributes to two terms in the Karnaugh map as shown in this figure.



The following Karnaugh shows groupings of 1's. Final Boolean equation become:

$$\mathbf{\overline{C}.D + \overline{B}.C + \overline{A}.\overline{C} + A.\overline{B}.C}$$



1.5.5 Karnaugh Maps with 5 and 6 Variables

In the case of 5 variable system, Karnaugh map contains $2^5 = 32$ cells. All the cells are organised two four variable maps (4x4 cells) as shown in Figure 1.12. First variable A is used to select which of the four variable map while other four variables BCDE are used to select the cell in the four variable map.

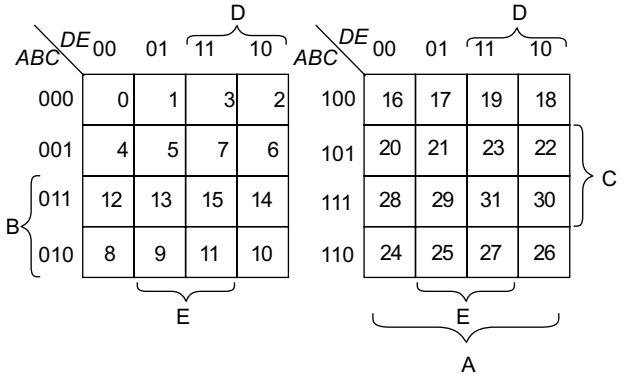
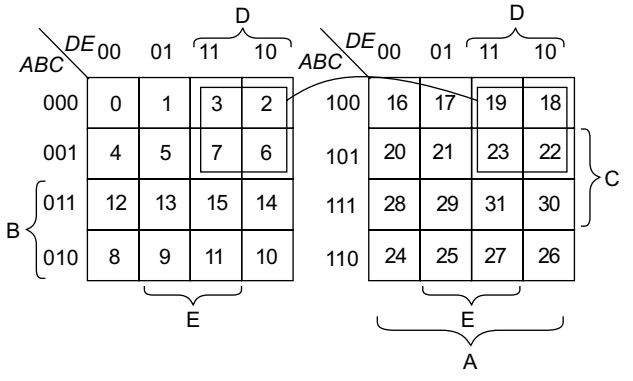


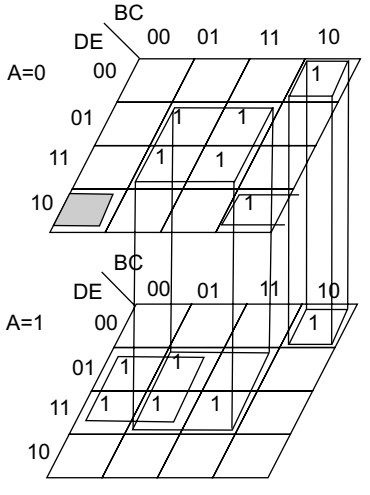
Figure 1.12 A 5-variable Karnaugh Map

The numbering of the cells (minterms) also shown in Fig. 1.12. Here, we can group cells of 1s of 2, 4, 8, 16 and 32. While grouping we can consider the two four variable karnaugh maps as stacked, i.e., one on the top of the other. For example, in the following figure we have taken a group of 8

1s as a cell in which four 1s are in first part while the other four are in the second part. If we visualise these two four cells pairs will be one above the other. If we observe the selected cells, half of them are in A and half in A'. Thus, A will not appear in the final Boolean term. If one observes, this is true with C and E also. Also, all the 1s of this group are completely in D and B'. Thus, simplified Boolean equation for this group is B'D.

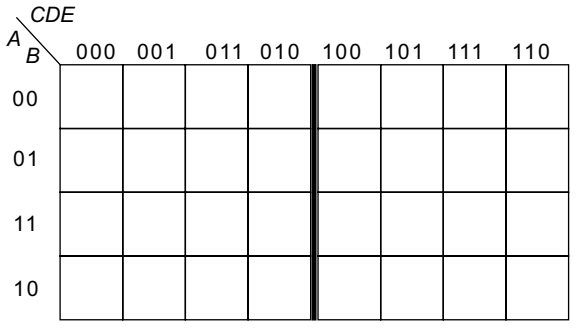


The above thing can be realised like the following also. Here, middle variable C is zero in first half and 1 in the second half. This we can verify the codes on the top of the Karnaugh map. The following figure further enlightens about grouping in five variable Karnaugh maps. In Figure 1.13, we have grouping of single 1 (A'B'C'DE'), two 1s(BC'D'E'), etc.



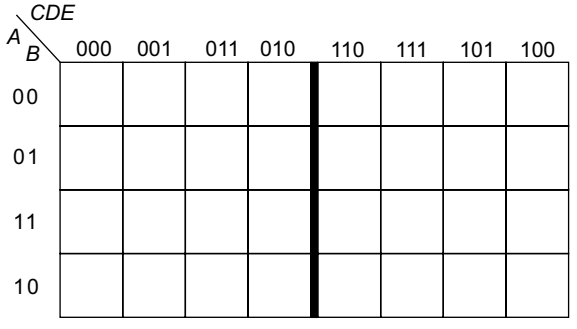
$f(A,B,C,D,E) = \Sigma M\ (2,5,7,8,10,13,15,17,19,21,23,24,29,31)$
$= CE + AB'E + BC'D'E'$
$+A'C'D'E'$

Figure 1.13 Grouping in 6-variable Karnaugh Map



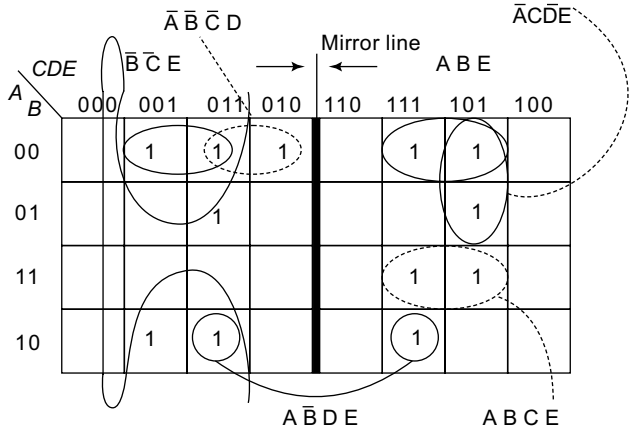
5- variable Karnaugh map (overlay)

In old text books, we have some other form of 5 variable Karnaugh map is used in which gray code is used at the top of the Karnaugh map.



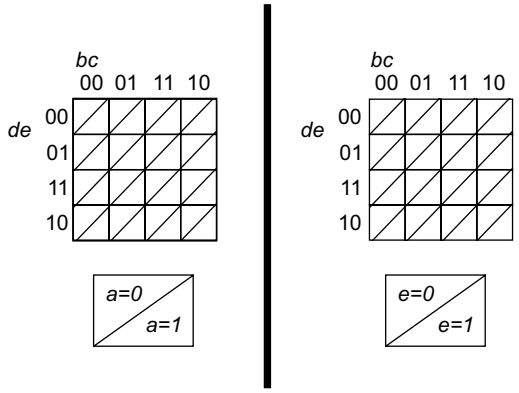
5-variable Karnaugh map (Gray Code)

When using this version of the K-map, we look for mirror groups of 1s in both the halves of the map. For example, in the following 5-variable map, we have shown groups of 1s which are mirrors in both the halves and their respective Boolean equation.



5-variable Karnaugh map (Gray Code)

We can also represent 5-variable Karnaugh maps as shown below. Here, each cell is devided into two halves by drawing a diagonal. First (or last) variable and its complement will be shown in the same shell. For example, AB'C'D'E' and A'B'D'C'E' will shown in 0^{th} row and 0^{th} column, first one is below the diagonal and second one as above the diagonal.

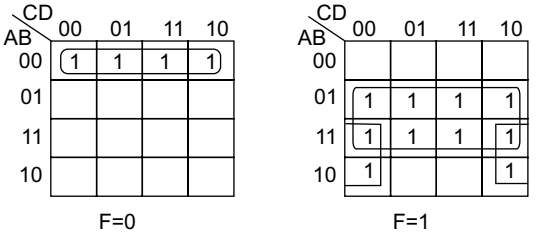


■ **Example** Simplify the Boolean function

F(A, B, C, D, E) = Σ(0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31) Writing decimals in binary,

Decimal	A	B	C	D	E
0	0	0	0	0	0
2	0	0	0	1	0
4	0	0	1	0	0
6	0	0	1	1	0
9	0	1	0	0	1
11	0	1	0	1	1
13	0	1	1	0	1
15	0	1	1	1	1
17	1	0	0	0	1
21	1	0	1	0	1
25	1	1	0	0	1
27	1	1	0	1	1
29	1	1	1	0	1
31	1	1	1	1	1

We will construct two Karnaugh maps for variables A,B,C and D when E = 0 and E = 1



From Karnaugh map E = 0, the selected group of 1s gives Boolean term A'B'E' from Karnaugh map E=1, the selected groups of 1s gives Boolean terms BE and AD'E. Thus,

$$F = A'B'E' + BE + AD'E$$

1.5.5.1 Karnaugh Map for 6-Variable System

In the case of six-variable system, we will be having 2^6= 64 cells. For ease of graphical derivation, these 64 cells are considered as four 4 variable Karnaugh maps as shown in Figure 1.14. First two variables AB can be used to select the four of the four Karnaugh maps.

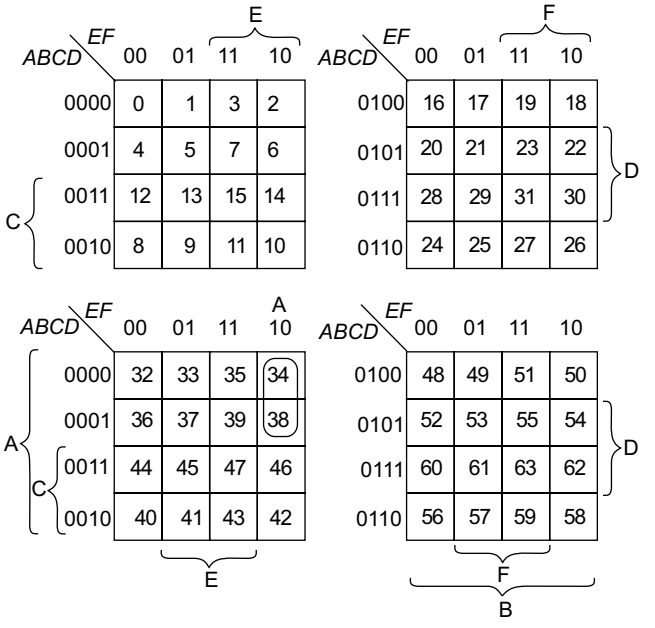


Figure 1.14 Grouping in the 6-variable Karnaugh Map example

In this also, we consider all of these four variable Karnaugh maps are stacked one by one while grouping the cells. For example, in the above figure some cells are rounded whose minterm numbers are 2, 6, 18, 22, 34, 38, 50 and 54. If assume these cells are having 1s, then we can consider them as two 1s in four layers. The Boolean equation for this group can be identified as: C'EF'.

For example, the following Karnaugh map shows a group of 4 1s which are spread across four four variable Karnaugh map and the respective Boolean term.

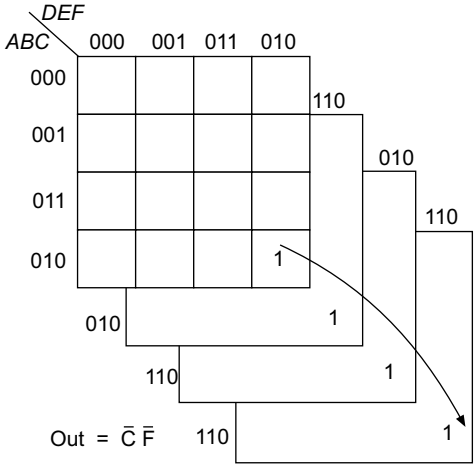


Figure 1.15 demonstrates the groupings in a 6-variable Karnaugh map and respective numbering of minterms. Readers are advised to observe the groupings carefully and check the respective prime implicants.

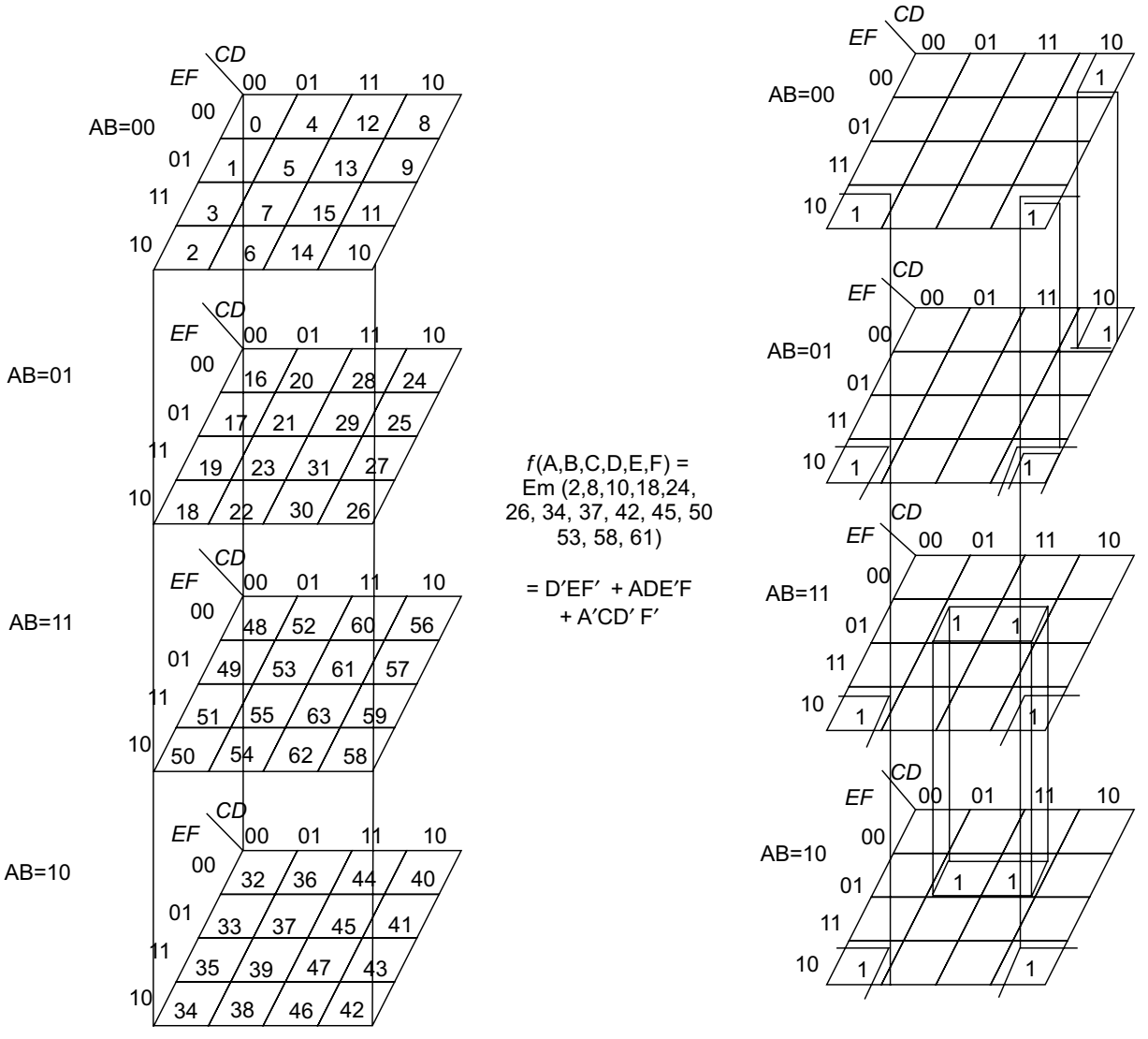


Figure 1.15 Grouping a 6-variable Karnaugh Map

1.6 Quine Mc_Cluskey Theory

Quine-McCluskey optimisation is named after two people: McCluskey and the contemporary philosopher W.V.O. Quine. The Quine McCluskey method is an algorithmic method that finds prime implicates, necessary prime implicates, and minimum sum-of-products expressions for digital systems. Its idea is same as using K-maps, except that K-maps are very hard to use with variables greater than (5). The advantages of this method compared to other methods like K-maps can be summarised in three main points:

- There is no limit in the number of the variables used in the function (theoretically no limit).
- The algorithm can be implemented directly and easily as a software program, which will reduce the human efforts and the possibilities of human errors.
- The final result of the algorithm is the totally minimised (SOP) function.

Minimising a five variable system like the following using K-Map approach is laborious and cumbersome. Thus, QMA method is proposed to find out simplified Boolean equation in table driven manner that can be realized through SW such that it becomes useful for system with large number of Boolean variables.

$$f(A, B, C, D, E) = \Sigma(0, 1, 3, 4, 5, 7, 9, 10, 12, 13, 21, 24, 26, 28, 29)$$

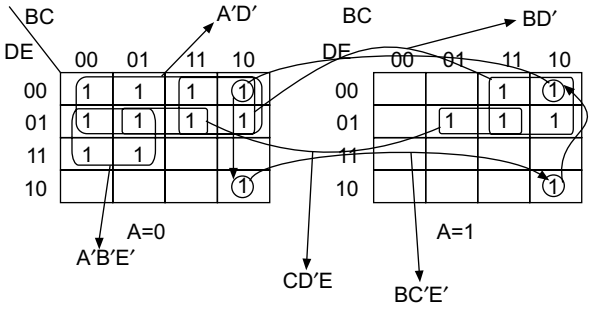


Figure 1.16

$\therefore$ **f = CD′E = A′B′E + A′D′ + BC′E′ + BD′**

In a nutshell, this approach contains two prominent stages: 1. To find out prime implicates, and 2. To find out essential prime implicates to get the simplified Boolean equation.

Like the K-map, the QM procedure uses the Boolean Law: xy +xy' = x to find the prime implicates by comparing and combining adjacent product terms.

The method is essentially a tabular one using a sequence of lists and charts to eventually zero in on the minimum solution.

The method also uses the fact that the indices of adjacent minterms differ by a power of 2, since adjacent minterms differ by only 1 variable.

The QM procedure can be given as follows.

Step 1

Create a list (LIST 1) of minterm indices, grouping them according to the number of 1's they contain. Initially, the indices are written in binary but later we will perform the procedure using decimal indices.

Step 2

Starting from group 0, compare each term in group I with the terms in group I+1, that is the adjacent group.

When two terms differ in one variable, put a mark (a tick will do) next to both terms.

The result of the combination is a term with all of the original variables but with a "–" in the location of the variable that has been eliminated.

The new term is entered into a 2^{nd} list (LIST 2). List 2 is completed when all possible comparisons from LIST 1 are performed.

Each group in LIST 2 (again grouped according to the number of 1's the terms contain) corresponds to combinations between a single pair of terms in LIST 1.

Step 3

Continue the process with LIST 2 to generate LIST 3. Compare only terms in adjacent groups AND which have the same variables eliminated from STEP 2 (i.e. a "–" in the same location).

Repeat the process until no more combinations are possible. If duplicate terms arise they can be discarded, but be sure to check the terms which are combined.

Step 4

The unchecked terms in all of the lists represents the PIs (prime implicates) of the function.

The next stage is to find the minimum cover.

This is done using a PI (prime implicates) chart and identifying the essential PIs. We shall explain the same with following example.

■ **Example** Simplify the function

$F(w, x, y, z) = \Sigma(0, 1, 2, 8, 10, 11, 14, 15)$

The binary equivalents of the minterms are shown in the following table:

Minterm w, x, y, z	Binary
0	0000
1	0001
2	0010
8	1000
10	1010
11	1011
14	1110
15	1111

Let us group them according to the number of 1's in each number:

List 1 w, x, y, z

Grp o	0	0000
Grp 1	1	0001
	2	0010
	8	1000
Grp 2	10	1010
Grp 3	11	1011
	14	1110
Grp 4	15	1111

We now compare adjacent groups, Grp 0 with Grp 1, Grp1 with Grp 2 and so on to generate LIST 2.

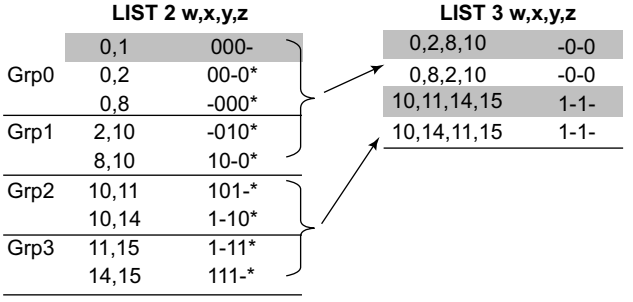
LIST 1 w,x,y,z			LIST 2 w,x,y,z	
Grp 0	0	0000*	0,1	000-
Grp 1	1	0001*	0,2	00-0
	2	0010*	0,8	-000
	8	1000*	2,10	-010
Grp 2	10	1010*	8,10	10-0
Grp 3	11	1011*	10,11	101-
	14	1110*	10,14	1-10
Grp 4	15	1111*	11,15	1-11
			14,15	111-

The asterisks check off the terms which have been combined.

The interpretation is as follows. If we examine minterms 0 and 1, they are equivalent to w'x'y'z' and w'x'y'z. These can be combined and simplified by eliminating the z variable to give w'x'y'.

Thus the 1st entry in LIST 2 which is 000-, is binary for w'x'y' with the "–" indicating the eliminated variable z.

LIST 2 is again grouped according to the number of 1's in each term. Adjacent groups are again combined, this time looking for terms with the "–" in the same position. The adjacent groupings already ensure that the terms only differ by one variable.



Note that no combination was possible between Grp1 and Grp2, since none of the entries contained a "–" in the same position.

We are left with 3 terms (the shaded entries) that cannot be combined further. The QMA algorithm has terminated. What we are left with are the Prime Implicants of the function.

The minimised function is then:

$$F = w'x'y' + x'z' + wy$$

This can be compared with the answer obtained from the K-map technique.

Recall from Kmap theory that in order to ensure a minimal solution we had to identify the Essential Prime Implicants (EPIs).

From the map these were the sets of ones that were covered only by 1 prime implicant.

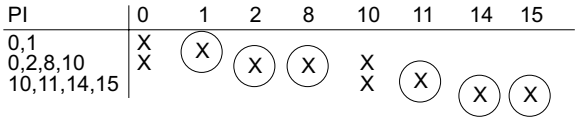
With the QM algorithm, a table called a PI chart is used to select the EPIs.

The chart tabulates the minterms in a row across the top of the table and the unchecked PIs in a column.

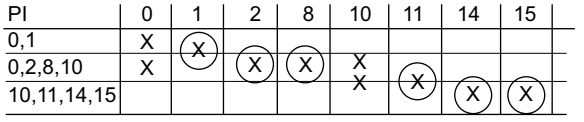
The chart is used as follows :

1. In each row, place an "X" in the column corresponding to each minterm of the PI.
2. When all the PI minterms have been identified, find all the columns which just 1 "X". Circle these X's. The corresponding PIs are the Essential PIs. Check off these PIs.
3. Draw a line through each EPI and then a vertical line through each minterm covered by that EPI (the X's in each row). These minterms are now covered and need not be considered any further.
4. The remaining PIs can now be selected to cover the remaining minterms. Each PI should be chosen to eliminate as many minterms as possible. Sometimes it is helpful to draw a second PI table containing the unselected minterms.
5. The minimal expression is now obtained from the EPIs and the remaining PIs from Step 4

Applying these rules to our example:



The circles represent minterms that are covered by only one PI. The corresponding PI is an EPI.



When we draw a line through the minterms and horizontally through the PIs, in this example we cover all of the minterms. There is no need to develop a second table.

The answer obtained from the LISTs was in fact minimal.

■ **Example** Minimise the following function using the QM algorithm:

$$f(a,b,c,d) = \Sigma(0,1,2,5,6,7,8,9,10,14)$$

Step 1: We develop the lists based on the number of ones in each minterm, group them and eliminate those which differ by one literal.

List 1		List 2		List 3	
0	√	0, 1 (1)	√	0, 1, 8, 9 (1, 8)	*
1	√	0, 2 (2)	√	0, 2, 8, 10 (2, 8)	*
2	√	0, 8 (8)	√	~~0, 2, 8, 10 (2, 8)~~	
8	√	0, 5 (4)	*	2, 6, 10, 14 (4, 8)	*
5	√	0, 9 (8)	√	~~2, 6, 10, 14 (4, 8)~~	
6	√	0, 6 (4)	√		
9	√	2, 10 (2)	√		
10	√	8, 9 (1)	√		
7	√	8, 10 (2)	√		
14	√	5, 7 (2)	*		
		6, 7 (1)	*		
		6, 14 (8)	√		
		10, 14 (4)	√		

Note that we have used decimal numbers. The procedure is the same in that numbers from adjacent sections are com-

pared with those *differing* in a power of 2 being combined. In List 2 both numbers are compared.
The PIs left are:

PIs	*abcd*	PIs
1,5 (4),	0001	*a'c'd*
5,7 (2),	0101	*a'bd*
6,7 (1);	0110	*a'bc*
0,1,8,9 (1,8)	0000	*b'c'*
0,2,8,10 (2,8)	0000	*b'd'*
2,6,10,14 (4,8)	0010	*cd'*

$$f = a'c'd + a'bd + a'bc + b'c' + b'd' + cd$$

Clearly this is not minimal. We go to the PI chart to determine the essential PIs

	0	1	2	5	6	7	8	9	10	14
1,5		x		x						
5,7				x		x				
6,7					x	x				
0,1,8,9	x	x					x	(x)		
0,2,8,10	x		x				x		x	
2,6,10,14			x		x				x	(x)

EPI EPI

We must include (0, 1, 8, 9) and (2, 6, 10, 14).
We now need to cover minterms 5 and 7 which we see can be done with PI (5, 7)
Hence the minimum solution is:

$$f_{\min} = b'c' + cd' + a'bd$$

1.7 NAND and NOR Implementation

Digital circuits are more frequently constructed with NAND or NOR gates than with AND and OR gates as NAND and NOR gates are easier to fabricate.

- NAND and NOR gates may be represented using alternative symbols.

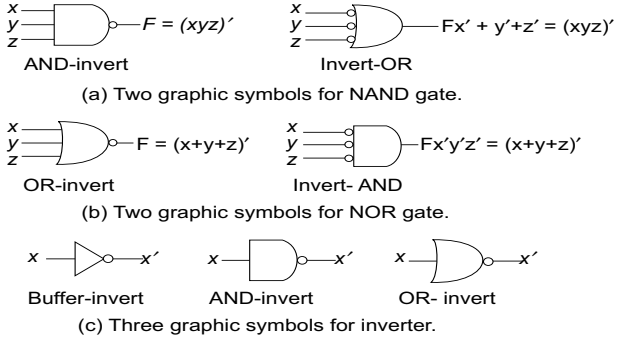


Figure 1.17 Graphic Symbols for NAND and NOR gates

- Inverters (NOT) may be represented by NAND and NOR gates whose inputs are all connected together.

NAND Implementation

- Requires the Boolean function to be simplified in SOP form

Method 1 (two logic levels)

(1) Simplify the function and express it in SOP form.
(2) Draw Level-1 gates
- One 2-input NAND gate with both inputs connected together for each primed literal
- One 2-input NAND gate with both inputs connected together for each unprimed literal that will go directly to Level-2
- One n-input NAND gate for each product term

(3) Draw Level-2 gate
- One n-input NAND gate using the INVERT-OR symbol

Method 2 (three logic levels)

(1) Simplify the *complement* of the function and express it in SOP form.
(2) Draw Level-1 gates
- One 2-input NAND gate with both inputs connected together for each primed literal
- One 2-input NAND gate with both inputs connected together for each unprimed literal that will go directly to Level-2
- One n-input NAND gate for each product term

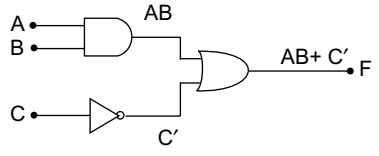
(3) Draw Level-2 gate
- One n-input NAND gate using the INVERT-OR symbol

(4) Draw Level-3 gate
- One 2-input NAND gate with both inputs connected together to invert the function

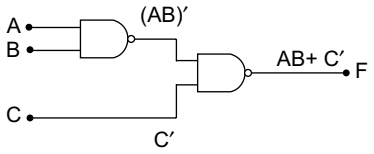
■ **Example** F = Σ(0, 2, 4, 6, 7)
Method 1, NAND logic

A\BC	B'C'	B'C	BC	BC'
A'	1	0	0	1
A	1	0	1	1

$F = AB + C'$



AND-OR-INVERT Logic



NAND Logic

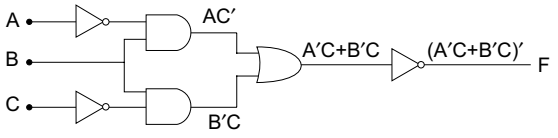
■ **Example** F = Σ(0, 2, 4, 6, 7)

Method 2, NAND logic

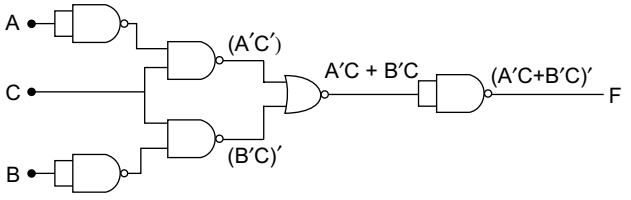
A\BC	B'C'	B'C	BC	BC'
A'	1	0	0	1
A	1	0	1	1

$F' = A'C + B'C$

$F = (A'C + B'C)'$



AND-OR-INVERT Logic



NAND Logic

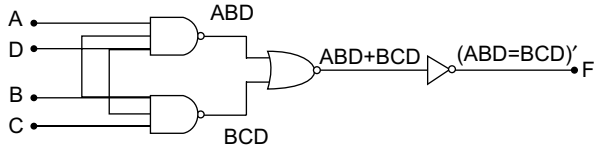
■ **Example** $F = \Pi(7, 13, 15)$

Method 2, NAND logic

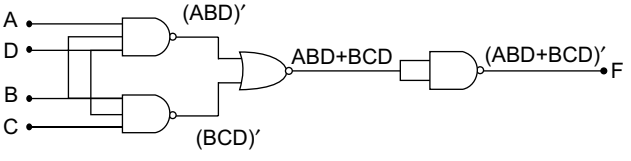
AB\CD	C'D'	C'D	CD	CD'
A'B'	1	1	1	1
A'B	1	1	0	1
AB	1	0	0	1
AB'	1	1	1	1

$F' = ABD + BCD$

$F = (ABD + BCD)'$



AND-OR-INVERT Logic



NAND Logic

NOR Implementation

- Requires the Boolean function to be simplified in POS form.

Method 1 (two logic levels)

(1) Simplify the function and express it in POS form.

(2) Draw Level-1 gates

- One 2-input NOR gate with both inputs connected together for each primed literal
- One 2-input NOR gate with both inputs connected together for each unprimed literal that will go directly to Level-2
- One n-input NOR gate for each product term

(3) Draw Level-2 gate

- One n-input NOR gate using the INVERT-AND symbol

Method 2 (three logic levels)

(1) Simplify the *complement* of the function and express it in POS form.

(2) Draw Level-1 gates

- One 2-input NOR gate with both inputs connected together for each primed literal
- One 2-input NOR gate with both inputs connected together for each unprimed literal that will go directly to Level-2
- One n-input NOR gate for each product term

(3) Draw Level-2 gate

- One n-input NOR gate using the INVERT-AND symbol

(4) Draw Level-3 gate

- One 2-input NOR gate with both inputs connected together to invert the function

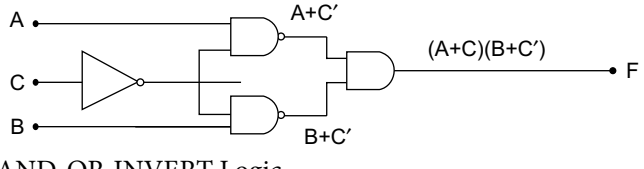
■ **Example** $F = \Sigma(0, 2, 4, 6, 7)$

Method 1, NOR logic

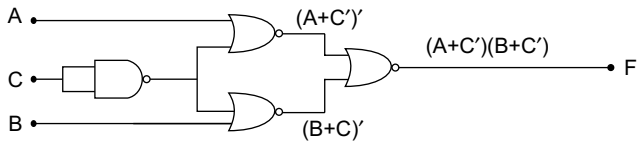
A\BC	B'C'	B'C	BC	BC'
A'	1	0	0	1
A	1	0	1	1

$F = A'C + B'C$

$= (A + C')(B + C')$



AND-OR-INVERT Logic



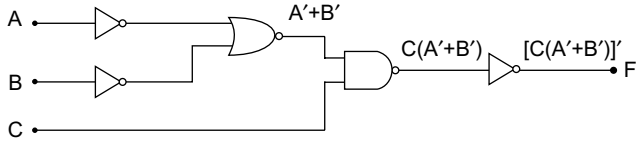
NOR Logic

■ **Example** $F = \Sigma(0, 2, 4, 6, 7)$

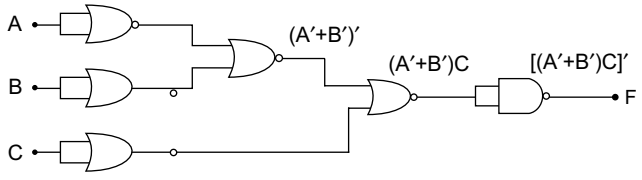
Method 2, NOR logic

A\BC	B'C'	B'C	BC	BC'
A'	1	0	0	1
A	1	0	1	1

$$F' = A'C + B'C$$
$$= C(A' + B')$$
$$F = [C(A' + B')]'$$



AND-OR-INVERT Logic



NOR Logic

Rules for NAND and NOR Implementation					
Case	**Function to simplify**	**Standard form to use**	**How to derive**	**Implement with**	**Number of levels to F**
(a)	F	SOP	combine 1's in map	NAND	2
(b)	F	SOP	combine 0's in map	NAND	3
(c)	F'	POS	complement F' in (b)	NOR	2
(d)	F'	POS	complement F in (a)	NOR	3

1.8 Conversions between Representations

In the previous sections, we have dealt many illustrative examples to convert Boolean equation in SOP to truth table, and vice versa. Also, we have employed many other examples to covert from one representation to another representation. In this section, we would like to summarise the same. Figure 1.18 illustrates the possible representations and conversions carried out in practice.

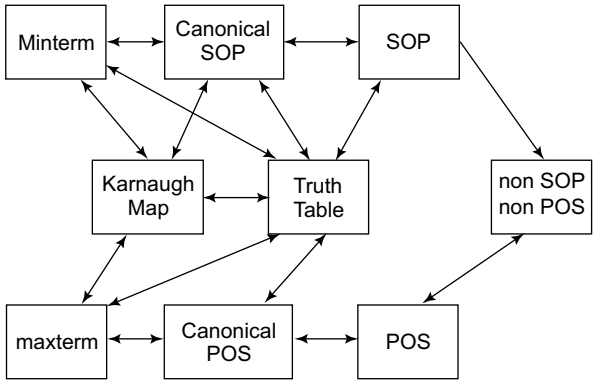


Figure. 1.18 Conversion between Representations

Converting Minterm to truth table or Karnaugh Map

Assuming that we were given a function as sum of minterms like f(A, B, C) = Σ(0, 4, 6, 7). To achieve the required thing, we shall follow the following steps:

1. Represent minterm numbers in binary fashion like: f(A,B,C) = Σ(000,100,110,111)
2. Place 1 in the truth table (Karnaugh Map) for each minterm entry.
3. Take all the remaining entries as 0s.

Converting Maxterm to truth table or Karnaugh Map

Assuming that we were given a function as sum of maxterms like f(A, B, C)= π (0, 4, 6, 7). To achieve the required thing, we shall follow the following steps:

1. Represent minterm numbers in binary fashion like: f(A, B, C) = π (000, 100, 110, 111)
2. Place 0 in the truth table (Karnaugh Map) for each maxterm entry.
3. Take all the remaining entries as 0s.

Converting Minterm to Canonical SOP

Assuming that we were given a function as sum of minterms like f(A, B, C) = Σ(0, 4, 6, 7). To achieve the required thing, we shall follow the following steps.

1. Represent minterm numbers in binary fashion like: f(A, B, C) = Σ(000, 100, 110, 111)
2. Replace 1s and 0s with respective literals and their complements such as f(A, B, C) = Σ(A′B′C′, AB′C′, ABC′, ABC) (Do remember the literals ordering)
3. Then take sum of the terms like: A′B′C′, + AB′C′+ ABC′+ ABC

Converting Canonical SOP to minterms

Reverse the steps of above.

Converting Maxterm to Canonical POS

Assuming that we were given a function as sum of maxterms like f(A,B,C)= π(0,4,6,7). To achieve the required thing, we shall follow the following steps.

1. Represent minterm numbers in binary fashion like: f(A,B,C)=π(000,100,110,111)
2. Replace 0s and 1s with corresponding literal and its complement and prepare sum terms like: f(A,B,C) = π(A+B+C, A′+B + C, A′+B′+C, A′ + B′ + C′)
3. Take product of the sum terms like: (A+B+C) (A′ + B + C) (A′ + B′+ C) (A′ + B′ + C′)

Converting Canonical POS to maxterms

Reverse the steps of above.

POS to SOP and vice versa

We can achieve this by using Boolean postulates which are explained in the previous chapters.

SOP to Canonical SOP

We can expand the product terms by replacing each of the missing literal(X) in the product terms with X+X′ till we get full canonical product sums. At the end, we can eliminate common terms.

POS to Canonical POS

We can expand the sum terms by replacing each of the missing literal(X) in the sum terms with XX′ till we get full canonical sum terms. At the end, we can eliminate common terms.

Canonical SOP to minimal SOP

We can use Karnaugh Maps and groups 1s.

Canonical POS to minimal POS

We can use Karnaugh Maps and group 0s.

Non-SOP to SOP and Non-POS to POS

We can employ Boolean postulates as explained in the previous section.

SOP to Truth Table

Place a logic 1 in each truth table output entry whose input value satisfies a given product term is 1. A k-variable product term will produce 2^{n-k} 1s in the truth table where n is the total number of input variables

POS to truth table

Place a logic 0 in each truth table output entry whose input value satisfies a given sum term is 0. A k-variable sum term will produce 2^{n-k} 0s in the truth table where n is the total number of input variables

1.9 XOR and XNOR patterns from Karnaugh Map (Reed-Muller Logic)

Some digital functions can be difficult to optimise if they are represented in the conventional sum-of-products or product-of-sums forms, which are based on ANDs, ORs, NANDs, NORs, and NOTs. In certain cases, it may be more appropriate to implement a function in a form known as Reed-Müller logic, which is based on XORs and XNORs.

One indication as to whether a function is suitable for the Reed-Müller form of implementation is if that function's Karnaugh Map displays a checkerboard pattern of 0s and 1s. Consider a familiar two-input function as shown in *the following examples.* That is, as of now we have concentrated square or rectangular groups of 1s or 0s in the Karnaugh map. In fact, we can group even diagonal cells with 1s or cells with some specific offset values. These groups may lead to XOR or XNOR terms as explained below.

Consider the figure.

C \ ab	00	01	11	10
0		1		1
1	1		1	

For the above Karnaugh Map, we can write simplified Boolean equation as: a ⊕ b ⊕ c.

Larger checkerboard patterns involving groups of 0s and 1s also indicate functions suitable for a Reed-Müller implementation. Larger checkerboard patterns involving groups of 0s and 1s also indicate functions suitable for a Reed-Müller implementation. Once we have recognised a checkerboard pattern, there is a quick "rule of thumb" for determining the variables to be used in the Reed-Müller implementation. Select any group of 0s or 1s and identify the significant and redundant variables, and then simply XOR the significant variables together (the significant variables are those whose values are the same for all of the boxes forming the group, while the redundant variables are those whose values vary between boxes). For example, consider Fig. 1.19 and respective simplified Boolean equations in XOR.

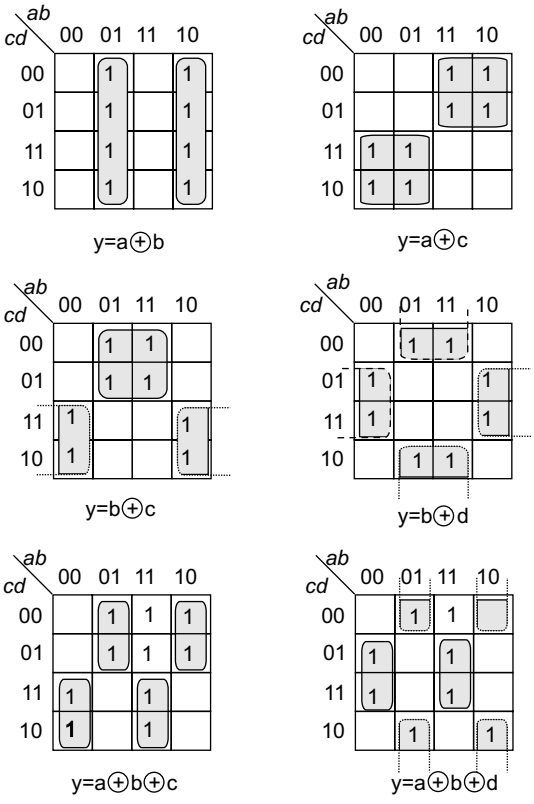
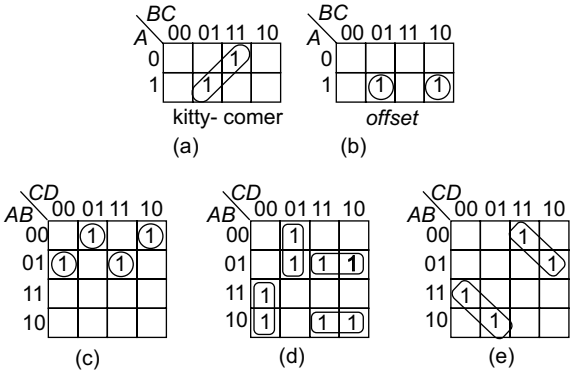


Figure 1.19

As all of the checkerboard patterns shown in the above illustrations include a logic 0 in the box in the upper left corner (corresponding to all of the inputs being logic 0), the resulting Reed-Müller implementations can be realized using only XORs. However, any pair of XORs may be replaced with XNORs, the only requirement being that there is an even number of XNORs.

Alternatively, if the checkerboard pattern includes a logic 1 in the box in the upper left corner, the Reed-Müller implementation must contain an odd number of XNORs. Once again, it does not matter which combinations of inputs are applied to the individual XORs and XNORs.
Reed-Müller implementations are often appropriate for circuits performing arithmetic or encoding functions.

■ **Example** Arrive Reed-Muller simplification of Karnaugh maps for the following functions.



In the figure (a) we have marked two cells which are in diagonal fashion as group. That is, A'BC and AB'C are grouped. Thus, we can say they represents (A ⊕ B)C.

Similarly, figure (b) contains two 1s separated by a single empty cell. If we join the respective minterms i.e., AB'C and ABC' we find they represents A(B ⊕ C).

Similarly, the third figure contains a special pattern of 1s. If we add them, we get

$$= A'B'C'D + A'B'CD' + A'BC'D' + A'BCD$$

$$= A'B'(C \oplus D) + A'B(C \oplus D)'$$

$$= A'[(B')(C \oplus D) + (B)(C \oplus D)']$$

$$= A'[B \oplus (C \oplus D)]$$

Similarly, figure (d) contains another pattern of 1s which we add we may get

$$= A'C'D + AC'D' + A'BC + AB'C$$

$$= C'(A \oplus D) + C(A \oplus B)$$

In the last figure, we have some other pattern of 1s in the Karnaugh map whose addition may give:

$$= A'B'CD + A'BCD' + ABC'D' + AB'C'D$$

$$= (A \oplus C)(B \oplus D)$$

■ Example What will be the nature of Karnaugh map for an equation F(A, B, C, D) = A ⊕ B ⊕ C ⊕ D

■ **Answer:** We know from the basic definition of 2-input XOR, its value is zero if both the inputs are same; otherwise 1. Thus, for this also we will get 0 if all the variables (literals) are prime (1111) or non-prime (0000), otherwise we will get 1. Thus, the Karnaugh map my look like the following.

	1	1	1
1	1	1	1
1	1	1	1
1	1	1	

Similarly, F' = (A ⊕ B ⊕ C ⊕ D)' may be having the following Karnaugh map.

1			
			1

1.10 Hazards and Glitches

Unexpected transient output changes are called *glitches*. A glitch is an unwanted pulse at the output of a combinational logic network – a momentary change in an output that should not have changed. A logic circuit is said to have

a *hazard* if it has the potential for these glitches. Thus, hazards are unwanted switching transients which may appear at the output of a combinational circuit and are caused by propagation delays of the input signals. To be specifically a hazard is a condition in a *logically correct* digital circuit or computer program that may lead to a logically incorrect output. Do note that a hazard is something intrinsic about a circuit; a circuit with hazard may or may not have a glitch depending on input patterns and the electric characteristics of the circuit.

There are two types of hazards, viz., function and logic hazards, if we classify them by specific causes. A function hazard is associated with multiple-input changes while a logic hazard is caused by a single-input change. Function hazards are not in the purview of this book. We can further classify hazards as being static or dynamic by their output waveforms. The presence of undesirable glitches at the output during the transition of two input states that have the same steady state output is called a static hazard. It is a static 0 (1)-hazard if the steady-state output is 0 (1). Dynamic hazard occurs when the steady-state output values of two input states are different and the output experiences more than one change.

Static Hazards

If the change of a *single* variable causes a momentary change in other variables, which should not occur, then a *static hazard* is said to exist. Rather, output of a gate which is expected to stay consistently either at 1 or 0, will not stay consistently. For a small duration, it may get reverses and comes back to its original level. There are two types of static hazards.

Static 1 hazard: Output should be at constant 1, but when one input is changed drops to 0 and then recovers to 1. It cannot occur in a POS implementation of the circuits.

Static 0 hazard: Output should be at constant 0, but when one input is changed rises to 1 and then drops back to 0. It cannot occur in an SOP implementation

Dynamic Hazards

If, after switching an input, the output has multiple transitions for a short time, then a *dynamic hazard* is said to be existing. It is impossible to see dynamic hazards in 2-level circuits. Dynamic hazards often occur in larger logic circuits where there are different routes to the output (from the input).

For example

- S/B: $0 \rightarrow 1$
- IS: $0 \rightarrow 1 \rightarrow 0 \rightarrow 1$

Now let us consider the following circuit in which x and x› (NOT x) are the input of the AND gate. We know for any value of x, we are supposed to get 0 as the output. However, it is not the case with the circuit because of the propagation delay of the NOT gate. Because of this delay, we get a spike as shown in Figure 1.20. That is, output value which is supposed to be 0 will raise to 1 for a while and then comes back to 0 again. This is the unwanted behavior of circuits because of the delays and is an example for static 0 hazard.

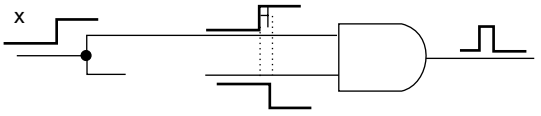


Figure 1.20 Glitch due to Delay

Similarly, the following circuit demonstrates the static 1 hazard. That is, we expect 1 from the OR gate. However, we may observe that the output will be changing to 0 for a while and returns back to 1 because of the delay of previous gates.

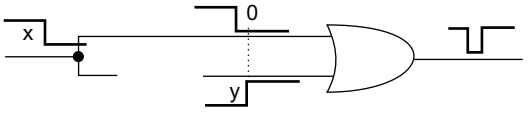


Figure 1.21 Static Hazard

Now, consider the following circuit x'y'+yz. If the input *y* is changed from 0 to 1, control of the output of the OR gate shifts from one AND gate to the other. Any difference in delays between the two AND gates will result in a glitch in the output of the OR gate.

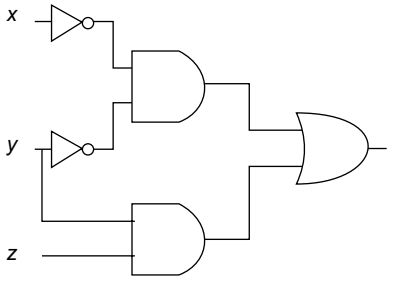


Figure 1.22 An Example Circuit having Hazard

Timing diagram for the above circuit further illustrates the static 1 hazard. We may find unexpected or unwanted drops in the output signal which indicates the static 1 hazard.

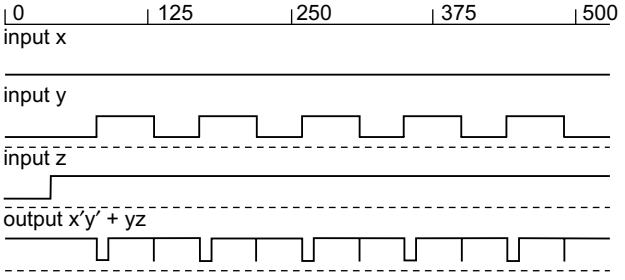


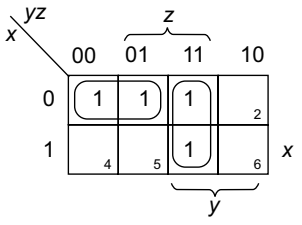
Figure 1.23 Timing Information of the Hazard of the Circuit above

Static 1 hazard detection using a Karnaugh map:
Reduce the logic function to a minimal sum of prime implicants. A Karnaugh map that contains adjacent, disjoint prime implicants is subject to a static 1 hazard. Similarly, we can also represent a function as product of sums by grouping maxterms (or 0s). Here also, if we find zero groupings which are adjacent and disjoint may give raise static 0 hazard.

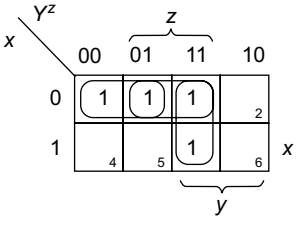
Adjacent prime implicants are the ones for which only one variable needs to change value to move from one prime implicant to the other.

Disjoint prime implicants are the one for which no prime implicant covers cells of both (all) of the disjoint prime implicants.

If we have two minterms and there is a literal in one minterm and its complement in the other minterm, and there exists just one pair of a literal and its complement, then these two minterms are adjacent and a hazard could occur between them.
For the above circuit, Karnaugh map is given as:



We may find two adjacent, disjoint prime implicants x'y', yz. These two prime implicants are adjacent as first implicant contains y' where as the other contains y. Thus, we can say they are adjacent and disjoint. Thus, it shows static 1 hazard. In order to avoid this hazard, we have to add a redundant minterm "to bridge the gap" between the terms. We can apply AND the remaining literals of both the minterms to get the redundant term. (This is applicable to maxterms in staic 0 hazard also). That is, the Karnaugh map with new term looks like:



The resultant equation F= x'y'+yz+x'z. The circuit which can be free from hazard is given as shown in Fig. 1.24:

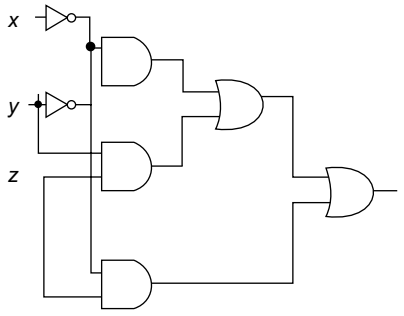


Figure 1.24 Circuit that is Free from Hazard

The timing diagram for the above diagram (Fig. 1.24) is shown in Fig. 1.25 which shows that the circuit is free from hazard.

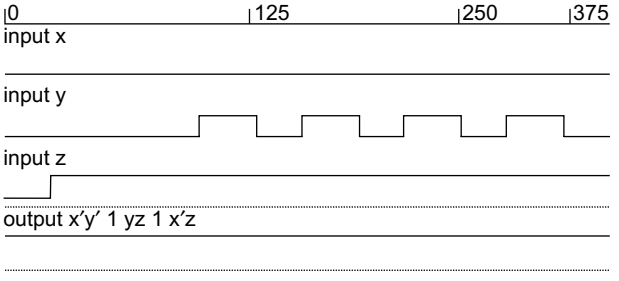


Figure 1.25 Timing details of the Circuit that is Free from Hazard

We can take care of hazards even by adding delays on the other paths. However, this is little cumbersome.

■ **Example** Will there be any hazard in the following function?

$$F = A' C' D' + ABC$$

■ **Answer:** In this function A and its complement occurred in two minterns, but also C and its complement occurred in the same two minterms. So the two minterms are not adjacent and no hazard occurs between them. As we can see the two minterms are not adjacent, so no hazard occurs.

■ **Example** Identify whether the following Boolean function has any hazards. If it has hazards problem, find out which redundant terms to be added to avoid the hazards.

$$F = A'D + ABC + AB'C'$$

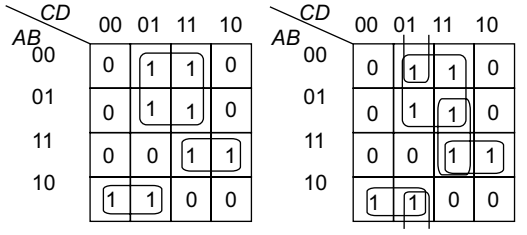
(1) (2) (3)

To detect and avoid hazard for this Boolean function the program implement the following sequence:

- Search for a pair of literal and its complement between (1) and (2), (A′ and A) were found.
- Check that it is the only pair between the two minterms.

- Because no other pair exists, AND the remaining literals in the two minterms, and store the resulting minterm in *hazard*. At this step *hazard* = BCD.
- Search for another pair between (1) and (3), (A′ and A) were found.
- Check that it is the only pair between the two minterms.
- Because no other pair exists, AND the remaining literals in the two minterms, and store the resulting minterm in *hazard*. The new added minterm will be ORed with the one produced before. At this step *hazard* = BCD + B′C′D.
- Search for a pair of literal and its complement between (2) and (3), (B and B′) was found.
- Check that it is the only pair between the two minterms. Because (C and C′ also exist in the two minterms, (2) and (3) are not adjacent and no hazard occurs between them.

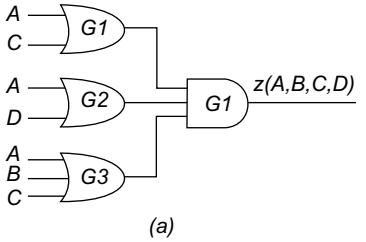
The final output of the Hazard function is hazard = BCD + B′C′D. The following figure contains the Karnaugh before and after adding the minterms to avoid hazards.



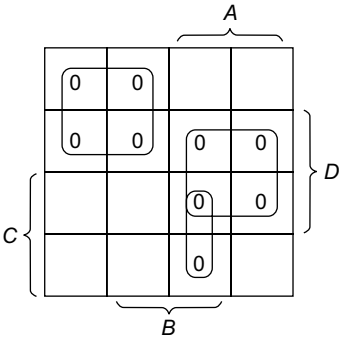
Do remember that we may find repeated to minterms as redundant terms when we apply the above automatic procedure. Thus, we can avoid repeated redundant minterms. Also, we may get a redundant minterm added to be a subset of another minterm. Thus, we can eliminate it in the final circuit design.

■ **Example** The following figure explains how to take care of static 0 hazard.

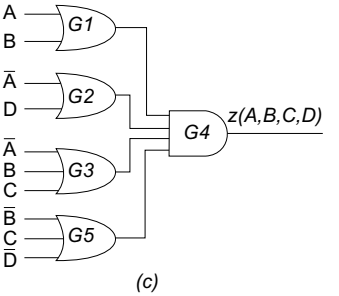
We may find a redundant term and as shown in the (d). The resultant modified circuit is also shown in the figure.



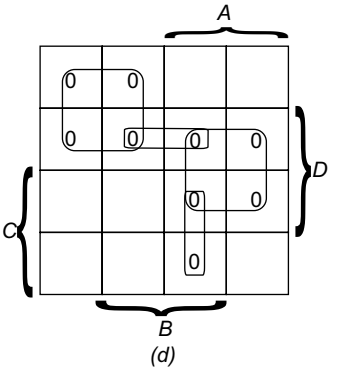
(a)



(b)



(c)



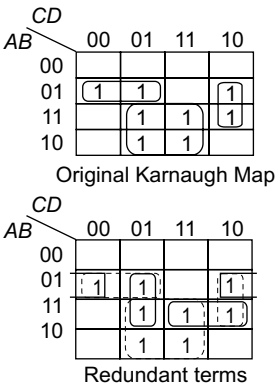
(d)

Why do hazards matter?

- The output of a hazard-prone circuit or program depends on conditions other than the inputs and the state
- The signal passed to another circuit by a hazard-prone circuit depends on exactly when the output is read
- In edge-triggered logic circuits, a momentary glitch resulting from a hazard can be converted into an erroneous output

■ **Example** Modify the equation F = A′BC′ + AD + BCD′ so that the logic is hazard free.

■ **Answer:** We prepare Karnaugh map and then identify the adjacent prime implicants. The following figure shows Karnaugh map and redundant minterms to be added to avoid hazards.



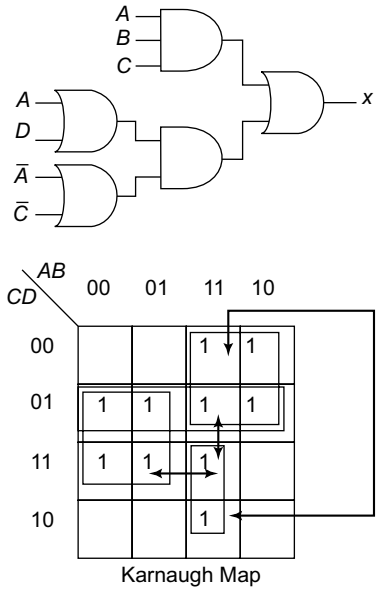
Notice there are three pairs of adjacent 1's not covered by an implicant. Transitions between these minterms can cause false outputs. We eliminate these hazards by adding redundant implicants to cover them. We add the prime implicants BC'D, ABC, and A'BD' giving the new equation:

$$F = A'BC' + AD + BCD' + BC'D + ABC + A'BD'$$

■ **Example** Can the logic equation F = X'Y + WZ + XZ have false outputs? Why or why not?

■ **Answer:** Yes. Draw the K-map to see that not all adjacent 1's are covered by the same implicants. For example, the W'XYZ and W'X'YZ terms are adjacent but are not both covered by a single term in the equation for F. As a result, if the inputs transition from WXYZ = 0111 to WXYZ = 0011, the terms X'Y and XZ could both be momentarily false, even though logically one of them should always be true. This is due to the delay caused by the inverter on X.

■ **Example** Consider the following circuit and analyse its susceptibility for hazards.



From the circuit, X=ABC+AC'+A'D+C'D. Karnaugh map is shown above along with hints about adjacent prime implicants. If ABCD = 1100 changes to 1000 then no hazard

- 1100 and 1000 is within the same 1-term group on the Karnaugh map
- *X* remains true because the term *AC* does not change

X=ABC+AC′+A′D+C′D

1100 0 1 0 0

1000 0 1 0 0

If ABCD = 0111 changes to 1111 then there could be a hazard

- 0111 and 1111 are adjacent but in different minterm groups on the Karnaugh map
- *X* could momentarily transition to 0

X=ABC+AC'+A'D+C'D

0111 0 0 1 0

1111 1 0 0 0

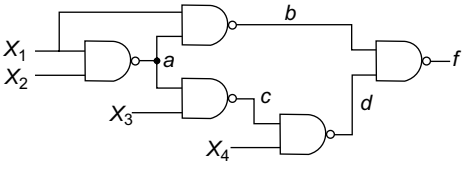
Thus, we need to add redundant terms. Terms ABC, A'D are adjacent. The redundant term becomes BCD. Similarly, AC' and A'D are adjacent. The redundant becomes C'D, which is already a prime implicant. Similarly, AC' and C'D are adjacent terms. Thus, redundant term become AD. Thus, in order to avoid hazards, we have to add BCD, and AD terms to our equation.

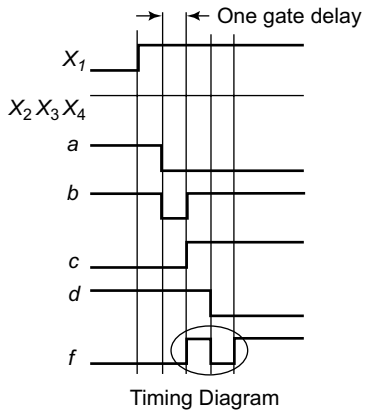
Dynamic Hazards

Dynamic hazards rarely occurs in two level circuits. They are commonly seen in multi-level circuits. Detecting dynamic hazards and alleviating them is tedious task. Thus, it is highly recommended to design practical circuits in two levels.

The following example illustrates the existence of dynamic hazard. For an output to change $0 \rightarrow 1 \rightarrow 0 \rightarrow 1$ (i.e. 3 times) in response to a single input change, there must be at least 3 paths of different length in the circuit

- 2 gate delays: $x_1 \rightarrow b \rightarrow f$
- 3 gate delays: $x_1 \rightarrow a \rightarrow b \rightarrow f$
- 4 gate delays: $x_1 \rightarrow a \rightarrow c \rightarrow d \rightarrow f$





Timing Diagram

■ **Example** How do you avoid hazards?

The fundamental strategy for eliminating a hazard is to add redundant prime implicants (extra prime implicants won't change F (final output of a circuit), but can cause F to be asserted independently of the change to the input that cause the hazard).

■ **Example** Design a combinational circuit to find 3 digit (binary) prime number finder. That is, input lines for this circuit are three and output line is 1. On the three input lines, we can have inputs from 000 to 111 which may represents decimal numbers from 0 to 7. If the number given is prime number, circuit has to give 1 as output, otherwise it should give 0 as output. The following table explains the required input and output relations ships. Of course, we do know that in the field of Mathematics, there exists debate on 0, 1 and 2 about their primality.

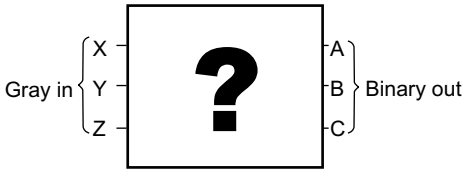
Decimal	a_2	a_1	a_0	Q
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

We can draw the Karnaugh map for the above system as:

$a_2 \backslash a_1\ a_0$	00	01	11	10
0	0	0	1	0
1	0	1	1	0

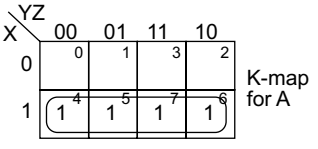
Thus, we can find the Boolean equation for Q = $a_1a_0 + a_2a_0$

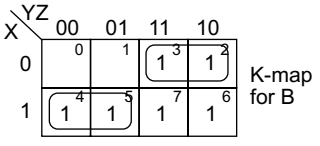
■ **Example** Design 3 bit gray code to 3 bit binary code. We have given gray codes in chapter on number system. We assume X, Y, Z as Boolean variables representing the bits of 3 bit gray code, while A, B, C as the Boolean variable representing bits of required binary code.

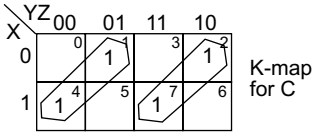


This can be represented as finding the block box content which satisfies input and output requirement. The following truth table demonstrates the possible input and output requirements of the required circuit. We have prepared Karnaugh for each of the output variable A, B and C.

X Y Z	A B C
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 1
0 1 1	0 1 0
1 0 0	1 1 1
1 0 1	1 1 0
1 1 0	1 0 0
1 1 1	1 0 1







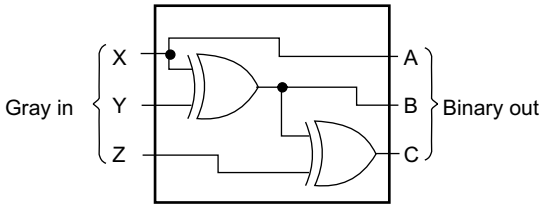
From the Karnaugh maps and their groupings, we can write equations for A, B, and C as:

A = X

B = XY'+X'Y= X ⊕ Y

C = XY'Z'+X'Y'Z + XYZ + X'YZ' = X ⊕ Y ⊕ Z

The resultant circuit is:



■ **Example** Four bit Binary to Gray code and vice versa.

Binary to gray code:

Truth Table:

No.	Binary				Gray			
	D	C	B	A	G3	G2	G1	G0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1

No.	Binary				Gray			
7	0	1	1	1	0	1	1	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

We shall draw Karnaugh maps for each of them and then find out the simplified Boolean equations for each of the ABCD. Karnaugh map for A can be given as:

	1		1
1		1	
	1		1
1		1	

Karnaugh map for B can be given as:

	1		1
	1		1
1		1	
1		1	

Karnaugh map for C can be given as:

	1		1
	1		1
	1		1
	1		1

If one observes the D column in the truth table, it is same as G3. Thus, equations for ABCD can be arrived as:
Possible Equations for A, B, C and D are given as:

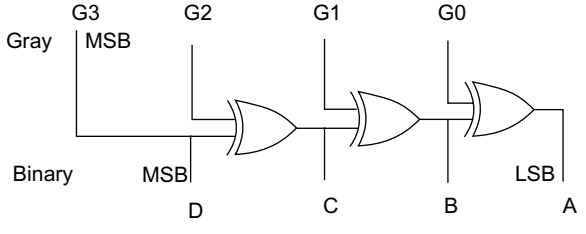
$$A = (G3 \oplus G2) \oplus (G1 \oplus G0)$$

$$B = (G3 \oplus G2 \oplus G1)$$

$$C = G3 \oplus G2$$

$$D = G3$$

Possible circuit diagram for the above is:



We can do reverse design also. That is, given binary code we want gray code. That is, given ABCD, we want G3G2G1G0. We shall draw Karnaugh maps for each of them and then find out the simplified Boolean equations for each of the G3G2G1G0.
Karnaugh map for G0 can be given as:

1	1	1	1
1	1	1	1

Karnaugh map for G0 can be given as:

	1	1	
	1	1	
1			1
1			1

Karnaugh map for G0 can be given as:

	1		1
	1		1
	1		1
	1		1

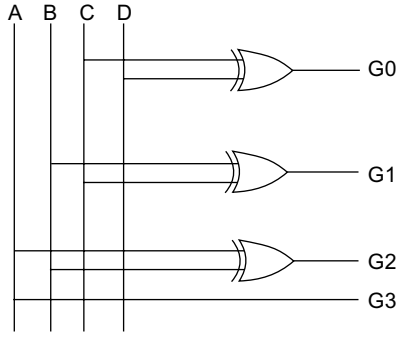
If one observes the G3 column in the truth table, it is same as D. Thus, equations for G3G2G1G0 can be arrived as:

$$G0 = B \oplus A$$

$$G1 = C \oplus B$$

$$G2 = D \oplus C$$

$$G3 = D$$



■ **Example** Refer standard books on number systems for bit stuffing for error detection and correction. We take that bit number 1 (2^0), 2 (2^1), 4(2^2), 8(2^3) etc., are the parity bits. If we assume, 8 bit data is used then we will be having 4 parity bits indexed as 1, 2, 4 and 8. If we assume, C_1, C_2, ,,,,C_{12} are the parity bit stuffed data, then C_1, C_2, C_4, C_8 are the

parity bits. We take about how to calculate values for parity bits from the data bits. The same can be represented as:

$$C_1 = C_3 \oplus C_5 \oplus C_7 \oplus C_9 \oplus C_{11}$$

$$C_2 = C_3 \oplus C_6 \oplus C_7 \oplus C_{10} \oplus C_{11}$$

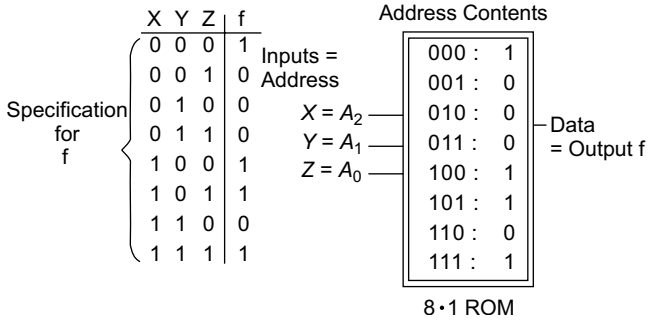
$$C_4 = C_5 \oplus C_6 \oplus C_7 \oplus C_{12}$$

$$C_8 = C_9 \oplus C_{10} \oplus C_{11} \oplus C_{12}$$

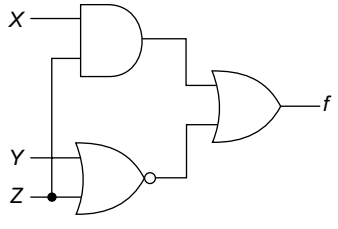
Draw circuit diagram from the above equations.

1.11 Realising Combinational Circuits Using ROMS

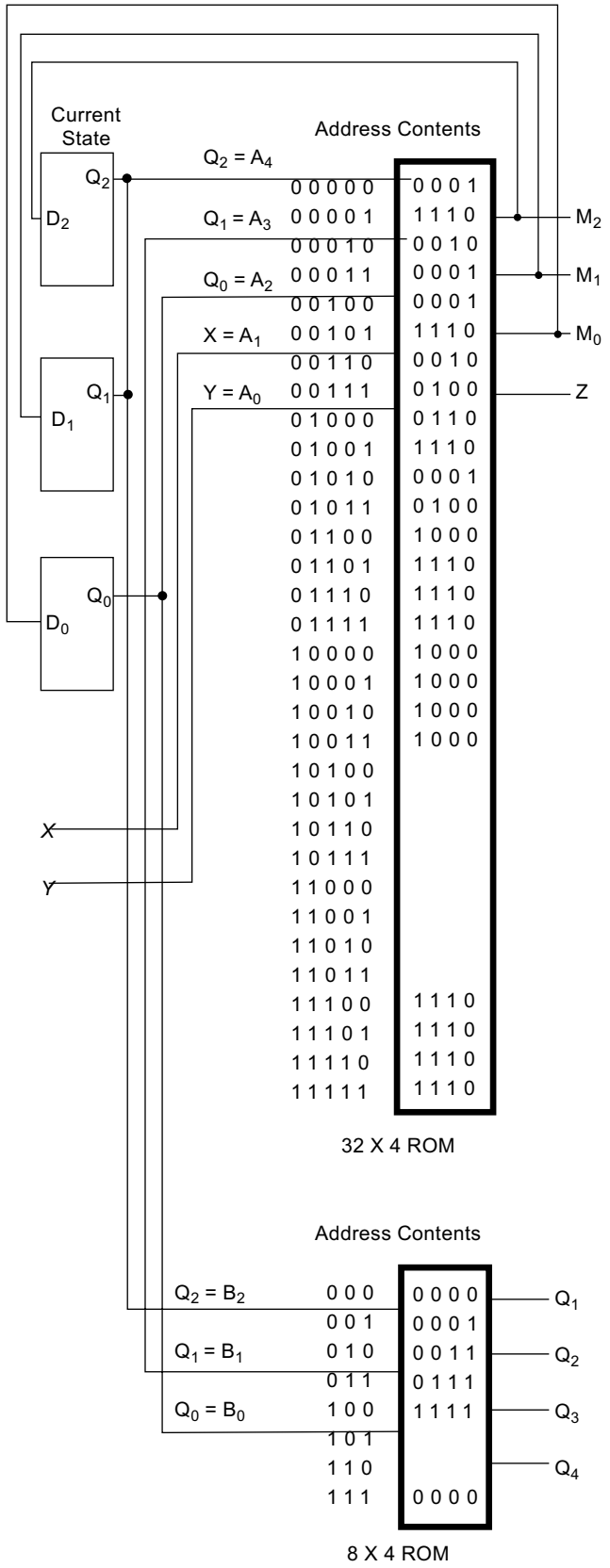
The information in the truth table specification for a combinational circuit can be viewed as specifying the contents for a ROM implementation of the circuit; e.g., the circuit specification for f below can be implemented by an 8 H 1 ROM whose contents are the given by the following specification. Simply we store function f values at the addresses (XYZ) of ROM. That is, at 000 we store 1, at 001 we store 0, and vice versa. When we want this function, simply we present XYZ values to ROM as address which outputs the stored value at that address which is nothing but our f values which we have stored earlier.



■ **Example** See the following combinatorial circuit. Does it emulates the above function f?

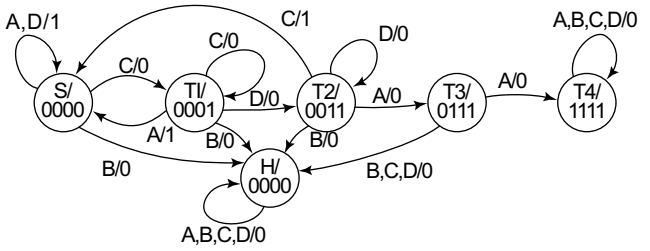


■ **Example** Consider the following circuit which contains a set of three D flip-flops and RAMS. Analyse working of this circuit assuming initial state of system is 000. That is, all the three flip-flops contains 000. Also, assume XY values are changed as 00, 01, 10, and 11 in continuous cyclic fashion. Prepare state diagram for the same.



As mentioned above, the current state information is maintained in the 3 D flip-flops given by Q2, Q1, Q0. The next state is given by the memory data lines labeled M2, M1, M0.

The M2, M1, M0 output values are applied to the inputs of the D flip-flops, ready to be latched on the transition to the next state. Since the output associated with each state does not rely on the transition inputs X, Y, a smaller memory unit is sufficient for representing this requirement of the circuit specification. Thus, 8x4 ROM is used to store output of the state.



We have observed 6 possible states of the system which we have labeled as S(Start denoted 000), T1 (001), T2(010), T3(011), T4(101) and H (Halt denoted by 111). The possible input states and state outputs are given in the following tables.

Inputs	X	Y
A	0	0
B	0	1
C	1	0
D	1	1

States/State Outputs	0_1	0_2	0_3	0_4
S	0	0	0	0
T1	0	0	1	1
T2	0	1	0	1
T3	0	1	1	1
T4	1	0	0	1
H	1	1	1	0

If we observe the circuit diagram, XY and output of the three flip-flops are used to address the 32x4 ROM while the three flip-flops output are used to address 8x4 ROM. Thus, now consider that 000 is available in the three flip-flops. That is system in state S. If we assume XY is made as 00, then address of 32x4 ROM is 00000 and for smaller RAM is 000. We find 0001 and 0000 at those locations. Thus, 000 will be taken as next state of system, 1 becomes Z (output), while 0000 becomes state outputs. Next if we consider XY as 01. Then the address for larger RAM becomes 00001 where we find value as 1110. That is, next state as 111 and output Z as 0. At the same time, we will be getting 0000 from the smaller ROM which is state outputs. In this fashion, this system changes its states. The following table elaborates the working of the above circuit.

	Q_2	Q_1	Q_0	X	Y	Q_2^n M_2	Q_1^n M_1	Q_0^n M_0	Z
S	0	0	0	0	0	0	0	0	1
	0	0	0	0	1	1	1	1	0
	0	0	0	1	0	0	0	1	0
	0	0	0	1	1	0	0	0	1
T_1	0	0	1	0	0	0	0	0	1
	0	0	1	0	1	1	1	1	0
	0	0	1	1	0	0	0	1	0
	0	0	1	1	1	0	1	0	0
T_2	0	1	0	0	0	0	1	1	0
	0	1	0	0	1	1	1	1	0
	0	1	0	1	0	0	0	0	1
	0	1	0	1	1	0	1	0	0
T_3	0	1	1	0	0	1	0	0	0
	0	1	1	0	1	1	1	1	0
	0	1	1	1	0	1	1	1	0
	0	1	1	1	1	1	1	1	0
T_4	1	0	0	0	0	1	0	0	0
	1	0	0	0	1	1	0	0	0
	1	0	0	1	0	1	0	0	0
	1	0	0	1	1	1	0	0	0
	—	—	—	—	—	—	—	—	—
H	1	1	1	0	0	1	1	1	0
	1	1	1	0	1	1	1	1	0
	1	1	1	1	0	1	1	1	0
	1	1	1	1	1	1	1	1	0

■ **Example** Realise F(A, B, C, D) = A′BC + AD + AC using ROM.

■ **Answer:** First, we will expand the above product terms into canonical terms and eliminate common terms. The final equations becomes:

$$F(A.B.C,D) = A'BCD + A'BCD' + AB'C'D + AB'CD + ABC'D + ABCD + AB'CD' + ABCD'$$

0111 0110 1001 1011 1101 1111 1010 1110

$$= \Sigma\, m(6, 7, 9, 10, 11, 13, 14, 15)$$

Thus, ROM with 16x1 should contains Function F values. ABCD are the addresses for the RAM.

ROM Address ABCD	ROM Contents
0000	0
0001	0
0010	0
0011	0
0100	0
0101	0
0110	1
0111	1
1000	0
1001	1
1010	1
1011	1
1100	0
1101	1
1110	1
1111	1

■ **Example** A ROM based combinatorial circuit which takes a 4-bit 2s complement number N and outputs (N+25) mod 16 is proposed to be designed. Assume we have 32 × 8 bit ROM at our disposal. Do calculate how many ROM locations gets unused and also calculate ROM content bits will be unused in our schema of function storage.

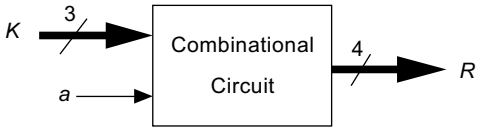
■ **Answer:** First, we have to prepare truth table which elaborates the input/output requirements. The truth table is given as:

Input	Output
0000	1001
0001	1010
0010	1011
0011	1100
0100	1101
0101	1110
0110	1111
0111	0000
1000	0001
1001	0010
1010	0011
1011	0100
1100	0101
1101	0110
1110	0111
1111	1000

ROM Address	ROM Contents
00000	00001001
00001	00001010
00010	00001011
00011	00001100
00100	00001101
00101	00001110
00110	00001111
00111	00000000
01000	00000001
01001	00000010
01010	00000011
01011	00000100
01100	00000101
01101	00000110
01110	00000111
01111	00001000

Thus, we have to store output column of the above table in ROM. As, ROM contains 8 bit words these four bit outputs has to be stored in least significant 4 bits of each word.

■ **Example** Design a combinational circuit which takes a 3-bit signed number (K) and gives K+1 as R if external input a = 0 else gives N^2 mod 7 as R. Realise this circuit using 16x2 ROM chips.



■ **Answer:**

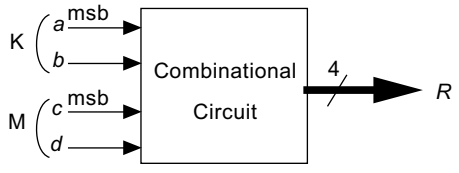
a	Input	Input in Decimal	Required Output	Output in Binary
0	000	0	1	0001
0	001	1	2	0010
0	010	2	3	0011
0	011	3	4	0100
0	100	0	1	0001
0	101	–1	0	0000
0	110	–2	–1	1001
0	111	–3	–2	1010
1	000	0	0	0000
1	001	1	1	0001

a	Input	Input in Decimal	Required Output	Output in Binary
1	010	2	4	0100
1	011	3	9 mod 7 = 2	0010
1	100	0	0	0000
1	101	–1	1	0001
1	110	–2	4	0100
1	111	–3	9 mod 7 = 2	0010

In order to store the output, we need two 16x2 ROM. The following figure illustrates the contents of ROM.

ROM Address	ROM0	ROM1
0000	00	01
0001	00	10
0010	00	11
0011	01	00
0100	00	01
0101	00	00
0110	10	01
0111	10	10
1000	00	00
1001	00	01
1010	01	00
1011	00	10
1100	00	00
1101	00	01
1110	01	00
1111	00	10

■ **Example** Design a combinational circuit which takes two 2-bit unsigned integers (K, M) and gives their output R as unsigned integer. Implement the same using ROM.



■ **Answer:** We consider ab lines for K and cd lines for M as shown in the figure. We know largest possible two bit unsigned number is 11(3). Thus, the product will be at most 9 for which we need 4-bits at most. Thus, output R is considered as 4-bit number as shown in the figure. We consider abcd as the address to the ROM having 4-bit result R. The following figure demonstrates the ROM contents.

ROM Address	ROM
0000	0000
0001	0000
0010	0000
0011	0000
0100	0000
0101	0001
0110	0010
0111	0011
1000	0000
1001	0010
1010	0100
1011	0110
1100	0000
1101	0011
1110	0110
1111	1001

■ **Example** Specify a truth table for a ROM which implements:

F = AB + A'BC'

G = A'B'C + C'

H = AB'C' + ABC' + A'B'C

■ **Answer:** First, we calculate functions F, G, and H values for each combination of input variables A, B, and C. These values are stored in ROM as shown this in figure.

ROM Address	F	G	H
000	0	1	0
001	0	1	1
010	1	1	0
011	0	0	0
100	0	1	1
101	0	0	0
110	1	1	1
111	1	0	0

ROM Content

1.12 Introduction to Sequential Circuits

Unlike combinatorial circuits, sequential circuit's state or output depends on the previous state or outputs in addition to current inputs. Evidently, sequential circuits contain memory elements in addition to other digital components (Fig. 1.26). Rather, one can state that previous contents of memory elements influence the outputs and possible next values (states) of the memory elements.

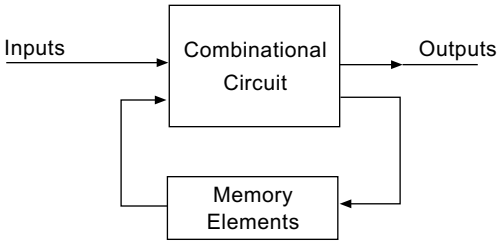


Figure 1.26 A typical Sequential Circuit

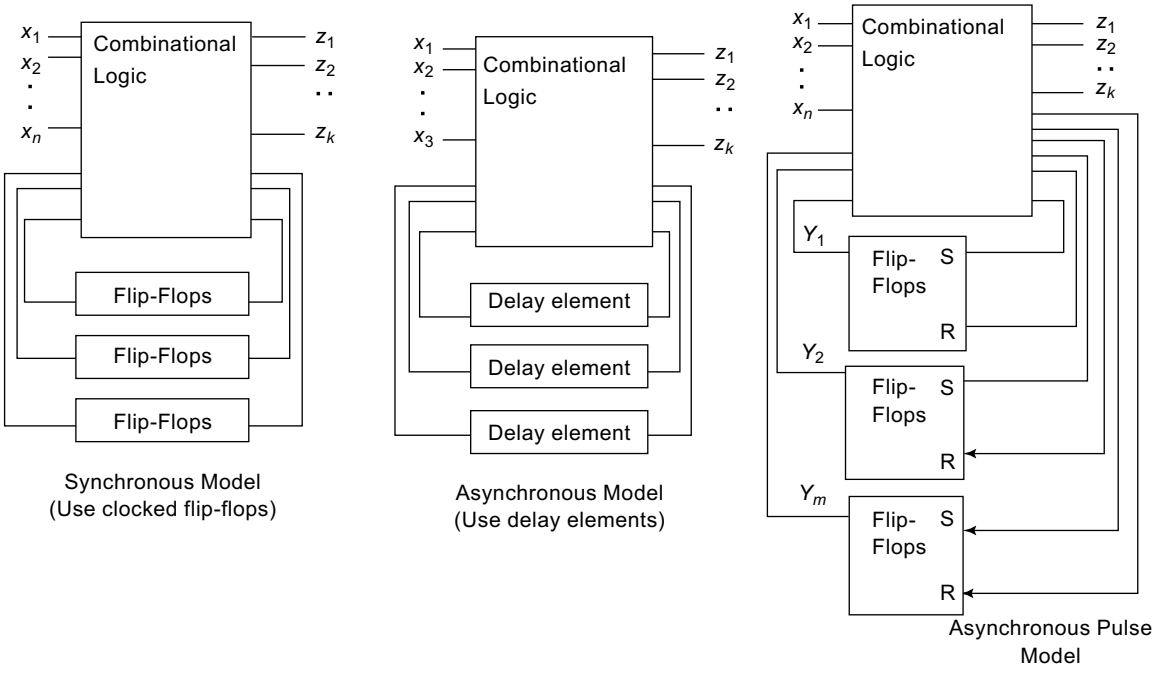


Figure 1.27 Classification of Sequential Circuits

Sequential Circuits are classified into:

1. **Synchronous:** Common Clock is used with Flip-Flops (i.e. elementary memory elements)
2. **Asynchronous**: Employs delays as memory elements. Only non zero delays are used as memories.
3. **Pulse Mode:** Employs asynchronous flip-flops such as the ones with **Reset, Clear, Preset.**

Figure 1.27 illustrates the difference among the variants of the sequential circuits.

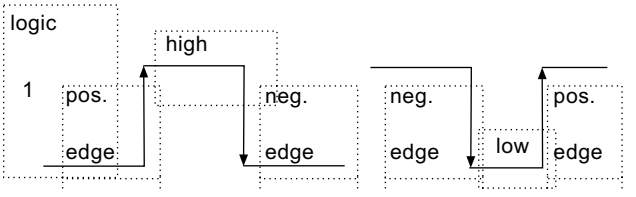
Further, we know that digital systems respond to high level or low level or transitions in voltage. Based on their response characteristics especially in response to clock signal, they are classified as:

High level trigger: a response that occurs in the presence of a high voltage level in the clock signal

Low level trigger: a response that occurs in the presence of a low voltage level in the clock signal

Positive edge-trigger is a response to a low-to-high voltage or clock signal transition

Negative edge-trigger: is a response to high-to-low voltage or clock signal transition



1.12.1 Flip-flops

A flip-flop is a semiconductor device that has a digital output which can be toggled between two stable states by providing it with the appropriate digital input signals. Once the output is put in one state, it remains there until a change in the inputs causes it to toggle again. This toggling between two logic states is also referred to as 'flip-flopping.' There are several types of flip-flops, few of which are described as follows.

Three classes of flip-flops are commonly employed in digital design based on their response.

- **latches:** outputs respond immediately while enabled (no timing control)
- **pulse-triggered flip-flops:** outputs response to the triggering pulse
- **edge-triggered flip-flops:** outputs responses to the control input edge

1.12.1.1 The Set-Reset (S-R) Flip-flop

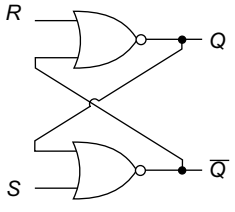
The Set-Reset (SR) flip-flop refers to a flip-flop that obeys the truth table shown in Table 1.4. It will be having two inputs, namely, a **Set** input, or **S**, and a **Reset** input, or **R**. It also has two outputs, the main output Q and its complement Q.

Table 1.4 The S-R Flip-flop Truth Table

S	R	Q_{N+1}	Q'_{N+1}
0	0	Q_N	Q'_N
0	1	0	1
1	0	1	0
1	1	Not Used	

A simple representation of an S-R flip-flop is a pair of cross-coupled NOR gates, i.e., the output of one gate is tied to one of the two inputs of the other gate and vice versa (see the circuit used at the beginning of this section which is used to explain sequential circuits. Just replace NAND with NOR). The free input of one NOR gate is used as R while the free input of the other gate is used as S (Figure 1.28)

The output of the gate with the 'R' input is used as the Q output while the output of the gate with the 'S' input is used as the Q output. Thus, resetting an S-R flip-flop's output Q to '0' requires R=1 and S=0, while setting Q to '1' requires S=1 and R=0.



S	R	Q	$\overline{Q}$
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	0	0

Figure 1.28 SR Flip-Flop

There are two common versions of the SR Flip-Flop:

Low to High activated SR Flip-Flop.

A Flip-flop (figure 1.28) made from two NORs. The output will latch when an appropriate Low to High transition is sent to an input.

High to Low activated Flip-Flop

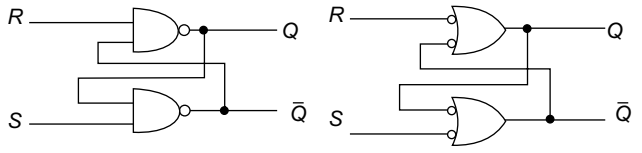
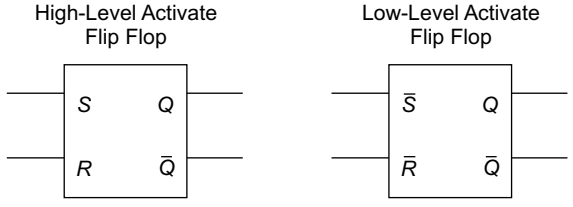


Figure 1.29 High to Low activated SR flip-flop (a) with NAND (b) OR with inverted inputs

If an SR Flip-Flop is built out of NAND gates instead of NOR gates, the flip flops output is latched when a High to Low transition occurs at an input state. While the circuit is correctly shown using the NAND gates, the preferred gate diagram is to use ORs with inverted inputs indicating that Low levels activate the flip-flop.

When working with digital IC chips, these flip flops are shown using the following alternate circuit diagrams:



Timing Diagram

Since Flip-Flops are sequential devices, truth tables are not enough. The state levels are sometimes shown by use of timing charts. It's easier to keep track of the states using timing charts especially for devices which are triggered by edge transitions. Figure 1.30 shows timing diagram of an SR flip-flop. We may find when **SET** line goes from low to high, output is becoming high. Similarly, even after **SET** lows changes from high to low, output is high, which supports that it has remembered 1. Same behavior is observed with **RESET** line also.

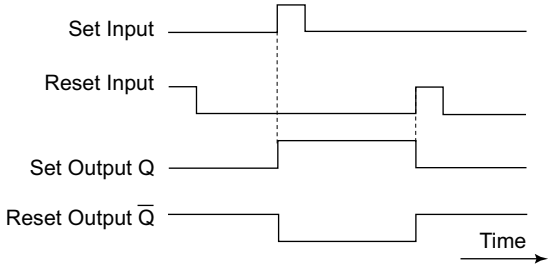


Figure 1.30 Timing Details of an SR flip-flop

■ **Example** Fill in the table below assuming that the flip-flop is High to Low activated with ORs as shown in Figure 1.29.

S	R	Q	$\overline{Q}$
1	0	1	0
1	1	X	X
0	1	0	1
1	1	X	X
1	0	1	0
1	1	X	X
0	1	1	0
0	0	1	0

Time

Latch vs Flip-Flop (The Unbroken Dilemma)

Latch— a device where any data change on the input will be transferred to the output whenever the proper logic level is present on the **enable** input.

Flip-Flop—a device where data on the input will be transferred to the output when the proper edge-trigger occurs on the **clock** input.

In real-world applications, flip-flops are 'clocked' so that one can control the exact moment at which the output changes its state in response to changes in inputs. The clock digital input of clocked flip-flops is usually denoted as C.

The fundamental latch is the simple *SR flip-flop*, where S and R stand for set and *reset,* respectively. It can be constructed from a pair of cross-coupled NAND logic gates as shown in the figure. The stored bit is present on the output marked Q.

Normally, in storage mode, the S and R inputs are both low, and feedback maintains the Q and Q outputs in a constant state, with $\overline{Q}$ the complement of Q. If S (***Set***) is pulsed high while R is held low, then the Q output is forced high, and stays high even after S returns low; similarly, if R (***Reset***) is pulsed high while S is held low, then the Q output is forced low, and stays low even after R returns low (Table 1.5).

In electronics design, an **excitation table** shows the minimum inputs that are necessary to generate a particular next state when the current state is known. They are similar to truth tables and state tables, but rearrange the data so that the current state and next state are next to each other on the left-hand side of the table, and the inputs needed to make that state change happen are shown on the right side of the table. Excitation table for an SR Flip Flop ("X" is "don't care") is shown in Table 1.5. Here, Q(t) is the flip-flop state at time t and Q(t+1) is its state at time t + 1, i.e., next state. In some books, we may find the use of Q_N and Q_{N+1} also for these quantities.

The behavior of a flip flop can be described by a **characteristic table** which is basically a truth table. It contains short description about actions associated with each input. For example, Table 1.5 contains characteristic table of SR flip-flop.

Table 1.5 Characteristic and excitation table of S-R Flip-Flop

SR Flip-Flop operation ('X' denote a Don't care condition; meaning the signal is irrelevant)							
Characteristic table			**Excitation table**				
S	R	Action	Q (t)	Q (t + 1)	S	R	Action
0	0	Keep state	0	0	0	X	No Change
0	1	Q = 0	0	1	1	0	Set
1	0	Q = 1	1	0	0	1	Reset
1	1	Unstable combination,	1	1	X	0	No Change

1.12.1.2 The JK Flip-flop

The JK flip-flop is a flip-flop that obeys the truth table in Table 1.6. The J-K flip-flop differs from the S-R flip-flop in the sense that its next output is determined by its present output state as well, aside from the states of its inputs. Note that in the J-K flip-flop, the S input is now called the J input and the R input is now called the K input. Thus, in a JK flip-flop, the output will not change if both J and K are '0', but will toggle to its complement if both inputs are '1'. Unlike SR (or RS) flip-flop, with JK flip-flop there is no racing condition when both the inputs are 1s. Instead, if both J,K inputs becomes 1, flip-flop output gets complimented as shown in the following characteristic table of JK flip-flop in Table 1.6.

Table 1.6 J-K Flip-flop's Truth Table

J	K	Q_{N+1}
0	0	Q_N
0	0	0
1	0	1
1	1	Q'_N

Figure 1.31 gives details about the JK flip circuit and its symbol that is used commonly in the digital designs.

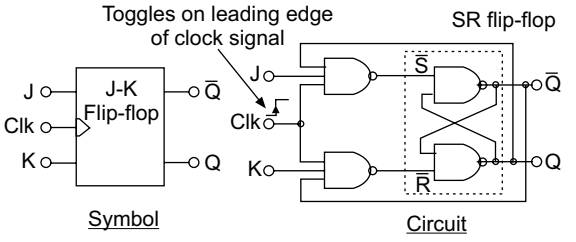


Figure 1.31 JK Flip-Flop Circuit and Symbol that are commonly used.

1.12.1.3 Master-Slave J-K Flip-Flops

The Master-Slave Flip-Flop is basically two J-K bi-stable flip-flops connected together in a series configuration with the outputs from Q and Q from the "Slave" Flip-flop being fed back to the inputs of the "Master" with the outputs of the "Master" flip-flop being connected to the two inputs of the "Slave" Flip-flop as shown in figure 1.32.

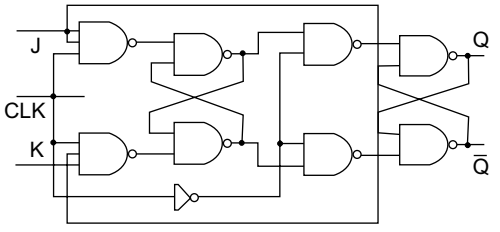


Figure 1.32 Master Slave JK Flip-flop

The input signals J and K are connected to the "Master" Flip-flop which "locks" the input while the clock (CLK) input is high at logic level "1". As the clock input of the "Slave" Flip-flop is the inverse (complement) of the "Master" clock input, the outputs from the "Master" Flip-flop are only "seen" by the "Slave" Flip-flop when the clock input goes "LOW" to logic level "0". Therefore on the "High-to-Low" transition of the clock pulse the locked outputs of the "Master" Flip-flop are fed through to the J-K inputs of the "Slave" Flip-flop making this type of Flip-flop edge or pulse-triggered. Then, the circuit accepts input data when the clock signal is "HIGH", and passes the data to the output on the falling-edge of the clock signal. In other words, the Master-Slave J-K Flip-flop is a "Synchronous" device as it only passes data with the timing of the clock signal.

Figure 1.33 contains circuit diagram of JK Master Flip-Flop along with its timing diagram. If we observe the above figure, J-K flip-flop output will be changing only when clock's falling edge is encountered.

When clock pulse is high slave is isolated while clock pulse is at low (goes to low) master will be isolated. That is, when slave is isolated means external input will not change its current content. That is, any other component which takes feed from this flip-flop will continue to get current value of the flip-flop as input. Similarly, when master is isolated its content will be transferred to slave. Assume a digital system with a set of flip-flops with the outputs of some flip-flops feeding others. At the beginning of a clock pulse, some of the masters change their states (that they may get feed from other flip-flops) but all the flip-flops outputs will be at their previous states. When clock pulse returns to 0, some of the outputs change but they will not be influencing any others masters till next clock cycle. Thus, simultaneously flip-flops states will be changed in the same clock cycle.

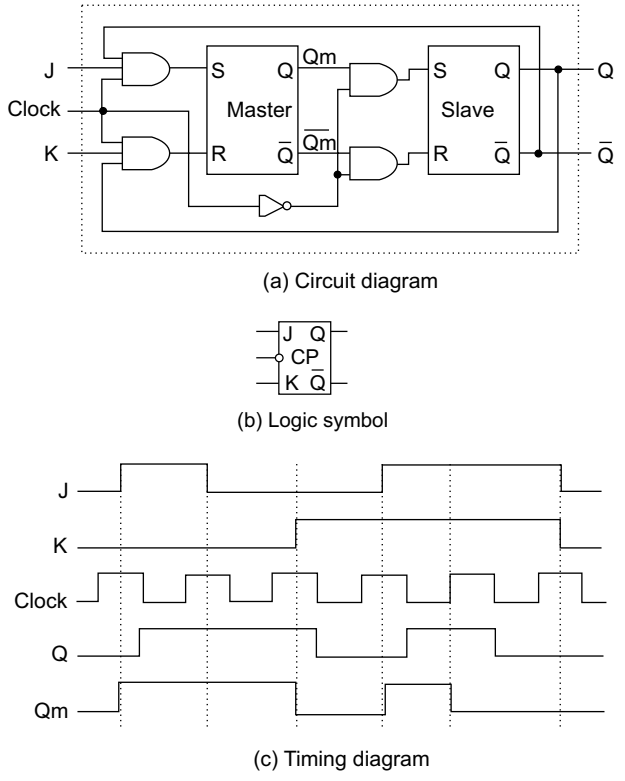


Figure 1.33 JK Master Slave Flip-flop with timing details

1.12.1.4 The T Flip-flop

If the T input is high, the T flip-flop changes state ("toggles") whenever the clock input is strobed. If the T input is low, the flip-flop holds the previous value. The schematic diagram of T flip-flop is given in figure 1.34.

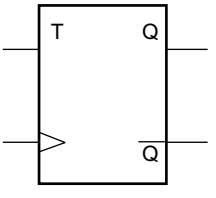


Figure 1.34

Table 1.6 (characteristic table) contains the behavior of T flip-flop.

Table 1.7 Characteristic table of T flip-flop

T	Q_N	Q_{N+1}
0	0	0 (Q_N)
0	1	1 (Q_N)
1	0	1 ($Q_{N'}$)
1	1	0 ($Q_{N'}$)

In fact, T flip-flop can be realised using J-K flip-flop itself. In Fig. 1.35. T line is added to J and K lines of J-K flip-flop. We know that J-K flip-flop state will not change if J and K are at 0. Also, we know that J-K flip-flop state gets toggled when we make J and K lines at 1. This principle is simply used in designing Toggle flip-flop.

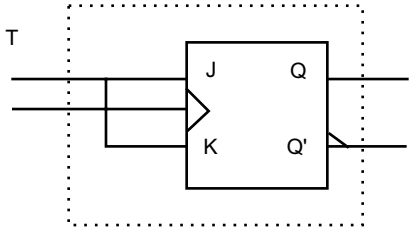


Figure 1.35

When T is held high, the toggle flip-flop divides the clock frequency by two; that is, if clock frequency is 4 MHz, the output frequency obtained from the flip-flop will be 2 MHz. This 'divide by' feature has application in various types of digital counters. A T flip-flop can also be built using a D flip-flop (T input and $Q_{previous}$ is connected to the D input through an XOR gate).

1.12.1.5 The D-type Flip-flop

The D-type flip-flop is just a clocked flip-flop with a single digital input D. Every time a D-type flip-flop is clocked, its output follows whatever the state of D its D line is. A flip-flop may be used to store or 'lock' one bit of information. This locking of information is also known as 'latching', so a flip-flop may be referred to as a single-bit latch.

There now exist many digital IC›s consisting of a set of several flip-flops, whose main function is to latch several bits of data. These IC›s are known as 'latches', and are used to capture data from the data bus of a digital system at precise moments in time. In fact, simple computer-controlled circuits use latches as I/O devices. The flip-flop is also the basic building block of SRAM's.

Positive Edge Triggered D Flip-flop

In a positive edge-triggered D Flip-flop, the output looks at the input only during the instant that the clock changes from low to high. At every raising edge input is set to output (See Figure 1.36).

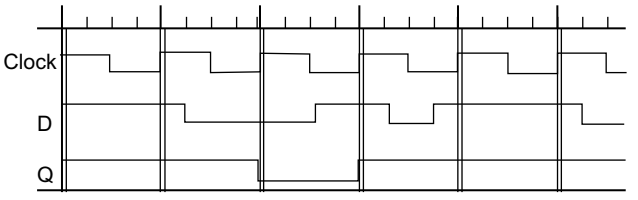
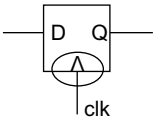


Figure 1.36 Timing diagram of a positive edge triggered D Flip-flop

Following block symbol is used representing positive edge triggered D flip-flop. The "carrot" symbol means edge-triggered.



Negative Edge Triggered D Flip-flop Here, at every falling edge of the clock, input is set to output. The following timing diagram illustrates the same.

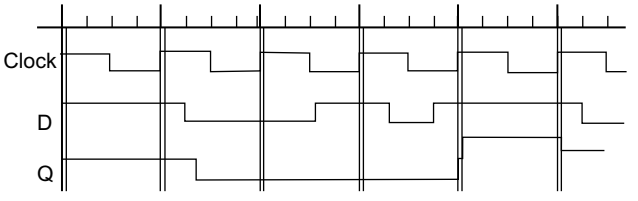
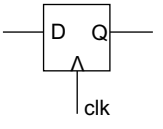


Figure 1.37 Timing diagram of negative edge triggered D Flip-flop

The following block diagram is used for representing negative edge triggered D flip-flop. See the difference between this and previous one.



Triggering on Edge Sometimes we want to enable flip-flop briefly for some time on rising edge. To achieve this we can use a raising edge detect circuit with a NOT and AND gates as shown in Figure 1.38. Here, NOT is used to induce delay. Figure 1.38 also contains the timing diagram of the effect of NOT and AND gates for clock input. We may find that the NOT gate is inducing a delay which is the reason why we are getting a narrow width pulse when a raising edge of the clock arrive. This, we are referring edge and is used to enable the D flip-flop. Output of the D flip-flop follows input when this edge signal arrives.

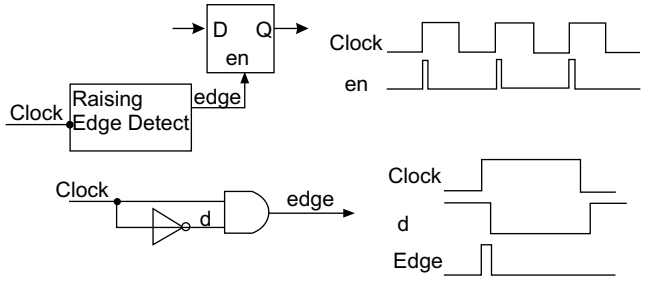


Figure 1.38 Edge Triggered Flip-flop

D-Latch

Another type of latching circuit which doesn't have an illegal state is shown below in Figure 1.39. This is called a D-Latch. A D-Latch is often used as a bi-stable memory circuit with a single input called D. Think of it as a single bit memory.

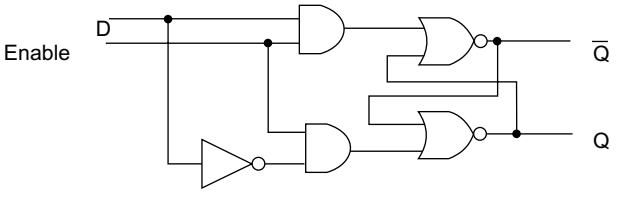
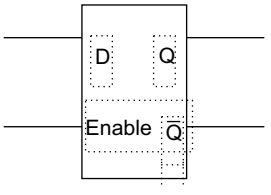


Figure 1.39 D-Latch

The output Q is set to the state of input D whenever the Enable is held to High. When the Enable is held Low, the output states are latched and will not change regardless of the value of input D. This is summarised in the truth table below. Symbol is also given below.

Inputs		Outputs	
Enable	D	Q	$\overline{Q}$
0	0	Latched	
0	1	Latched	
1	0	0	1
1	1	1	0



1.12.1.6 Conversion of Flip-Flops from One Type to Another

Sometimes it just happens that we need a particular type of flip-flop for a specific application, but all we have available is another type. This often happens with an application needing T flip-flops, since these are not generally available in commercial packages. Rather, it is necessary to re-wire an available type to perform as a T device.

Fortunately, this is not hard. we have already seen that a JK flip-flop with its J and K inputs connected to a logic 1 will operate as a T flip-flop.

Converting a D flip-flop to T is quite similar; the Q' output is connected back to the D input.

Converting an RS flip-flop to T flip-flop involves a bit more, as shown in Figure 1.40. However, the simple feedback connections shown will ensure that the S and R inputs will always tell the flip-flop to change state at each clock pulse.

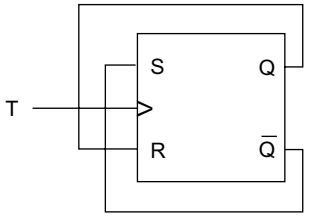


Figure 1.40 T Flip-flop

Another conversion that is required on occasion is to convert an RS flip-flop to D flip-flop. Figure 1.41. contains a solution. This change eliminates the possibility of an illegal input condition, which could otherwise cause spurious results in some applications.

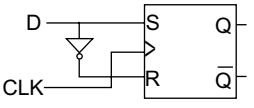


Figure 1.41 D Flip-flop

In this case, we do need to add an inverter to supply the R input signal, as shown in Figure 1.42.

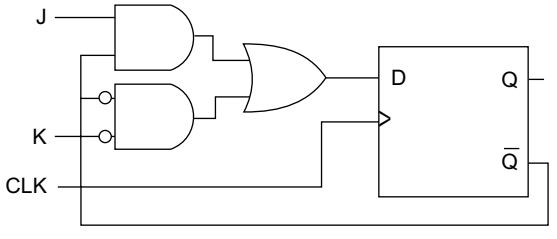
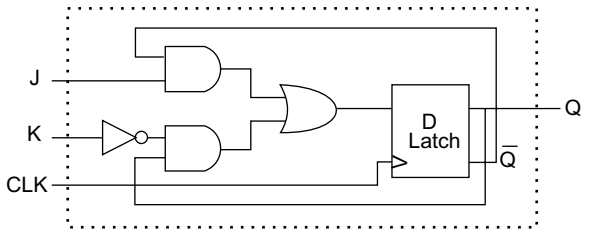


Figure 1.42 A stable D Flip-flop

A much more complicated circuit, shown to the above, is the gating structure needed to convert a D flip-flop to JK flip-flop. This circuit implements the logical truth that D = JQ' + K'Q.

This input circuit is frequently used. CMOS flip-flops are typically constructed as D types because of the nature of their internal operation. This approach eliminates the internal latching effect, or "ones catching," that occurs with the general JK master-slave flip-flop. The J and K input signals must be present at the time the clock signal falls to logic 0, in order to affect the new output state.

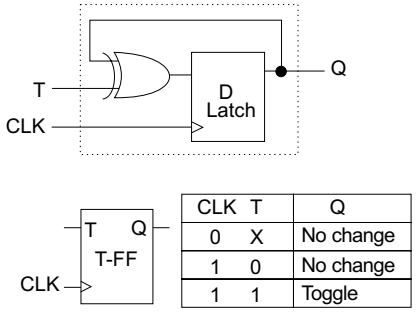
Figure 1.43 shown also another circuit to convert a D flip-flop to JK flip-flop. Readers are advised to see the conceptual difference between this and the above.



CLK	JK	Q
0	XX	No change
1	00	No change
1	01	0
1	10	1
1	11	Toggle

Figure 1.43 JK Flip-flop with its Excitation Table

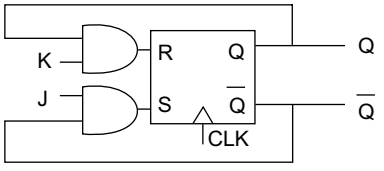
The following circuit behaves similar to T flip-flop. We have given characteristic table in Figure 1.44. which shows its behavior as same as T flip-flop.



CLK	T	Q
0	X	No change
1	0	No change
1	1	Toggle

Figure 1.44 A Circuit which behaves like T Flip-flop

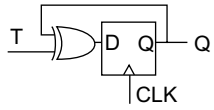
■ **Example** The following circuit is proposed to avoid forbidden state of RS flip-flop. Analyse and illustrate functioning of this circuit.



■ **Answer:** The following truth table illustrates the working of the above RS flip-flop. We may find that racing condition of normal RS flip-flop is avoided here.

J	K	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1 (HOLD)
0	1	0	0
0	1	1	0 (CLEAR)
1	0	0	1
1	0	1	1 (SET)
1	1	0	1
1	1	1	0 (TOGGLE)

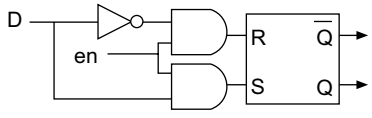
■ **Example** Analyse the following circuit.



■ **Answer:** The following table illustrates the behavior of the above circuit. It behaves similar to T flip-flop.

T	Q	Next State Q'
0	0	0
0	1	1
1	0	1
1	1	0

■ **Example** Explain the behavior of the following circuit.



■ **Answer:** When en (enable) is low, S and R will be 0. Thus, flip-flop will now have any change of its state. If enable is high and D is 0 then R, S values becomes 1 and 0 thus flip-flop gets cleared while D is 1 then R, S values becomes 0 and 1 thus flip-flop gets set. This type of circuit is called as level-sensitive flip-flop.

1.12.1.7 Set and Clear Signals

Practical circuits have setting and clearing flip-flops in an asynchronous manner, i.e., independent of other signals including clock. Clear forces the output to low regardless of the other inputs. Preset forces the output to high regardless of the other inputs. The following circuits (Figure 1.45) shows this behavior. The following first circuit we find flip-flop will be getting preset if preset line is set. That is, if preset is at logic 1, then R and S lines becomes 0 and 1, respectively thus flip-flop will be kept at set state. While the second circuit sets flip-flop state to clear independent of other signals. That is, if clear =1 then R and S inputs of the flip-flop becomes 1 and 0, respectively independent of other signals. Thus, flip-flop will be set in clear or 0 state.

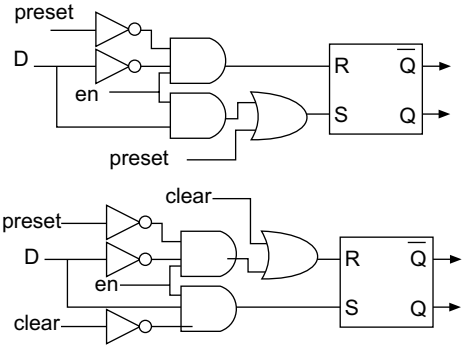


Figure 1.45 Set and Clear lines with Flip-flops

1.12.2 Shift Registers

A register is a semiconductor device that is used for storing several bits of digital data. It basically consists of a set of flip-flops, with each flip-flop representing one bit of the register. Thus, an n-bit register has n flip-flops.

A special type of register, known as the shift register, is used to pass or transfer bits of data from one flip-flop to another. This process of transferring data bits from one flip-flop to the next is known as 'shifting'. Shift registers are useful for transferring data in a serial manner while allowing parallel access to the data.

A shift register is simply a set of flip-flops interconnected in such a way that the input to a flip-flop is the output of the one before it. We know that a D flip-flop will retain value on its D line as its stored bit when a clock pulse arrives. Thus, a flip-flop of this register will be getting the data from a flip-flop that is left to it. Output of right most flip-flop is considered as serial output or DATA OUTPUT as shown in Figure 1.46. Also, left most flip-flop will be taking incoming serial data as its data for each clock cycle. Clocking all the Flip-flops at the same time will cause the bits of data to shift or move to the right in one direction (i.e., toward the last flip-flop). Figure 1.46 shows a simple implementation of a 5-bit shift register using D-type flip-flops.

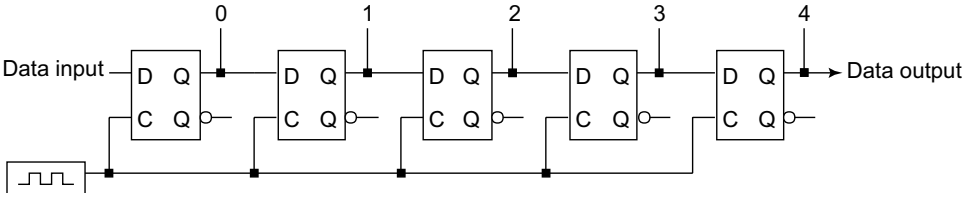


Figure 1.46. A Simple Shift Register Consisting of D-type Flip-flops

Data can be taken out in a parallel or serial fashion, as shown in Fig. 1.46. Tapping from the points 0,1,2,3, and 4 at the same time, we get parallel output. That is, we will be having the contents of each flip-flop on these lines. If we take them at the same time, we will be getting parallel data. Thus, the shift register in Figure 1.46 can be used for both serial to parallel transfer in addition to using the same as serial-in and serial-out register.

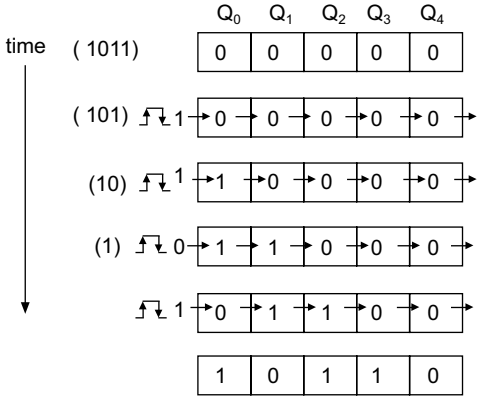


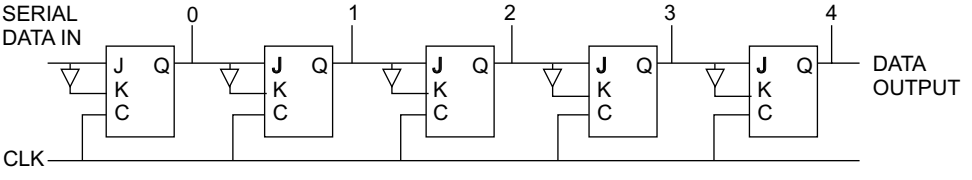
Figure 1.47 Operation of a serial register

Figure 1.47 illustrates the working of this circuit. Initially, we assume that all the flip-flops are at their cleared state. That is, 00000 is available in the register. If we assume serial input is 1011 then in the first clock cycle first flip-flop will be getting loaded with 1. During the second clock cycle the second flip-flop takes this 1 while first flip-flop takes 0. Like this, serial data will getting loaded and shifted one bit at a time into the register.

■ **Example** How can you modify the above circuit such that in place of D flip-flops we can replace T flip-flops.

■ **Answer:** We simply take output of left side flip-flop and make itself and its complement as the J and K lines of current flip-flop as shown below.

Under its basic operation of the above shift register, the data bit of the last flip-flop is lost once it is clocked out. In some applications there is a need to bring this back to the first flip-flop, in which case the data will just be circulated within the shift register. A shift register connected this way is known as an end-around-carry shift register, or simply 'ring rigster'.



1.12.2.1 Bi-Directional Shift Register

A more complicated version of a shift register is one that allows shifting in both directions, left or right. It is aptly and quite descriptively referred to as the Shift-Right Shift-Left Register. To accomplish this, a 'Mode' control line is added to the circuit. The state of this 'Mode' input determines whether the shift direction would be right or left. The following circuit in Figure 1.48 contains shift left and shift right register.

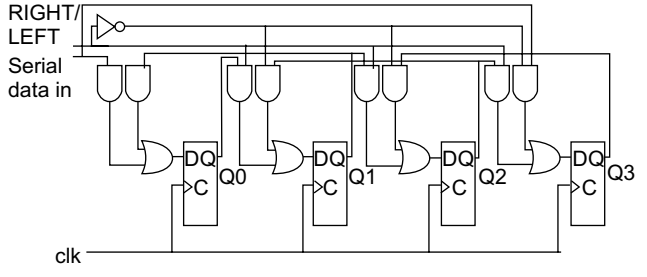


Figure 1.48 Bi-directional Shift Register using D Flip-flops

In the above circuit, we have line name RIGHT/LEFT. If we want right shift, this line should go high. Similarly, if we want left shift this line should go low. If we observe the circuit, we find with each D flip-flop an OR gate is connected to flip-flops D line. This OR gate is getting feed from two AND gates. Previous flip-flop output is connected to first AND gate while next flip-flops output is connected to the other AND gate. Also, RIGHT/LEFT line is connected to first AND gate while its complement is connected to the other AND gate. Thus, when RIGHT/LEFT line high previous flip-flop value is stored in the current while the same goes low next flip-flop value is retained. However, in the case of left most and right most flip-flops, data on the line serial data in is retained.

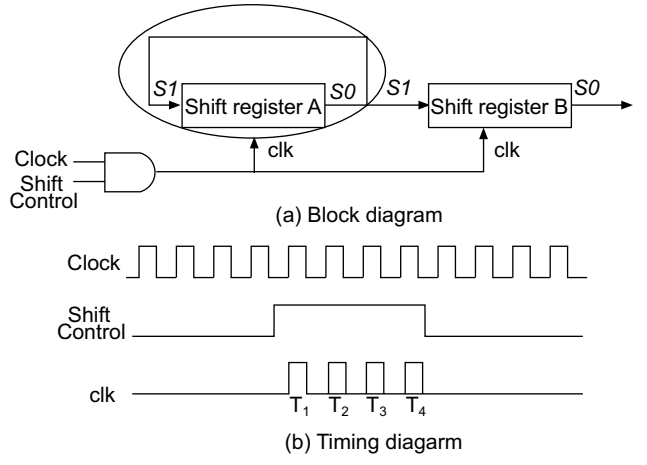


Figure 1.49 An application of shift register

The above diagram in Figure 1.49 illustrates one use of the shift register, i.e., to transfer content of a register to another in serial. The SO is again connected to SI in first register (see encircled portion) such that first register will retain its original value after transfer.

Figure 1.50 contains another use of shift register. While adding two numbers, we assume both the numbers are in two shift registers. Each time one bit of them are made available to Full Adder circuit which adds them and generates sum and carry bits. Sum bits are fed back to first shift register as a serial in such that final sum is available in the first shift register. Also, carry is stored in a D flip-flop and used while adding next bits. We have discussed same problem in state machines also.

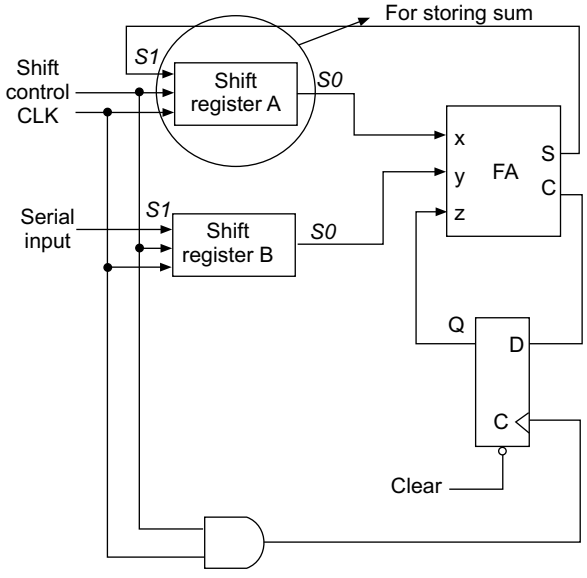


Figure 1.50 Application of shift register in adding two numbers

■ **Example** Does the circuit given in Figure 1.50 can be added to n numbers of size 8-bits. Do assume that shift register are of 8-bits size. If possible, explain how? Do inform any assumptions you make.

■ **Answer:** Assuming that flip-flops in both the registers are having clear line, we first clear all of them. Thus, both the registers contain 0s. Now, we present first number (A) via serial in line of second register while clearing carry flip-flop also. Then, we present next number and vice versa. After sending the second number (B), first register contains first number as 0+A=A. While sending third number via serial in line of second shift register, first two numbers will be added and result will be in first result. Like that, first shift register contains sum of n numbers.

1.12.2.2 Parallel Loading into Registers

In the above example, we have loaded serial data into a register. A second method of loading a register is by shifting in all the bits in parallel at the same time (for example from a decoder). An example of a 3-bit parallel shift register is

shown below in figure 1.51. Here, we assume X0,X1, X2 etc., are parallel lines. If SHIFT=1 then data on the X0, X1, X2 lines will be retained in the respective flip-flops.

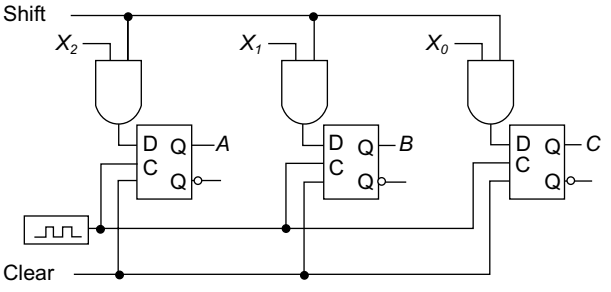


Figure 1.51 Parallel Shift Register with D flip-flops.

That is, when the shift line goes high, the outputs of the AND gates take on the values of X. On the next clock, this information is shifted into the register. To reset the register, the CLEAR line (an asynchronous signal) goes high so that all the Q's will turn low on the next clock pulse.

We can also achieve the same using JK flip-flops as shown below in Figure 1.52. We assume first RESET line will made in a clock cycle before loading the data into register. Thus, all the flip flops will be in this clear or 0 state. When we want to load the data we make SHIFT line to go high such that X values will be available as the output on the AND gates outputs in first cycle. During the next cycle, the same will be stored in the flip-flops. Do remember X values can be 0 or 1. Thus, if X value is 0 then J and K lines of any flip-flop will be 0. Thus, it retains its value, i.e., 0. Do remember, the register will be RESET before loading parallel data. Similarly, if X value is 1 then J and K lines of any flip-flop are 1 and 0 respectively. Thus, flip-flops will be in their set state or 1 (i.e., X) is said to be stored. Thus, parallel load is possible using this circuit.

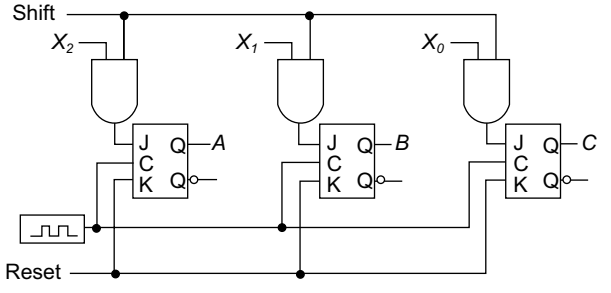
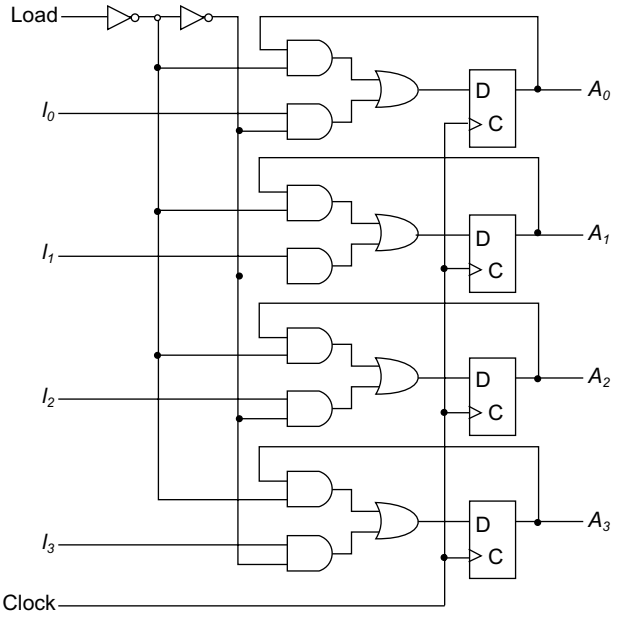


Figure 1.52 Parallel Load with JK Flip-flops

Thus this configuration requires two clock pulses; one to reset and one to load.

■ **Example** Analyse the following circuit which is proposed for parallel load operation into a register with D flip-flops. Explain what happens when load=1 and load=0. Compare the same with parallel load circuit in Figure 1.51.



■ **Answer:** Load = 1 ; the I inputs (I0 to I3) are transferred into the register. When load = 1, data on the I0 to I3 are available on the D lines of the flip-flops which will be stored in the D flip-flops.

Load = 0 ; maintain the content of the register. Because, the D flip-flop does not have a "no change" state like other flip-flops. That is, if load=0, then D lines of the flip-flops will have their contents itself. Thus, next value of the D flip-flops becomes current value itself.

Compared to the circuit in Figure 1.51, this does not have clear facility. Of course, adding it is not so difficult.

1.12.2.3 Parallel Load and to Serial Out Register

The following circuit in Figure 1.53 gives ability to load data parallel into a register and then outputs the same in serial manner. When LOAD/SHIFT is made 1, then the data on

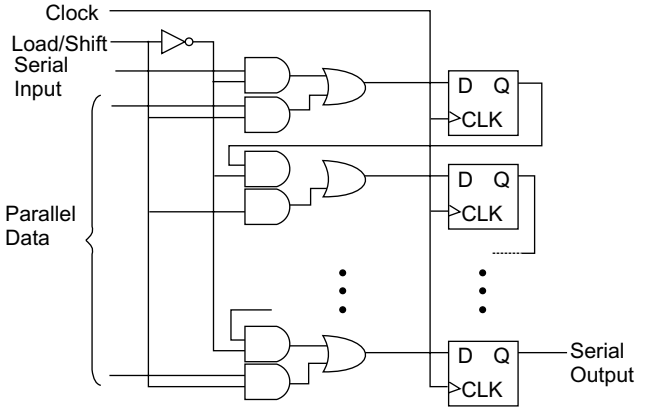


Figure 1.53 Parallel Data load and serial output.

parallel lines will be transferred to register. After that, for every one clock cycle the register content will shifted down.

Thus, the loaded data will be available in serial fashion on the line SERIAL OUTPUT. Here, also ith flip-flop will be getting feed from flip-flop above it. That is, value available in a flip-flop will be shifted to a flip-flop below it.

1.12.2.4 Parallel Load and Parallel Out Register

By simply taking output from each of the flip-flops in Figure 1.50, we can get parallel output from a register at any time.

1.12.2.5 Register to Register Parallel Transfer

Sometimes we need to transfer contents of a register to another register in parallel. This can be achieved between a register with D flip-flops to another register with JK flip-flops with the following type of circuit in Figure 1.54. If we see the circuit, we find output of a D flip-flop is connected to J and K lines of a JK flip-flop. If D flip-flop contains 0, then J and K lines becomes 0 and 1 thus the JK flip-flop retains 0. If D flip-flop contains 1, then J and K lines becomes 1 and 0 thus JK flip-flop will be retaining 1. Thus, the content of the register with D flip-flops will be transferred to the register with JK flip-flops.

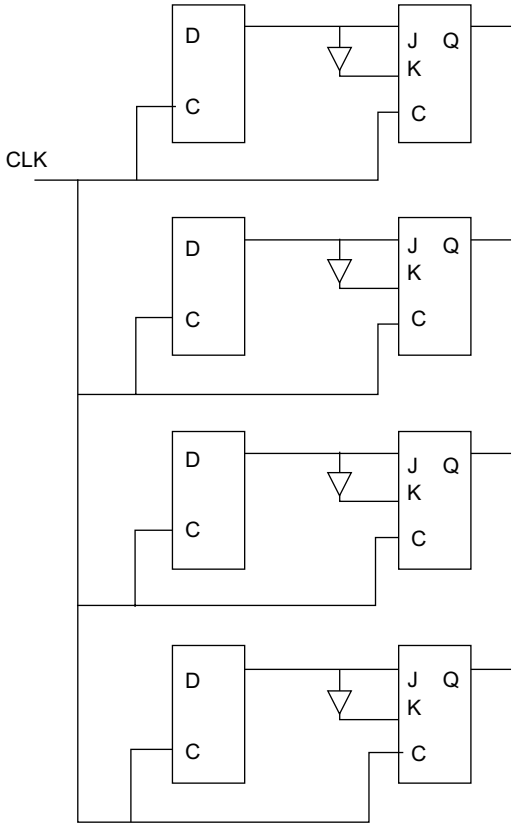


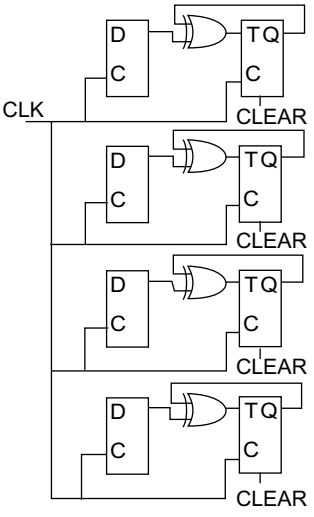
Figure 1.54 Register to Register Parallel Transfer

■ **Example** Is it possible to replace JK flip-flops with SR type without any other change?

■ **Answer:** Yes

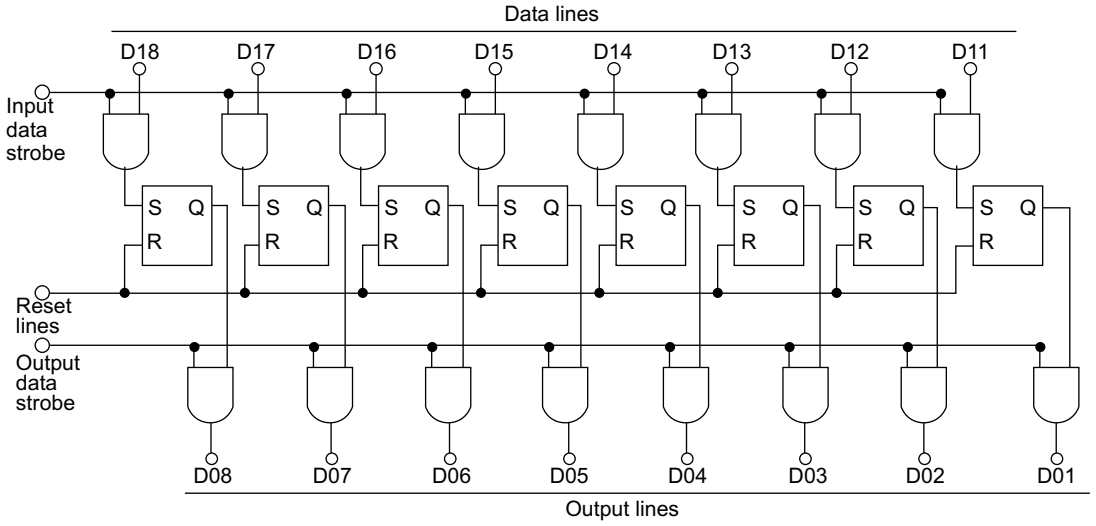
■ **Example** Is it possible to replace JK flip-flops with T type without any other change?

■ **Answer:** No. We need to change the circuit as shown below. First, we assume that the destination register with T flip-flops will be first cleared before parallel transfer. After clearing, each flip-flop of the destination register will be having 0. Now, T flip-flops output and D flip-flop output will be exclusive ORed. Thus, D flip value is 0, then XOR output will be 0. So, T flip-flop value will be 0 itself. If D flip-flop value is 1, then XOR output will be 1, thus T flip-flop valued will be toggled. Thus, T flip-flop value will become 1 from the clear state.



■ **Example** Explain the functionality of the following circuit with SR flip-flops.

■ **Answer:** This circuit can be used to load data on a parallel lines to a register with SR flip-flops. We assume that initially all the flip-flops are in cleared state. If we assume Input data strobe is 1 then data available on data lines will be available on S lines of flip-flops. If we assume Reset line 0, then the flip-flop will be in its clear state if S=0 else it will be moved to set state. That is, if a data line contains 1 then its respective AND gate output is 1, i.e., S=1 in its flip-flop. Thus, it will be set. Like this, data on the data lines will be transferred to register. Similarly, we have another part of the circuit with which data can be sent on output lines. If output data strobe becomes 1, data in the flip-flops will be available in the output lines.

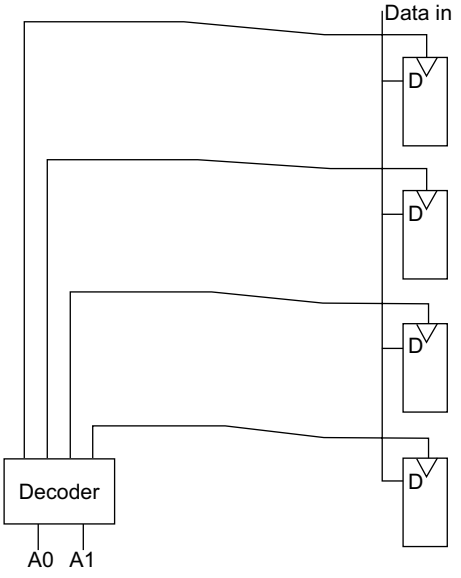


■ **Example** Writing something into one of the flip-flop of a register using decoder.

Decoder is a circuit used to select one line out of many. For example, in the following we are using 2x4 decoder. That is, one in one of the output lines we will have 1 based on two control inputs. The same can be represented as:

Input	Output
0 0	1 0 0 0
0 1	0 1 0 0
1 0	0 0 1 0
1 1	0 0 0 1

We want to store whatever available on **Data in** line in one of the four D flip flops. We know that D flip flop loads the available data on its input line when it receives CLK on its Chip select line. As output lines of decoder are connected to flip flop's CLK as shown in the following figure, thus only one flip flop will be selected out of the four. Thus, that flip flop loads the data available in **data in** line.



1.12.2.6 Universal Shift Register

If the register has both shifts and parallel load capabilities, it is referred to as a universal shift register. It will contain

- A clear control to clear the register to 0.
- A clock input to synchronise the operations.
- A shift-right control to enable the shift right operation and the serial input and output lines associated with the shift right.
- A shift-left control to enable the shift left operation and the serial input and output lines associated with the shift left.
- A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
- n parallel output lines
- A control state that leaves the information in the register unchanged in the presence of the clock.

This can be realised in many ways. However, designing using MUXes is somewhat easy. The following circuit uses 4x1 MUXes as shown in figure 1.55. Functioning of this circuit can be summarised as:

Control Inputs S1, S0	Action
00	No change in the register content.
01	Shift right: The serial input for shift-right is transferred to the A3.
10	Shift left: The serial input for shift-left is transferred to the A0.
11	Parallel load to the register.

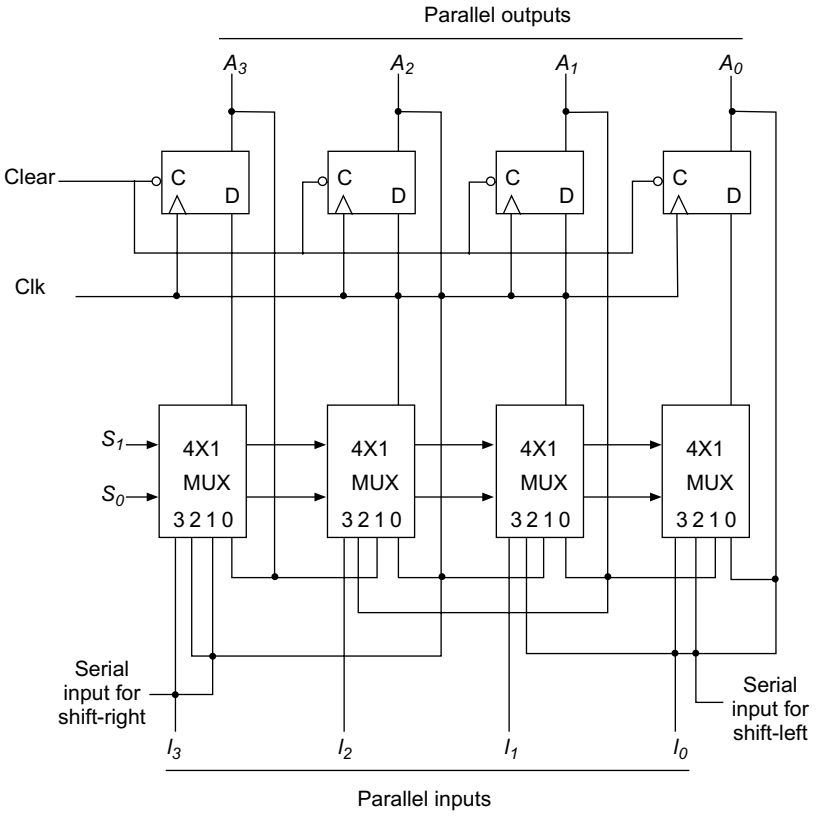
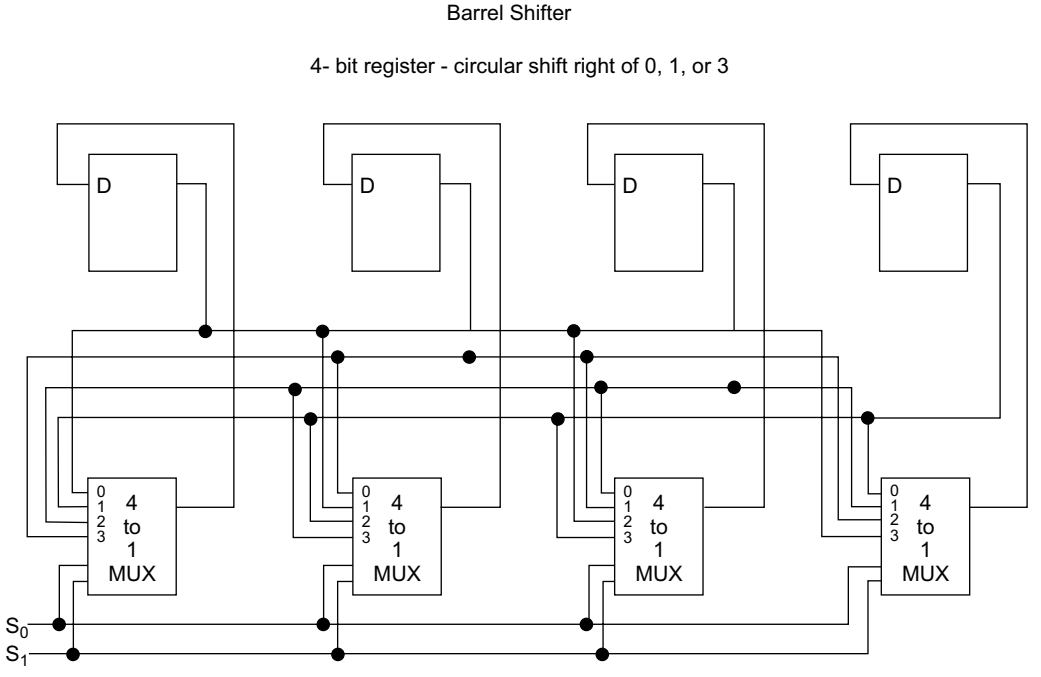


Figure 1.55 Universal Shift register

The MUXes used above taking current flip-flop output, next flip-flop output, previous flip-flop output and parallel input on its input lines. Based on the control inputs S1, S0 one of them are loaded into flip-flops.

■ **Example** The following circuit is proposed using D flip-flops and MUXes for circular shifting. Explore the functionality of the same. This is called as barrel shifter.



Note that each flip-flop is controlled by a multiplexer, which is used to select the input sent to the flip-flop. The multiplexer's function is to route the value selected according to its address lines to the flip-flop's input. To set up the circuit as a shift register, the 4 multiplexer input data lines are simply hooked up to the flip-flop outputs so that each address matches a shift value, with address 0 matching a shift of 0, address 1 matching a shift of 1, and so forth. Thus, the amount of the shift is entered through the address lines (S0, S1). The circuit can be reconfigured for different shift patterns by simply hooking up the multiplexer input data lines to the flip-flop outputs (or other data values) in different ways.

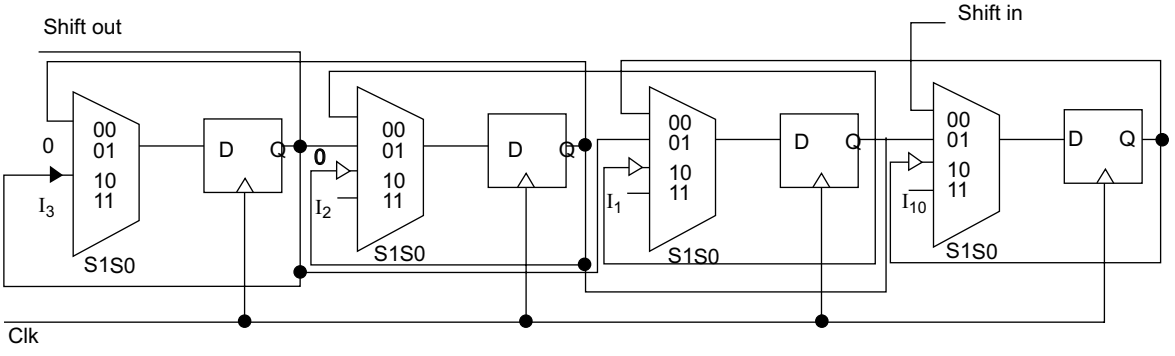
Assuming D3 to D0 are the D flip-flops and MUXes as M3 to M0 (all considered from left to right). We see output lines of all D flip-flops are inputs to all multiplexers. For example, multiplexer M3, outputs of flip-flops D3, D0, D1 and D2 are connected. Similarly for M2 MUX outputs of D2, D3, D0 and D1, for M1 MUX outputs of D1, D2, D3 and D0 and for M0 MUX outputs of D0, D1, D2, and D3 are connected. Output of a MUX is connected to D line of the corresponding flip-flop.

For example, if we consider a number 1011 (D3 to D0) is available in flips flops then M3, M2, M1, M0 MUXes input lines contains 11**1**0, 01**1**1, 10**1**1, and 11**0**1 respectively. Let, we want to shift the number by two bits. Thus, we set S0S1 as 10, which make MUXes to select their line 2 input and send to their output line which is really saved in the flip-flops. That is, 1110 (D3 to D0) is saved in flip-flops which is two bits right shifted (circular) version of 1011.

■ **Example** Design a 4-bit register that can shift left by one bit, shift right by two bits, invert its contents, and load a new value (respectively, ctrl = 00, 01, 10, 11). Provide a **shift_in** input and a **shift_out** output for use when left shifting. When right shifting, the register should shift in zeros and the **shift out** should be ignored. Use D flip-flops.

■ **Answer:** As we want four different types of operations on the content of the register, we propose to use output of a 4x2 MUX with each of the D flip-flop as shown in the following figure. We consider this is more like a barrel shifter. Select lines S1S0 decides the required function of the circuit. We assume the following:

1. If the control lines of the MUX is 00, system should shift the content of register left by one bit. Thus, ith flip flop output is fed to i-1th flip-flops MUX as shown in the figure.
2. If the control lines of the MUX is 01, system should shift the contents of the register right by two bits. From the problem statement, we can conclude that most significant two flip-flops should become 0 while least significant flip flops to get shifted values. Thus, we have taken MUX inputs of two most significant flip flops as 0s as shown in the figure while 3rd flip-flop output is connected to 1st flip-flops MUX and 2nd flip-flop output too 0th flip-flops MUX.
3. If the control lines of the MUXes are 10, we want complement of the register. Thus, output of a flip flop is complemented and given as third input to its MUX.
4. If the control lines of the MUXes are 11, we want to load I3I2I1I0 to the register. Simply, we have fed these lines as 4th inputs to each of the MUXes as shown in the following figure.



1.12.3 Counters

In digital logic and computing, a **counter** is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, often in relationship to a clock signal.

Counters are implemented using register-type circuits. We have two prominent types of counter designs known as:

- Asynchronous (ripple) counters
- Synchronous counters

1.12.3.1 Asynchronous counters

The counter's output is indexed by LSB (least significant bit) every time the counter is clocked. That is, add each time one to the current number. For example, if we assume 000

is the starting number, the sequence we expect from a 3-bit counter is given below.

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
0	0	0

If we observes the above, we can conclude that for the low order bit:

Just swap between 0 and 1 – i.e., it toggles

For higher order bits:

the n'th bit toggles when the n-1'th bit changes from 1 to 0.

For example, if we want to design a 2-bit binary counter, we can use the above facts. For example, the following circuit (Figure 1.56) can serve as a 2-bit counter which uses two JK flip-flops. Both the flip-flops J and K lines are kept at high state such that whenever a clock pulse arises, JK flip-flop state complements. Also, first flip-flop (Q0) outputs complement is the clock for next flip-flop (Q1). If we assume that both the flip-flops are in their clear state and when 1st clock pulse arises Q0 gets complemented, i.e., it value becomes 1. However, second flip-flop (Q1) will still be in its 0 state. When next clock pulse arrive to Q0 its state gets complimented again. That is, Q0 becomes 0 and while Q1 changes to 1. When third clock pulse arrives Q0 becomes 1 while Q1 will be at 1 itself. Like this, we may get 00, 10, 01 and 11. Do remember, Q0 is the LSB.

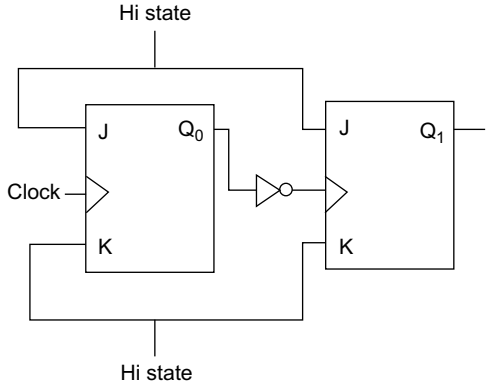


Figure 1.56 A simple 2-bit counter

Figure 1.57 contains a 4-bit binary up counter using JK flip-flops.

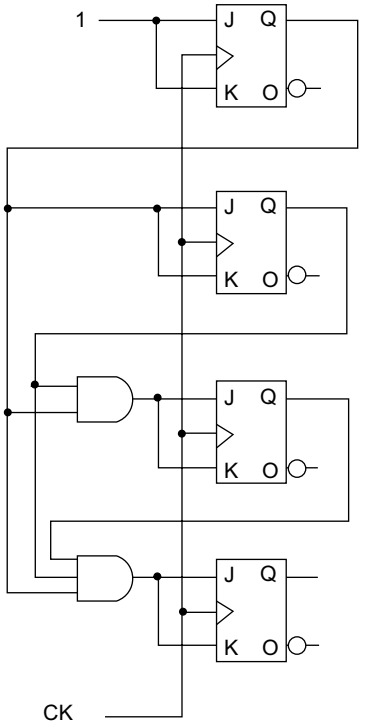


Figure 1.57 Design of Binary Down counter Using JK Flip-flops

Consider the following state table of a 3-bit binary down counter.

$Q_2Q_1Q_0$
000
111
110
101
100
011
010
001

We find the following features in the above table:

1. Q0 changes alternatively.
2. Q1 toggles whenever Q0 is 0.
3. Q2 toggles whenever Q1=Q=0.

We can generalise that Qi toggles if Qj=0 for all j <i, j>=0

We know that in order to toggle JK flip-flop contents, both J and K lines should be 1. Thus, we can join complement output of a flip-flop to J and K lines of next JK flip-flop. The following shows circuit for 4-bit count down counter using the above rules. As first flip-flop is supposed toggle each time, its J and K lines are kept at high (1) always.

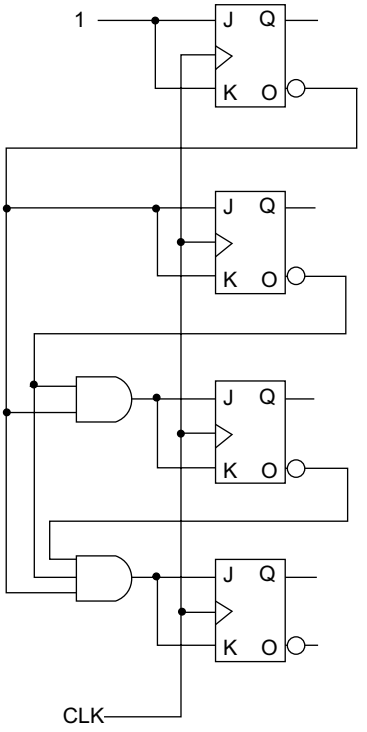


Figure 1.58 A 4-bit countdown counter using JK flip-flops.

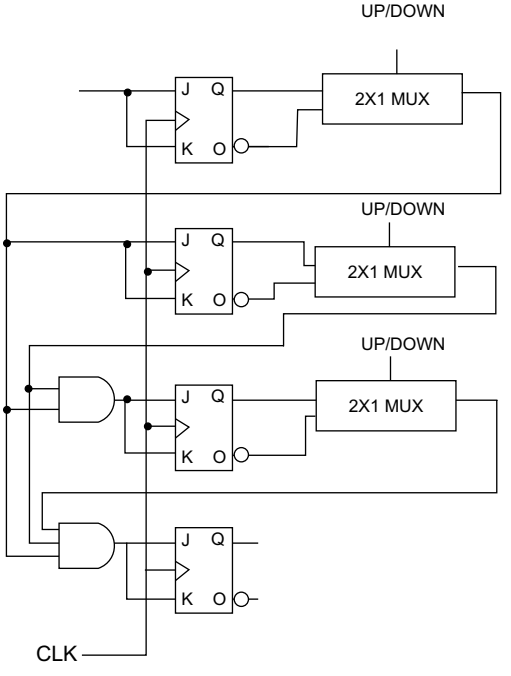


Figure 1.59 A 4-bit up/down counter using 2X1 MUXes

The following contains both binary up and down counter. Here, we are using a 2x1 MUX with a line UP/DOWN as selection line. If it is 0, counter works like a up counter else it works like a down counter.

■ **Example** How to design a binary up down counter using T flip-flops.

■ **Answer:** Simply replace JK flip-flops with T and make the J line as T and remove K line connections in the above figure.

1.12.3.1.1 Simple Binary Counter Using D and T Flip-flops

The simplest counter circuit is a single D-type flip-flop (or T type), with its D (data) input fed from its own inverted output. This circuit can store one bit, and hence can count from zero to one before it overflows (starts over from 0). This counter will increment once for every clock cycle and takes two clock cycles to overflow, so every cycle it will alternate between a transition from 0 to 1 and a transition from 1 to 0. Notice that this creates a new clock with a 50% duty cycle at exactly half the frequency of the input clock. If this output is then used as the clock signal for a similarly arranged D flip-flop (remembering to invert the output to the input), you will get another 1 bit counter that counts half as fast. Putting them together yields a two bit counter given in the following table:

Cycle	Q1	Q0	(Q1:Q0)dec
0	0	0	0
1	0	1	1
2	1	0	2
3	1	1	3
4	0	0	0

We can continue to add additional flip-flops, always inverting the output to its own input, and using the output from

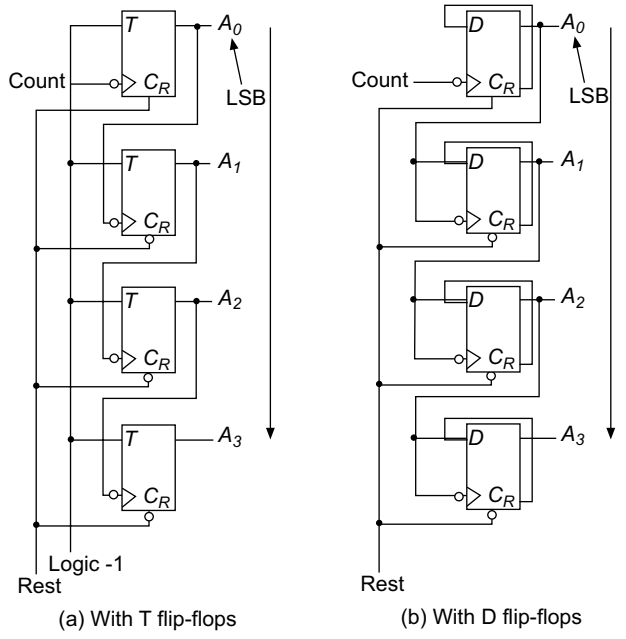


Figure 1.60 Simple binary counters using T and D flip-flops.

the previous flip-flop as the clock signal. The result is called a ripple counter, which can count to 2^n-1 where n is the

number of bits (flip-flop stages) in the counter. Figure 1.60 contains 4-bit ripple counters using D flip-flops and T flip-flops.

In the case of ripple counter with T flip-flops, both the T and clock inputs are kept at logic 1 for LSB flip-flop while other flip-flops T line is logic 1 while clock line is the output of previous flip-flop as shown in Figure 1.60 (a). Figure 1.61 displays the timing details of the ripple counter. We find that output signal at A0 will be half of the original clock (usually called divide by 2), at A1 output signal frequency will one fourth of the clock (divide by 4), at A2 output signal frequency will be one eighth of clock rate, and vice versa. Thus, we can have frequency dividers at various fractions of original clock rate using this circuit. We shall have more discussion about this in our next section.

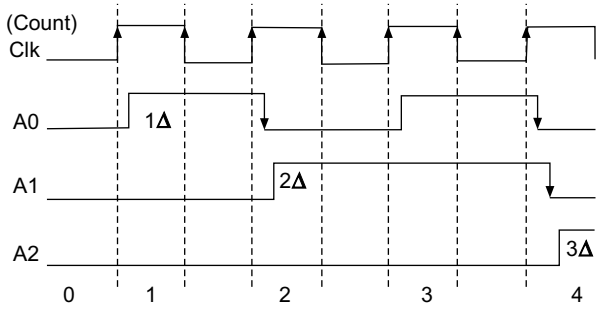


Figure 1.61 Timing information of a counter with either T or D flip-flops

Ripple counters suffer from unstable outputs as the overflows "ripple" from stage to stage, but they do find frequent application as dividers for clock signals, where the instantaneous count is unimportant, but the division ratio overall is. (To clarify this, a 1-bit counter is exactly equivalent to a divide by two circuit – the output frequency is exactly half that of the input when fed with a regular train of clock pulses).

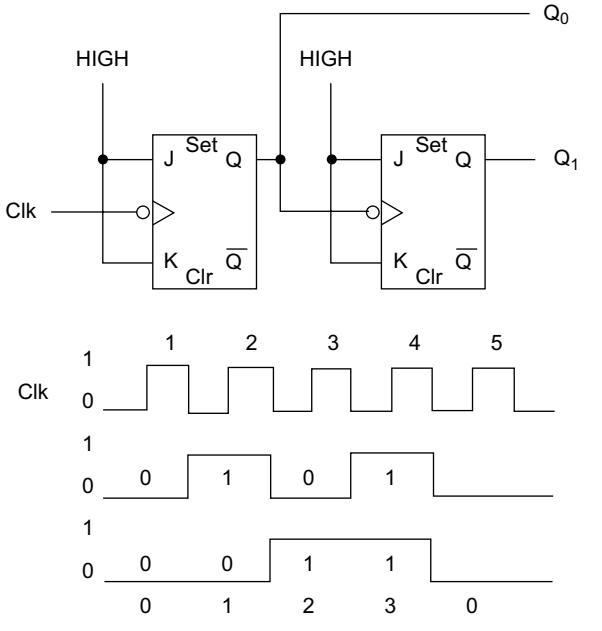
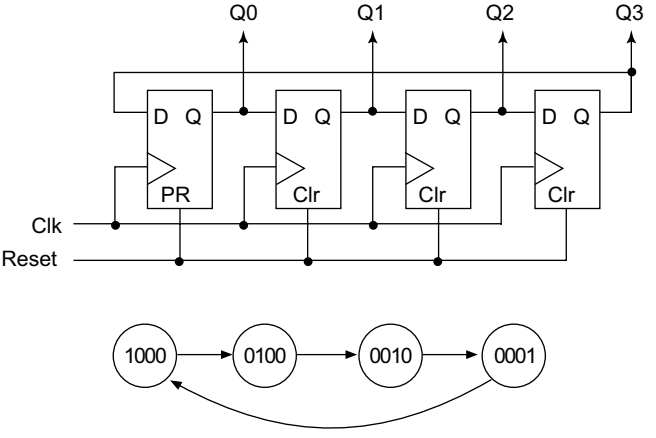


Figure 1.62 A simple counter circuit and timing diagram using JK flip-flops

■ **Example** Design a ring counter using D flip-flops.



■ **Answer:** In order to have a ring counter, we have to make output of MSB flip-flop as D line for LSB D flip-flop as shown above. We assume LSB flip-flop to be equipped with a special line PR (preset) while others to be having CLR line. Initially, when we make RESET=1, then except LSB flip-flop, remaining all will get cleared. That is, 1000 is assumed to be stored in the register. Now, for every clock cycle, how register content changes is shown in the above figure.

1.12.3.1.2 Asynchronous Counter created from JK Flip-flops

A two-bit asynchronous counter is shown in Figure 1.63. The external clock is connected to the clock input of the first flip-flop (FF0) only. So, FF0 changes state at the falling edge of each clock pulse, but FF1 changes only when triggered by the falling edge of the Q output of FF0. Because of the inherent propagation delay through a flip-flop, the transition of the input clock pulse and a transition of the Q output of FF0 can never occur at exactly the same time. Therefore, the flip-flops cannot be triggered simultaneously, producing an asynchronous operation.

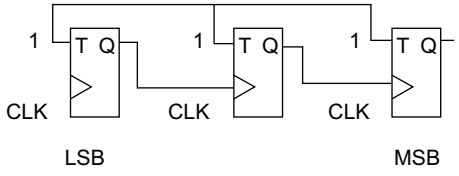
Note that for simplicity, the transitions of Q0, Q1 and CLK in the timing diagram showns simultaneous, even though this is an asynchronous counter. Actually, there is some small delay between the CLK, Q0 and Q1 transitions.

Usually, all the CLEAR inputs are connected together, so that a single pulse can clear all the flip-flops before counting starts. The clock pulse fed into FF0 is rippled through the other counters after propagation delays, like a ripple on water, hence the name Ripple Counter.

The 2-bit ripple counter circuit above has four different states, each one corresponding to a count value. Similarly, a counter with *n* flip-flops can have 2 *to the power n* states. The number of states in a counter is known as its mod (modulo) number. Thus a 2-bit counter is a mod-4 counter.

A mod-n counter may also described as a divide-by-n counter. This is because the most significant flip-flop (the furthest flip-flop from the original clock pulse) produces one pulse for every *n* pulses at the clock input of the least significant flip-flop (the one triggers by the clock pulse). Thus, the above counter is an example of a divide-by-4 counter.

■ **Example** The following circuit is proposed for binary count down counter using T flip-flop. Does it serve the purpose? What is the problem? How to correct it?



■ **Answer:** It will not give expected sequence. To get the correct results, connect a flip-flops complement output (Q') to next flip-flops clock line.

■ **Example** What is the term for the number of counts in one counter cycle?

■ **Answer:** Modulus of the counter

■ **Example** How is the modulus determined?

■ **Answer:** Based on number of flip-flops in the counter. If number of flip-flops are N then modulus can be given as 2^N. However, to be specifically modulus can be found by counting number of possible stable states of the counter.

1.12.3.1.3 Asynchronous Decade (decimal 10) Counter

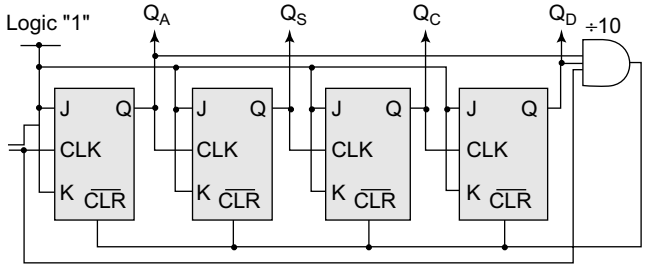


Figure 1.63 Mod 10 counter using JK flip-flops

This type of asynchronous counter counts upwards on each leading edge of the input clock signal starting from "0000" until it reaches an output "1010" (decimal 10). Both outputs QB and QD are now equal to logic "1" and the output from the NAND gate changes state from logic "1" to a logic "0" level and whose output is also connected to the CLEAR (CLR) inputs of all the J-K Flip-flops. This causes all of the Q outputs to be reset back to binary "0000" on the count of 10. Once QB and QD are both equal to logic "0" the output of the NAND gate returns back to a logic level "1" and the counter restarts again from "0000". We now have a decade or **Modulo-10** counter.

Clock Count	Output bit Pattern				Decimal Value
	QD	QC	QB	QA	
1	0	0	0	0	0
2	0	0	0	1	1
3	0	0	1	0	2
4	0	0	1	1	3
5	0	1	0	0	4
6	0	1	0	1	5
7	0	1	1	0	6
8	0	1	1	1	7
9	1	0	0	0	8
10	1	0	0	1	9
11	Counter Resets its Outputs back to Zero				

Using the same idea of truncating counter output sequences, the above circuit could easily be adapted to other counting cycles be simply changing the connections to the AND gate. For example, a scale-of-twelve (modulo-12) can easily be made by simply taking the inputs to the AND gate from the outputs at "QC" and "QD", noting that the binary equivalent of 12 is "1100" and that output "QA" is the least significant bit (LSB).

Standard IC asynchronous counters are available are the TTL 74LS90 programmable ripple counter/divider which can be configured as a divide-by-2, divide-by-5 or any combination of both. The 74LS390 is a very flexible dual decade driver IC with a large number of "divide-by" combinations available ranging form 2, 4, 5, 10, 20, 25, 50, and 100.

1.12.3.1.4 Modulus 16 counter

A 4-bit counter, which has 16 unique states that it can count through, is also called a modulo-16 counter, or mod-16 counter. By definition, a modulo-k or base-k counter is one that returns to its initial state after k cycles of the input waveform. A counter that has N flip-flops is a modulo 2^N counter.

A simple implementation of a 4-bit counter is shown in Figure 1.64, which consists of 4 stages of cascaded J-K flip-flops. This is a binary counter, since the output is in binary system format i.e., only two digits are used to represent the count i.e., '1' and '0'. With only 4 bits, it can only count up to '1111', or decimal number 15.

As one can see from Figure 1.64, the J and K inputs of all the flip-flops are tied to '1', so that they will toggle between states every time they are clocked. Also, the output of each flip-flop in the counter is used to clock the next flip-flop. As a result, the succeeding flip-flop toggles between '1' and '0' at only half the frequency as the flip-flop before it.

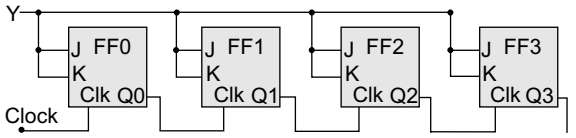


Figure 1.64 A Simple 4-bit Ripple Counter Consisting of J-K Flip-flops

Thus, in Figure 1.64 the last flip-flop will only toggle after the first flip-flop has already toggled 8 times. This type of binary counter is known as a 'serial', 'ripple', or 'asynchronous' counter. The name 'asynchronous' comes from the fact that this counter's flip-flops are not being clocked at the same time.

An asynchronous counter has a serious drawback. Whatever we have explained above is an ideal case unfortunately the output of a flip-flop, like any gate, is slightly delayed with respect to a changing input.

So, the toggling of Q1 will be delayed by a few nanoseconds.

Q2 will only toggle on a change in Q1 so it will be delayed by twice the amount.

This effect will keep on compounding.

These glitches can have quite a disturbing effect.

Consider the case when the counter is at 0111

At the next pulse we would expect the counter to show 1000

However because of the delays the following patterns will all occur

0110	Q0 changes but Q1 has not caught up yet
0100	Q1 has now changed but Q2 is still to flip over
0000	Q2 now has reacted but Q3 is still to change
1000	Eventually we get the right answer

The major problem with the counters shown is that the individual flip-flops do not change state at the same time. Rather, each flip-flop is used to trigger the next one in the series. Thus, in switching from all 1s (count = 15) to all 0s (count wraps back to 0), we don't see a smooth transition. Instead, output A falls first, changing the apparent count to 14. This triggers output B to fall, changing the apparent count to 12. This in turn triggers output C, which leaves a count of 8 while triggering output D to fall. This last action finally leaves us with the correct output count of zero. We say that the change of state "ripples" through the counter from one flip-flop to the next. Therefore, this circuit is known as a "ripple counter."

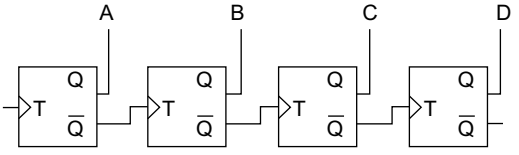
This causes no problem if the output is only to be read by human eyes; the ripple effect is too fast for us to see it. However, if the count is to be used as a selector by other digital circuits (such as a multiplexer or de-multiplexer), the ripple effect can easily allow signals to get mixed together in an undesirable fashion. To prevent this, we need to devise a method of causing all of the flip-flops to change state at the same moment. That would be known as a "synchronous counter" because the flip-flops would be synchronized to operate in unison.

Thus, its speed is limited by the cumulative propagation times of the cascaded flip-flops. A counter that has N flip-flops, each of which has a propagation time t, must therefore wait for a duration equal to N x t before it can undergo another transition clocking.

A better counter, therefore, is one whose flip-flops are clocked at the same time. Such a counter is known as a synchronous counter.

■ **Example** The following example demonstrates the 4-bit binary countdown counter. We will still use edge-triggered master-slave flip-flops The output of each flip-flop changes state on the falling edge (1-to-0 transition) of the T input. However, note that in this case each T input is triggered by the Q' output of the prior flip-flop, rather than by the Q output. As a result, each flip-flop will change state when the prior one changes from 0 to 1 at its Q output, rather than changing from 1 to 0. Because of this, the first pulse will cause the counter to change state from 0000 to 1111. Of course, one may argue that when we reduce (0), we should get -1. Does this mean, what we are getting in this counter, i.e., 1111 indicates -1?

The following figure simulates binary countdown counter.



Summary about Ripple Counters

- Only LSB flip-flop controlled by the clock input
- Also known as RIPPLE COUNTER
- Modulus = number of stable states or counts in each flip-flop cycle
- Modulus = 2^N, where N= number of flip-flops
- Highest number in count =2^N-1

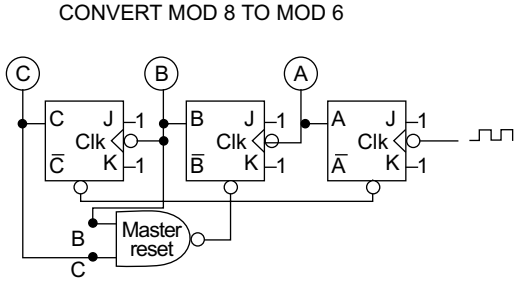
1.12.3.1.5 Building Counter from zero to X using JK Flip-flops (Any modulus counter)

Counters can be made to recycle after any desired count (say X) by using a gate to reset the counter. The following

steps can be followed to building a counter which goes from 0 to X.

1. First find out smallest number (N) of flip-flops needed. That is, $2^N >= X$.
2. Take NAND gate and connect its output to each flip-flop's clear line.
3. Determine for which flip-flop the output should be high for X. Connect those FFs Q lines as inputs of the NAND gate.

For example, we want to design mod 6 counter. So, we need 3 flip-flops as $2^3 >= 6$. Now, we connect Q lines of FFs B and C to NAND gate as shown below.

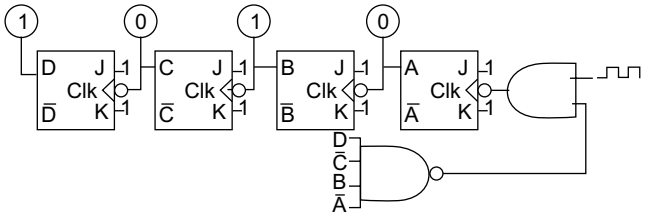


INPUT CLK			
	C	B	A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7			

UNSTABLE STATE

1.12.3.1.6 Building Self Stopping Counter

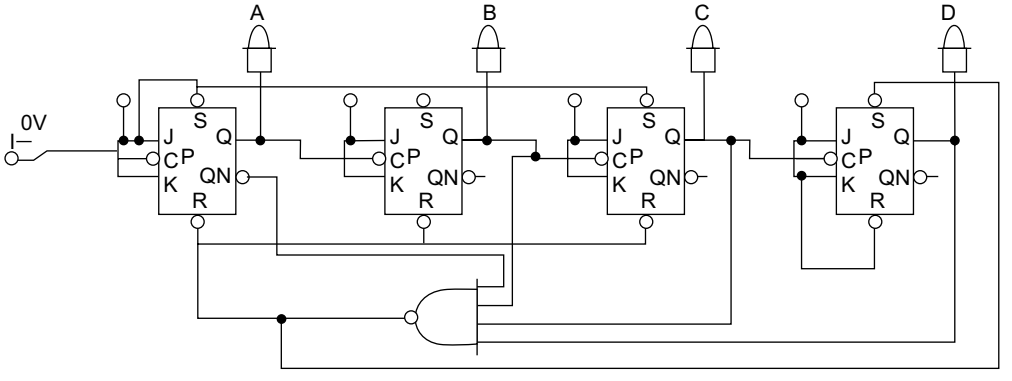
Counters can be made to stop counting after any desired count (X) by using a gate to inhibit the clock at the LSB FF.



For example, let us assume that we want counter to be stopped when its value reaches X(1010). We can simply achieve by taking a NAND gate connecting either output or complement outputs based on the binary code of the required X. Output of this NAND gate and clock are ORed with an OR gate. For example, the following counter stops when counter reaches 1010.

■ **Example** See the following counter circuit that uses JK flip-flops. Answer the following.

a. What is the value of the last usable state before the NAND gate resets the circuitry?
b. What value does the NAND gate reset the value to?
c. What is the modulus of this counter?
d. If count starts at decimal 11 and receives seven clock pulses, what is the new value on the counter?
e. What is the unstable state of the counter?



■ **Answer:**

a. $1101_2 = 13_{10}$ If we observe the above circuit, NAND gate inputs are FF A's complement and other FF's direct outputs. Thus, if FFs ABCD outputs are at 0111, then NAND gate resets all the flip-flops. Thus, last usable state is one less than 1110 (do remember A is LSB bit), i.e 1101=13.

b. $1000_2 = 8_{10}$. If we observe the circuit, we find NAND gate output is connected to R (reset) lines of FFs ABC while the same is connected to S (set) line of FF D. Thus, when the counter reaches 1110, it resets the same to 1000, i.e., to 8.

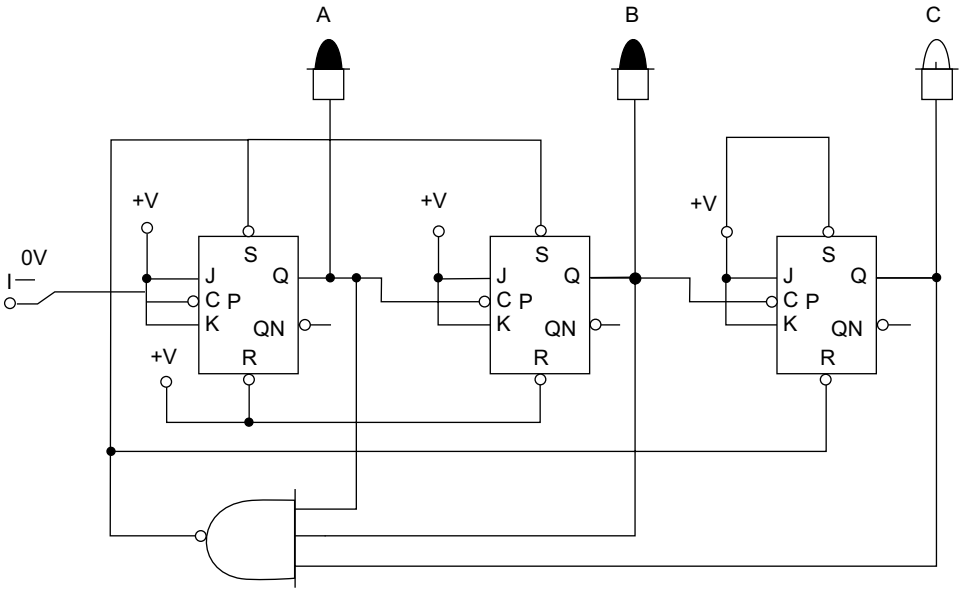
c. 6 . That is, we know from the above discussion that last usable state is 13 and least is 8. Thus, the number of states usable becomes 6.

d. 12_{10}. If the state is 11 (1011) then the probable sequence is: 1011, 1100, 1101, 1110 (unstable so goes to 1000) ->1000, 1001, 1010, 1011, 1100.

e. $1110_2 = 14_{10}$. (should be clear from the above discussion)

■ **Example** See the following circuit and answer the following.

a. What is the value of the unstable state, in decimal?
b. At what value does the NAND gate set the counter to?
c. If QA=1, QB=1, and QC=0, and 5 clock pulses are applied then what is the final stable state of the counter.
d. What is the modulus of this counter?

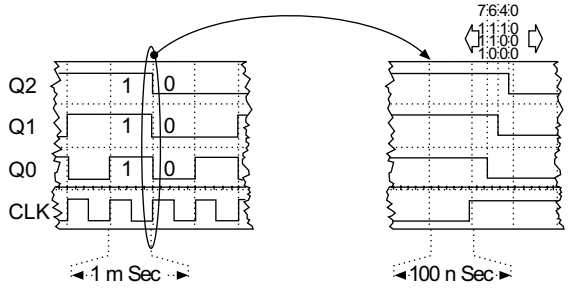


■ **Answer:**

a. 111_2 (7_{10}). If we observe the circuit, when NAND gates inputs becomes 111, counter goes to 011.
b. $011_2=3_{10}$
c. $100_2(4_{10})$
d. 4(Non stable state is 111. Therefore, last stable state is 110. When 111, counter goes to 011. Thus, number of possible states 110-011=6-3=4(011, 100, 101, 110).

■ **Example** What is ripple effect?.

As the clock input "ripples" from the first flip-flop to the last, the propagation delays from the flip-flops accumulate. This causes the Q outputs to change at different times, resulting in the counter briefly producing incorrect counts. For example, as a 3-bit ripple counter counts from 7 to 0, it will briefly output the count 6 and 4. The following figure contains timing diagrams one showing correct results while the other showing wrong results because of the accumulated delays.



1.12.3.2 Synchronous Counters

A synchronous counter is one in which all the flip-flops change state simultaneously since all the clocks inputs are tied together. Here, we will not be using the preceding bit as a pulse to any of the J-K flip-flops. Also, we assume that the same clock pulse must be used for each flip-flop. Now the following question arises.

How do we know when to toggle a flip-flop?

We may observe the expected of a binary counter as given above. The synchronous counter uses the following fact:

A bit will toggle if all the low-order bits in the previous state are 1.

That is, here we may make all flip-flops to toggle at the same time – but only if J-K's are high they will be acting as expected. Here, J-K's of a flip-flop are formed by ANDing the output of the previous bits (i.e., outputs of all the previous flip-flops).

Because, the new output comes after the toggle by a few nanoseconds. The new state is not involved in any of the inputs. The AND gates are sampling the previous state. We are deliberately using to our advantage the gate delay time. Figure 1.65 contains the 4-bit synchronous counter based on the above discussion.

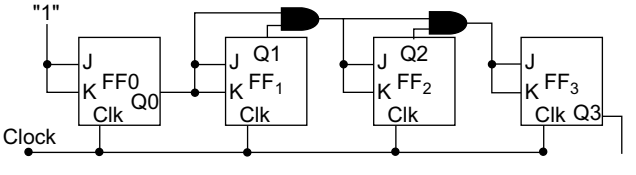


Figure 1.65 A Simple Synchronous Counter Consisting of J-K Flip-flops and AND gates

Counters are usually constructed of T flip-flops since the

flip-flops only have to toggle at a given sequence. A 3-bit synchronous counter is shown below in Figure 1.66.

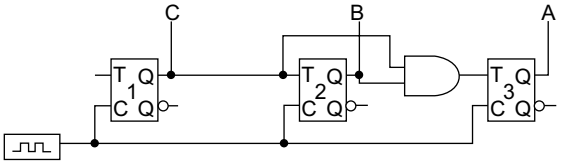


Figure 1.66 A 3-Bit synchronous counter using T flip-flops

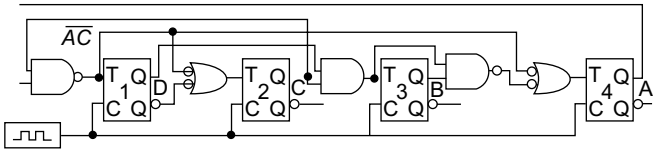
The equations for the flip-flops are *0*1 = 1; T2 = Q1; T3 = Q1 · Q2. Thus T1 toggles at every clock pulse, T2 toggles only when Q1 is high, on every other clock pulse, and finally T3 toggles when both Q1 and Q2 are high, or every fourth clock pulse. The counting sequence is shown below:

Count	A	STATE B	C
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0
9	r	e p e a	t

This type of counter that uses T flip-flops can be extended by the following set of equations.

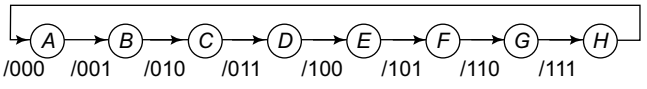
$$T_1 = 1$$
$$T_2 = T_1$$
$$T_3 = T_1T_2$$
$$T_n = T_1T_2 \ldots T_{n-1}$$

■ **Example** Prepare truth table and find out whether the following circuit can work like a Mod-11 counter or not.



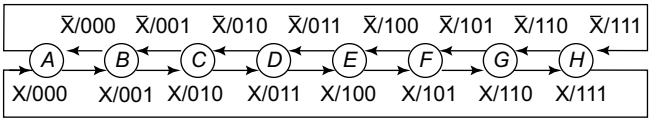
1.12.3.2.1 Synthesis of Synchronous Circuits

Let us take a simple example, the design of a Mod-8 counter using D-FFs. Since 8 distinct outputs are needed, the design calls for 8 states since each output will depend on the state we are in. The flow chart (transition diagram see chapter on State Machines) is as follows:



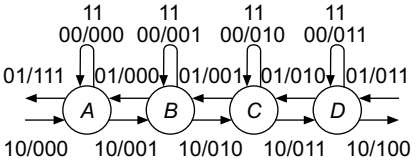
Each state is represented by a circled letter. The arrow points to the next state following a clock pulse. For a simple counter, the arrows just follow a string. The number following the slash shows the desired output at each transition. If there was an input it would be put in front of the slash.

Suppose we want an up/down counter dependent upon an external input, *X*. When *X* = 1 the counter has to count up and for *X* = 0 the counter has to count down. Thus, we now have arrows in both directions in the state diagram, dependent on *X*.



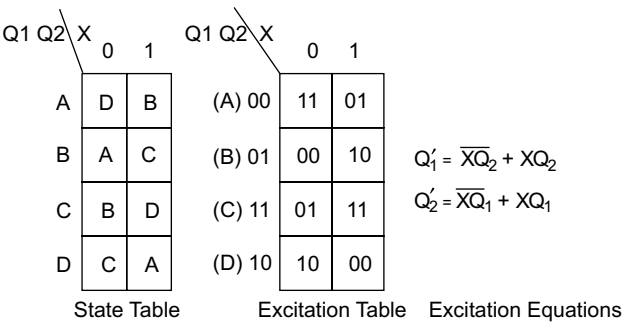
On $\bar{X}$ (when *X* = 0) the arrows point backwards. The outputs follow similar reasoning.

Now for a final extension assume that he new design calls for two inputs *X,Y*. On 0, 1 we want to have a down counter; on 1, 0 we want to have an up counter; and on 0,0 and 1,1 we want to stop counting. Only the first 4 states will be shown in the following state diagram.



Once a good working flow chart (or transition diagram) has been accomplished we arrive a state table and then driving equations are found by using K-Map.

Let us work on a simple 4 state up/down counter. As we need two states only, we need two flip-flops. We refer them as Q1 and Q2. Now, we shall prepare transition table. This table simply lists each state on the left, and the transition to the next state inside the box, dependent on the input variable *X*. From the state table, an excitation table is written. Using K-map techniques, a state representation is made and inserted for each state. Notice that only one variable changes at a time when using a K-map. Using K-map techniques, the equations for each flip-flop are found. Note that Q_1Q_2 are the outputs of flip-flops. The digits inside the box are for the inputs of the flip-flops.



Going back to the flow chart, an output table is written. It may seem a little odd that each state codes for two outputs, but you must realise that the coding is for the next state, not the immediate state. Finally, the equations for the outputs are written.

Q1 Q2 \ X	0	1
(A) 00	11	01
(B) 01	00	10
(C) 11	01	11
(D) 10	10	00

Output Table

$Q_1 = \overline{X}\overline{Q_2} + XQ_2 = Q_1'$

$Q_2 = \overline{Q_1}\overline{Q_2} + Q_1Q_2$

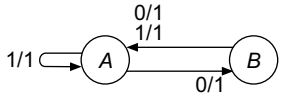
Output Equations

That is all there is to this type of synchronous synthesis. Once the equations are written, they can easily be converted to actual logic.

Here is one more example. It is desired to design a synchronous machine to decode a series of ones and zeros into a special output sequence. The input is *X* and the output is *Z*:

	start														continue →	
X_0	1	0	0	0	0	1	1	1	0	0	0	1	1	0	1	*X*
Z_0	1	1	0	1	0	1	1	1	1	0	1	1	1	1	1	*Z*

The first observation is that there is a 1 output for every 1 input. The next observation is that the output toggles on a zero input after the first zero input. Start with state *A*. On a 1 input, a 1 is output and there is no need to leave state *A*. On the other hand, there are two outputs for a zero input so it takes 2 states to code for the two outputs. Therefore the flow chart (or state diagram) is as follows:



Notation : INPUT/OUTPUT

Note that all possible inputs at every state are accounted for. In this particular case, there are a few simple flow charts that will do the trick. Experience will allow you to pick the simplest. In this case, since we will use a D-FF and assign 0 for state *A*, it is simplest to let everything fall back to *A*. The state-output table is:

X \	0	1
A	B/1	A/1
B	A/0	A/1

State-Output Table

Q \ X	0	1
(A) 0	1	0
(B) 1	0	0

Excitation Table

$Q'\text{-}X\overline{Q}$

Excitation Equation

The output table and equation is:

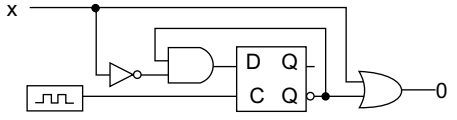
X \	0	1
(A) 0	1	1
(B) 1	0	1

Output Table

$Q\text{-}X=\overline{Q}$

Output Equation

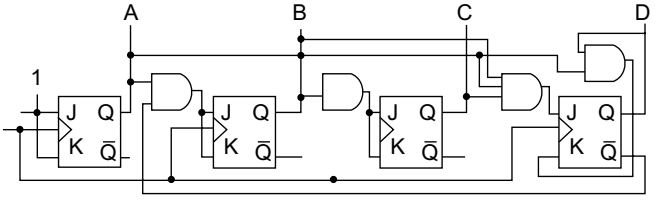
The final circuit is



■ **Example** Design and build a counter that will go through the following sequence. Show all work.

STATE A	B	C	D	
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	1	Note: state '4' is skipped
0	1	1	0	
0	1	1	1	
1	0	0	0	
0	0	0	0	
	r	e p e a	t	

■ **Example** Analyse weather the following works like a Decimal Counter or not.



■ **Answer:** In decimal counter, we want the counter should go from 0 to 9. After 9, it should come back to state 0.

Clock Cycle	ABCD	Remarks
	0000	Initially
1	1000	
2	0100	B sets while A gets cleared. AND gate of B gives 1 as A is 1 and D' is 1.
3	1100	AND gate of B gives 0 as A is 0.
4	0010	AND gate of C gives 1 as both A and B are 1.
5	1010	
6	0110	It continues till 9 and return to 0.

1.12.3.2.2 Design of Synchronous Counter

- Decide which type of flip-flop to be used in the counter design.
- Prepare table such that it will be having current state and next state of the counter. Also, based on the type of flip-flop selected identify what should be flip-flops control inputs to achieve the required transition. That is, for example we are using JK flip-flops and we want a flip-flop state to be changed from 0 to 1. This, we can achieve by either place 1, 0 or 1, 1 on its J and K likes. That is, its J and K lines can be 1 and d (don't care). Similarly, if want to change from 1 to 0 then J and K lines can be either 0, 1 or 1, 1. That is, J and K lines can be d (don't care), 1. Do include required output variables in the table.
- For each control inputs of the flip-flops, use combinational design to implement the required circuit. Repeat the same for output variables also.

■ **Example** Design of a 3-bit counter using JK flip-flops.

■ **Answer:** Before really proceeding to our design, first see the following table which contains the required inputs on J and K lines of a JK flip-flop to achieve a required transition. User are advised to remember that this is derived from excitation table of JK flip-flops discussed elsewhere in this chapter.

Required transition	J	K
O to 0	0	d
0 to 1	1	d
1 to 0	d	1
1 to 1	d	0

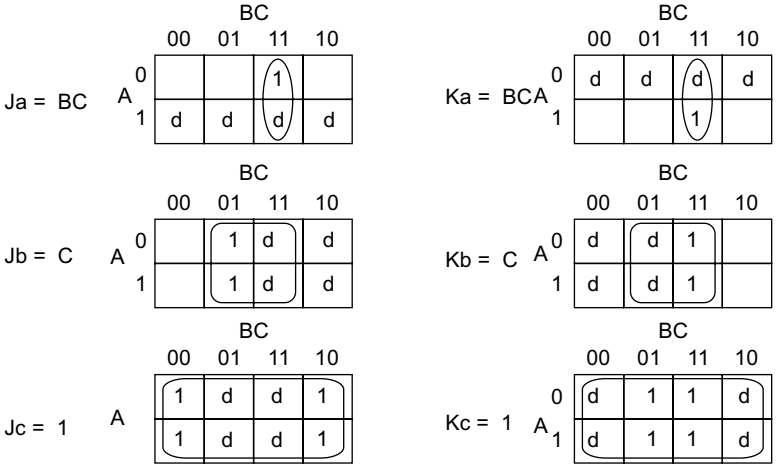
In the following, we have prepared table along with required control inputs on the J and K lines of each of the flip flops. For, example if we consider the counter is at 000 and we want it to go to 001 then on first and second flip-flop's J and K lines we have to place 0 and d while for third flip-flop we have to place 1 and d. Like this, the following table is prepared. We have used Ja, Ka, etc., to refer to J and K lines of first flip-flop A. Now, we have to prepare K-Map for each of the J and K lines and find out the driving equations for them as shown below.

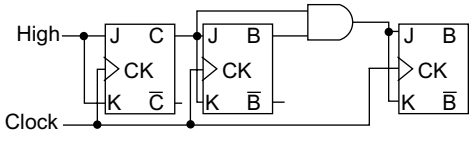
From the logic equations, we have drawn the counter diagram.

(a) Truth table

A	B	C	Ja	Ka	Jb	Kb	Jc	Kc
0	0	0	0	d	0	d	1	d
0	0	1	0	d	1	d	d	1
0	1	0	0	d	d	0	1	d
0	1	1	1	d	d	1	d	1
1	0	0	d	0	0	d	1	d
1	0	1	d	0	1	d	d	1
1	1	0	d	0	d	0	1	d
1	1	1	d	1	d	1	d	1

(b) Karnaugh maps





(c) Logic diagram

■ **Example** Design a 3-bit counter that counts in the sequence 101, 010, 000, 111, 101… Choose flip flops with the appropriate control signals so that a single reset signal causes the counter to load the initial value 101. Draw the gate-level schematic of the resulting circuit.

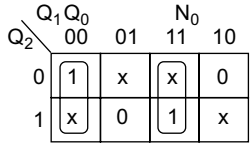
a. Use D flip-flops in the design.
b. Use T flip-flops only in the design
c. Use JK flip-flops in the design.

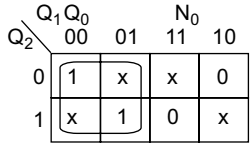
■ **Answer:**

a. First, we shall prepare transition table. The same is given below.

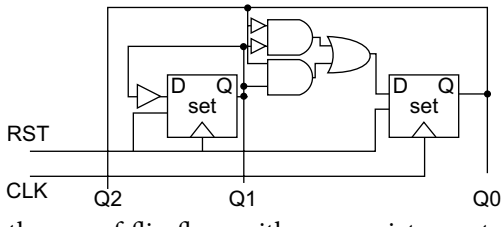
Q2	Q1	Q0	N2	N1	N0
0	0	0	1	1	1
0	0	1	X	X	X
0	1	0	0	0	0
0	1	0	X	X	X
1	0	0	X	X	X
1	0	1	0	1	0
1	1	0	X	X	X
1	1	1	1	0	1

Here, Q2, Q1, Q0 are the outputs of the D flip-flops while N2, N1, N0 are the lines which are inputs to the D flip-flops. Notice that N2 can always be equal to N0. This makes sense since in our count sequence, the most significant bit (MSB) is always the same as the least significant bit (LSB). Next we draw the K-maps for N0 and N1.





From the Karnaugh map, we can find N0=Q1'Q0+Q1Q0, N1=Q1'. Notice that N0 and N1 are not functions of Q2. Also, since N2 = N0, we know that Q2 = Q0. Therefore, there is no need of Q2 flip flop at all. The resultant circuit can be given as:

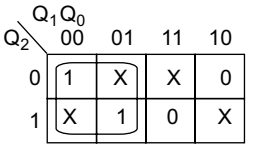


Notice the use of flip flops with appropriate reset and set capabilities. This will cause the counter to load the value "101" when reset is asserted.

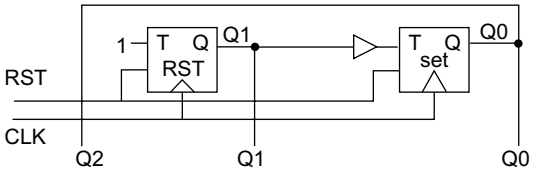
b. Even if we assume that we take T flip-flops, as Q2 is same as Q0 we need not required to worry about Q2. Let us consider preparing transition table for Q1 and Q0 only. We assume users know the behavior of T flip-flops which change (toggle) their state when their T lines becomes logic 1. Transition table can be given as:

Q2Q1Q0	Next State of Q2Q1Q0	T1T0
000	111	1 1
001	XXX	X X
010	000	1 0
011	XXX	X X
100	XXX	X X
101	010	1 1
110	XXX	X X
111	101	1 0

If we prepare Karnaugh map for T1 column and group 1s, we get T1=1. Similarly, Karnaugh map for T0 is given as:



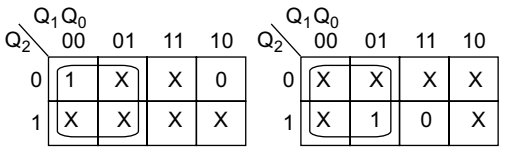
From the Karnaugh map, we can find T0=Q1'. Thus, the circuit is given as:



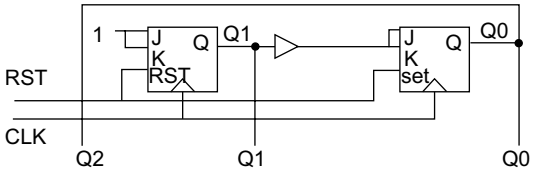
c. Now, we assume JK flip-flops instead of D flip-flops. As usual, we don't worry about Q2 as Q2 is same as Q0. Transition table and required J,K controls for flip-flops Q1 and Q0 is given below.

Q2Q1Q0	Next State of Q2Q1Q0	J1K1	J0K0
000	111	1 X	1 X
001	XXX	X X	X X
010	000	X 1	0 X
011	XXX	X X	X X
100	XXX	X X	X X
101	010	1 X	X 1
110	XXX	X X	X X
111	101	X 1	X 0

If we prepare Karnaugh maps for J1, K1 columns, we may find J1=K1=1. Similarly, Karnaugh maps for J0, K0 are given below from which we can find J0 = K0 = Q1'.



Thus, the final counter circuit with resetting feature can be given as:

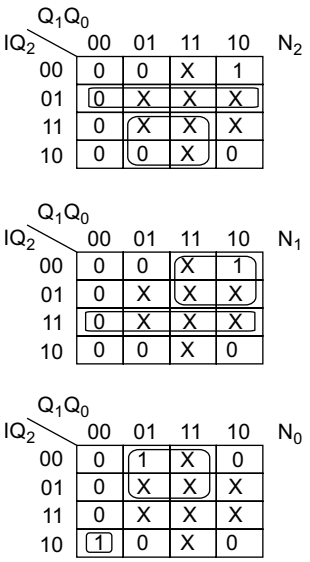


■ **Example** Design a counter, with an **inc** input, that counts in the sequence 00, 01, 00, 10, 00… Do remember that this is tricky counter as this counter has to give the same value twice in the count sequence.

■ **Answer:** Since the same value appears twice in the count sequence, we need an extra bit to tell us if the current count is the first "00" or the second "00". Let's use Q1Q0 as the count outputs and Q2 as the extra bit, where Q2 = 0 is for the first "00" and Q2 = 1 is for the second "00". The actual count sequence we'll implement is therefore 000, 001, 100, 010, 000…

The transition table assuming that D flip-flops are used is given below. We assume N2, N1, N0 are lines which are connected to D lines of flip-flops Q2, Q1, and Q0, respectively,

inc	Q_2	Q_1	Q_0	N_2	N_1	N_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	X	X	X
0	1	0	0	1	0	0
0	1	0	1	X	X	X
0	1	1	0	X	X	X
0	1	1	1	X	X	X
1	0	0	0	0	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	0
1	0	1	1	X	X	X
1	1	0	0	0	1	0
1	1	0	1	X	X	X
1	1	1	0	X	X	X
1	1	1	1	X	X	X



Karnaugh maps for N2, N1 and N0 are given above. From the Karnaugh maps, driving equations for

N2 =inc'Q2+inc.Q0

N1 =inc'Q1+inc.Q2

N0 =inc'Q0+inc.Q2'.Q1'.Q0'

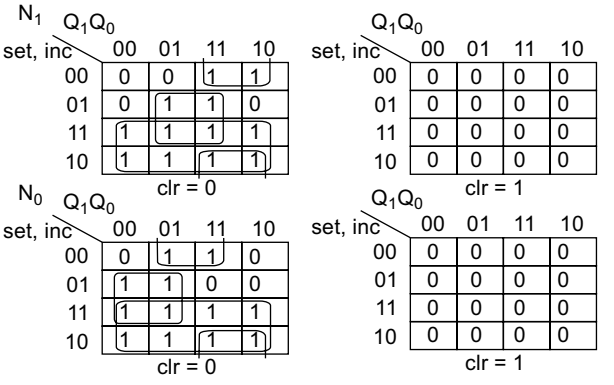
■ **Example** Design a 2-bit gray code counter (i.e., having the sequence 00, 01, 11, 10, 00…) with **clr**, **set**, and **inc** inputs. Make **clr** the highest priority, followed by **set** and **inc**. The **clr** input should cause the counter to load zeros, the **set** input should cause the counter to load ones, and the **inc** input should cause the counter to transition to the next sequential gray code value.

■ **Answer:** We propose to use two D flip-flops. We assume their control lines as N1, N0.

Inputs to the system are five which includes control signals set, clr, inc in addition to the outputs of the two D flip-flops which are used to represent counter values. The transition table is given as:

clr	set	inc	Q1	Q0	N1	N0
0	0	0	0	0	0	0
0	0	0	0	1	0	1
0	0	0	1	0	1	0
0	0	0	1	1	1	1
0	0	1	0	0	0	1
0	0	1	0	1	1	1
0	0	1	1	0	0	0
0	0	1	1	1	1	0
0	1	-	-	-	1	1
1	-	-	-	-	0	0

In the above table, we have not displayed all the rows related clr = 1, as all the possibilities are same. Karnaugh maps for N1 and N0 are given below.



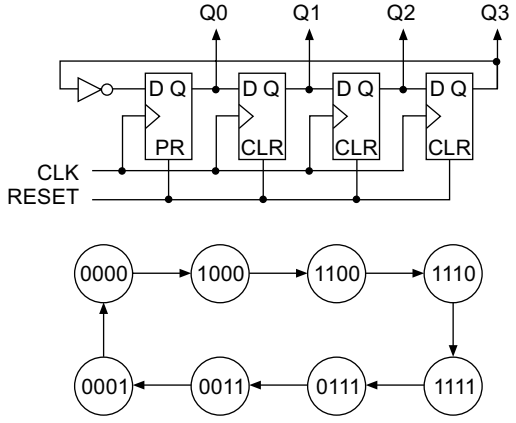
From the Karnaugh maps, the driving equation for N1, N0 are given as:

$$N1 = clr' \cdot (set + inc \cdot Q0 + inc' \cdot Q1)$$

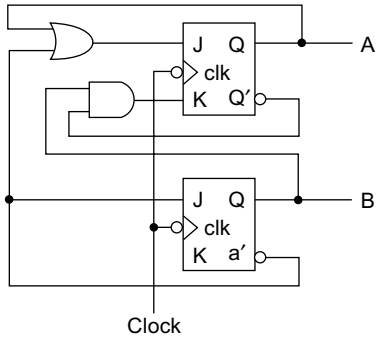
$$N0 = clr' \cdot (set + inc \cdot Q1' + inc' \cdot Q0)$$

Johnson Counter

Johnson counter gives the following sequence with a four flip-flops: 0000, 1000, 1100, 1110, 1111, 0111,0011, 0011, 0000. The following circuit satisfies our requirement. Here, we have simply added last flip-flops complement to D line of first flip-flop.



■ **Example** Analyse the following circuit which contains a JK and T flip-flop. Assume initially both the flip-flop's are at their clear state.

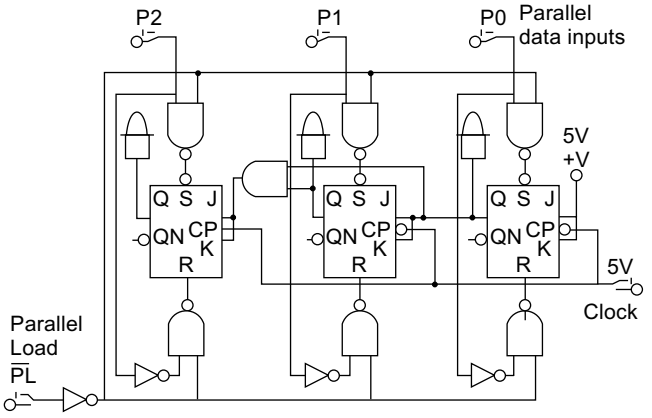


■ **Answer:** The following table illustrates the behavior of the above circuit for each clock. Circuit stays at 11.
Initial State

JK&T Flip-flops	AND gate output	OR gate output	Next state of JK & T Flip-flops
00	0	1	11
11	0	1	11

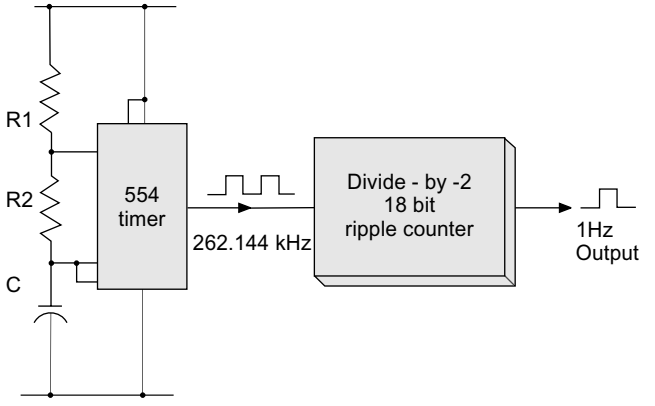
Pre-settable Counter

We can also design a counter with which we can specify required count value before incrementing. The following circuit illustrates the same. We have P0 to P2 are parallel lines. When we make PL line as low, some of the FFs will be set. That is, we can load data on parallel input lines to the counter in asynchronous manner. After wards, for each clock pulse counter will be changing its state as defined.



1.12.4 Frequency Dividers

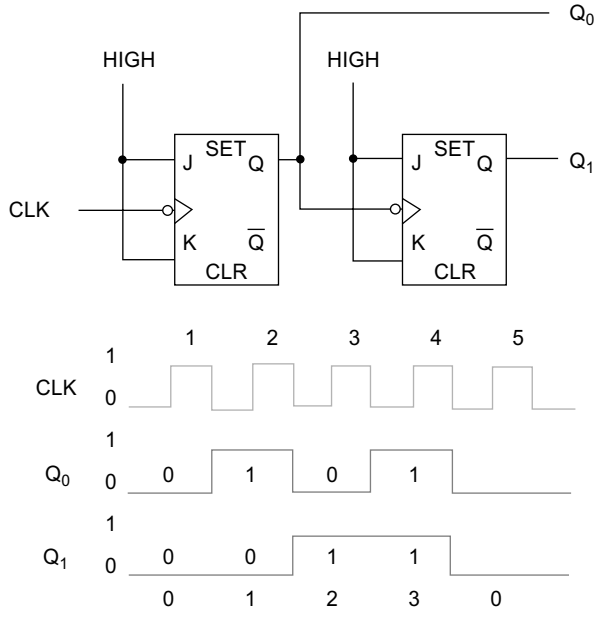
We have studied timing diagrams of ripple counters. We know that output of a N bit ripple counter generates output signal with the frequency of 1/2N of the frequency of the input clock. For example, assume we require an accurate 1Hz timing signal to operate a digital clock. We could quite easily produce a 1Hz square wave signal from a standard 555 timer chip but the manufacturers data sheet tells us that it has a typical 1-2% timing error depending upon the manufacturer, and at low frequencies a 2% error at 1Hz is not good. However, the data sheet also tells us that the maximum operating frequency of the 555 timer is about 300kHz and a 2% error at this high frequency would be acceptable. So by choosing a higher timing frequency of say 262.144kHz and an 18-bit ripple (Modulo-18) counter we can make a precision 1Hz timing signal as shown below. A simple 1Hz timing signal using an 18-bit ripple counter/divider.



What is a divider?

We have explained about a 2-bit counter using JK flip-flops along with timing details. The only input of the circuit is a square wave with a fixed frequency *f*. When we look at the output waves of each output terminal independently, we can easily find that the output Q_0, and Q_1, are both a square wave with a fixed frequency 0.5*f*, 0.25*f*, respectively.

Then, we can see for output Q_0, this circuit can be called as half-frequency divider and for output Q_1, it is called quarter-frequency divider. Thus, we can obtain that the counter and divider is almost the same machine with only difference of output terminals.



How to design a divider?

There are plenty of ways to design a divider with elements such as flip-flops and gates. We will follow the steps of designing a sequential machine. First, we should think about how many states are there. We can see that the amount of states depends on which frequency of signal we want to obtain (compared with basic frequency f[1]). 0.25f, 1/3f, or even 1/8f. For 0.25f, we need 4 states and for 1/3f we need 3 states, then, needless to say 1/8f need 8 states indeed.

Then we can make out a transition table make sure that what will the next-state be in each case and their outputs. The outputs are designed by ourself; however, we can't make them to be all 0 or all 1. Otherwise, we will see that the output signal is not a periodic signal at all the then saying some words such as frequency would be meaningless. Still, we can design in which case the output would be 1 and in which case it would be 0. How long does the output being 1 in one cycle would be mentioned as the term **'duty cycle'** which means that the ratio of time being 1 in each cycle and the time of one cycle.

With the help of Karnaugh maps and transition equations, we can derive out the expression of each input and then, we can draw a circuit diagram.

Building dividers with D flip-flop loops

One classical way to build a divider with D flip-flop is known as D flip-flop loop is given below. These kinds of circuits, unlike most sequential circuit, contain two parts only: input circuits and output circuits. Thus, the locations of input circuits are different than those in usual sequential circuit. In usual case, there would be input circuits at each input terminal of flip-flops (or say D terminal). But here, no matter how many flip-flops are there, we only need to consider about the input terminal of the first flip-flop. For other input terminals of following flip-flops, we just connect them directly with the output terminal of the previous flip-flops. The output circuits have the same function as in usual case.

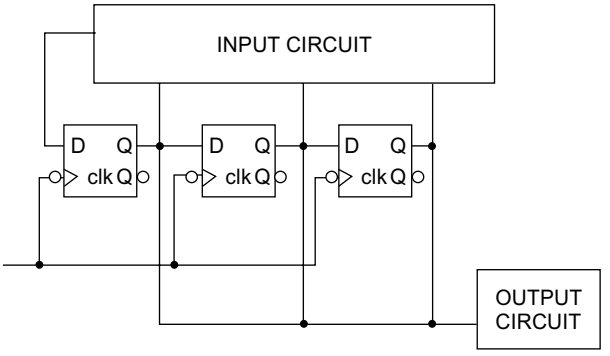


Figure 1.67 D flip-flop loop

First, we should find out a sequence of code that can be assigned to each state. For instance, if we have decided to build a 1/5-frequency divider, then we have to take 5 states in our machine. Then, we have to assign one unique code each state. However, designing a code system is not that easy as when we design a sequential machine. The code system which we have used is special and shared the same specificity: *a previous code in the system can be obtained by left shift the next code with a supplement bit either '1' or '0'. And for the last one, it should be obtained by left shift the first code.* This condition is really strict and we would be annoyed a lot by this condition when we design our own divider. Such complex code system would ensure that the state of our machine would run in a cycle.

In most cases, we can assign an initial value to each D flip-flop which means that the initial values of the terminal Qs can be assigned to any one state we have assigned with its code. Then, what we should do is just design parts of combinational circuits which make sure that the input of most left D flip-flop can make up the code of next state after one clock cycle.

In some situations, we can use the Q terminal of one D flip-flop as the output terminal. However, we may need to build more complex output circuit in order to change the duty cycle.

[1] In the following parts of the essay, if we don't make an announcement, the letter f will always mean the basic frequency of the original input signal.

■ **Example** 1/3-frequency divider

First, we should build our own code system. Fairly easy, we can obtain the following one: '00' →'10'→'01'→'00'. Then, we can receive the transition table:

Current State (Q_AQ_B)		Next State (D_BD_A)	
0	0	0	1
0	1	1	0
1	0	0	0
1	1	X	X

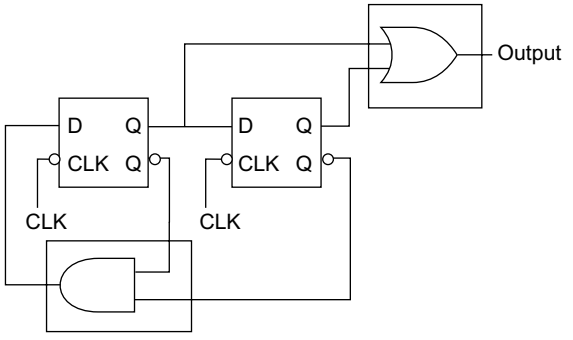
As we mentioned above, we only focus on the input terminal of first D flip-flop. Thus, next A (or D_A for avoiding misunderstanding) is using Karnaugh map:

$$D_A = (Q_A+Q_B)' = Q_{A'} \times Q_{B'}$$

Then, we should decide the output, which decides the duty cycle. Take duty cycle is 66.7% or 2:3, we can obtain output from the truth table as: Output = $Q_A + Q_B$

Current State (Q_AQ_B)		Output
0	0	0
0	1	1
1	0	1
1	0	1

In the red frame, it is input circuit, in the blue frame, it is output circuit.



■ **Example** 1/5-frequency divider

■ **Answer:** We take the following state codes and their transitions

'000' → '100' → '110' → '011' → '001' → '000'

Truth table of both Inputs and Outputs

Current State ($Q_AQ_BQ_C$)	Next D_A	Output (Duty Cycle: cycle 40%)	Output (Duty Cycle: 50%)		
0	0	0	1	0	~CLK
1	0	0	1	0	0
1	1	0	0	0	0
0	1	1	0	1	1
0	0	1	0	1	1

A. Input equation: $D_A = Q_{B'} \times Q_{C'} = (Q_B + Q_C)'$

Output equation: Output = QC (duty cycle: 40%)

Output = $(Q_{A'}+Q_C)\times(Q_C+CLK')$ (duty cycle: 50%)

Circuit Diagram can be given as:

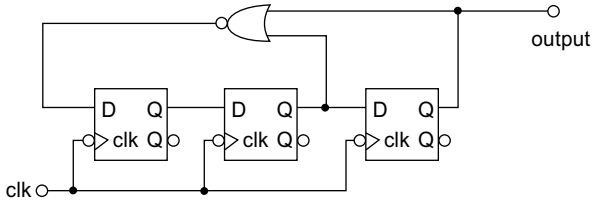


Figure 1.68 1/5-frequency divider (D flip-flop, Duty cycle: 40%)

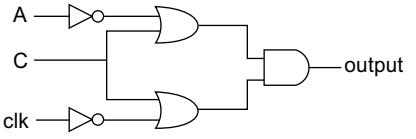


Figure1.69 Output circuit for 1/5-frequency divider (D flip-flop, Duty cycle: 50%)

■ **Example** Input circuits for 1/7-frequency and 1/13-frequency divider

■ **Answer:** The following state codes are taken for 1/7 frequency divider.

'000' → '100' → '010' → '101' → '110' → '011' → '001' → '000'

For 1/13-frequency divider:

'0000' → '1000' → '0100' → '1010' → '0101' → '0010' → '1001' → '1100' → '1110' → '1111' → '0111' → '0011' → '0001' → '0000'

Truth table for Next DA

Current State ($Q_AQ_BQ_C$)			Next D_A
0	0	0	1
1	0	0	0
0	1	0	1
1	0	1	1
1	1	0	0
0	1	1	0
0	0	1	0

Current State($Q_AQ_BQ_CQ_D$)				Next D_A
0	0	0	0	1
1	0	0	0	0
0	1	0	0	1
1	0	1	0	0
0	1	0	1	0
0	0	1	0	1
1	0	0	1	1
1	1	0	0	1

1	1	1	0	1
1	1	1	1	0
0	1	1	1	0
0	0	1	1	0
0	0	0	1	0

A. Next $D_A = Q_A$ XNOR Q_C (for 1/7-frequency divider)

Next $D_A = (Q_A + Q_D + Q_B \times Q_C)' + Q_A \times (Q_B$ Xor $Q_D)$
Circuit Diagram is given as:

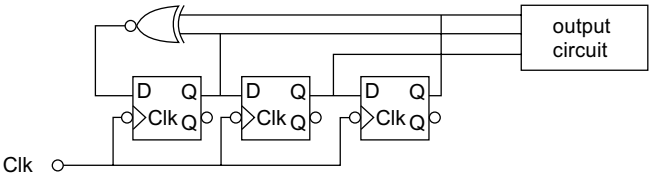


Figure 1.70 1/7-frequency divider (D flip-flop)

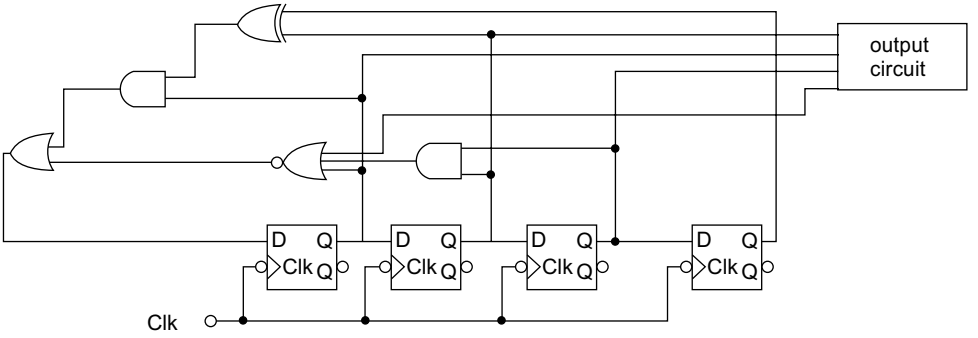


Figure 1.71 1/13-frequency divider (D flip-flop)

Building $1/2^n$ -frequency dividers with J-K flip-flop

We need n J-K flip-flops to build $1/2^n$ -frequency divider. All of them should be cascaded. And the output terminal should be the Q terminal of the last J-K flip-flop.

Build dividers with J-K flip-flops

First, we should decide how we count. Unlike the case of D flip-flop loop, the order of counting has no limitation. You can have your choice.

Then we can obtain a transition table, and see what would J_n K_n be at state n–1. Then, just as when we design a sequential circuit, we should use Karnaugh map to get the input equation of each J-K flip-flop.

You may think that the analysis would be too complex since you may use n-variable Karnaugh map for 2n times. However, the struggle for making out a code system would be vanished. Sometimes, in order to make the designing more easily, we connect J terminal and K terminal of a flip-flop together and then the designing would be little easier. When the output of the flip-flop remains to the previous state, the input ought to be one, otherwise it would be zero. Then we need only to use Karnaugh Map for n times. Sometimes if the output of the flip-flop doesn't contain the unit 'X→0→0→X', then you can connect the J terminal with the HIGH signal directly. Similarly, the K terminal can be connected with the HIGH signal directly if the output of the flip-flop doesn't contain the unit 'X→1→1→X'.

■ **Example** 1/3-frenquency divider

We need 2 flip-flops to obtain the final results, and suppose our count order is '00→01→10', then we can obtain the following transition table:

Current state (Q_AQ_B)		Next J_A	Next K_A	Next Q_A	Next J_B	Next K_B	Next Q_B
0	0	*0*	*1*	0	*1*	*1*	1
0	1	*1*	*1*	1	*0*	*1*	0
1	0	*1*	*1*	0	*0*	*1*	0

Although you can fill in the column Next J_A, K_A, J_B, K_B now by the truth table of J-K flip-flop, that would make the problem become little complicated. By observing the cycle of Q_A and Q_B, we see that the cycle of Q_A is '0→1→0→0→1→0', we can see that the cycle contains the unit 'X→0→0→X', thus, K_A terminal can be connected with the HIGH signal directly. For the same reason, the cycle of Q_B doesn't contain 'X→1→1→X', that is to say, the KB terminal may be connected with signal HIGH directly. Then, according to the truth table of J-K flip-flop, we can decide the value of J_A and J_B.

In this example, the value of KA and KB has been fixed. Then we may consider the input equation of JA and JB. By Karnaugh map, we can get

$$J_A = Q_A + Q_B$$

$$J_B = (Q_A + Q_B)' \text{ or } J_B = J_A'$$

Thus, the circuit diagram can be drawn as:

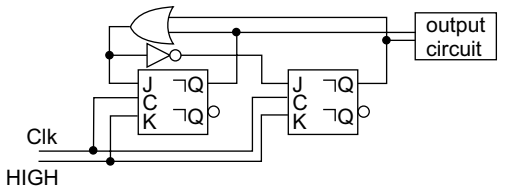


Figure 1.72 1/3-frequency divider (J-K flip-flop)

■ **Example** 1/5-frenquency divider

Current State ($Q_AQ_BQ_C$)			NEXT Q_A	NEXT Q_B	NEXT Q_C	NEXT J_A	NEXT K_A	NEXT J_B	NEXT K_B	NEXT J_C	NEXT K_C
0	0	0	0	0	1	1	1	1	1	0	0
0	0	1	0	0	1	0	0	1	1	1	0
0	1	0	0	0	1	0	0	1	1	1	1
0	1	1	1	1	1	1	1	1	1	0	0
1	0	0	0	0	1	1	1	0	1	0	0

The transition table and the input table have been already shown above. There is only one thing to say, when decide the value of J_B and K_B, we can see that the units, both 'X→0→0→X' and 'X→1→1→X' can be found. Then, we can connect the JB and KB together, once the output changes, the input should become both '1', otherwise, they remain both '0'.

Then, we can derive out the input equations:

$$J_A = Q_B \times Q_C \text{ and } K_A = \text{HIGH}$$

$$J_B = K_B = Q_B \text{ XNOR } Q_C$$

$$J_C = Q_A' \text{ and } K_C = \text{HIGH}$$

Then, the circuit diagram can be drawn as:

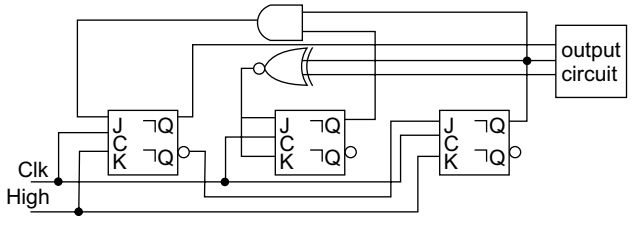


Figure 1.73 1/5-frenquency divider (J-K flip-flop)

Cascade

Just like the cascaded transistor circuit, $Av = Av_1 \times Av_2 \times \ldots \times Av_n$, the function of the dividers may be integrated as a bigger one. A half-frequency divider cascade with a 1/3-frequency divider can be used as a 1/6-frequency divider. Thus, if we want to change a 1 KHz square frequency to 1 Hz square frequency, since $1000 = 5 \times 5 \times 5 \times 2 \times 2 \times 2$, we can just cascade 3 half-frequency divider and 3 1/5-frequency divider. The output of the previous level divider should be connected with the terminal CLK in all the circuit diagrams above.

■ **Example** VI: 1/1000-frenquency divider

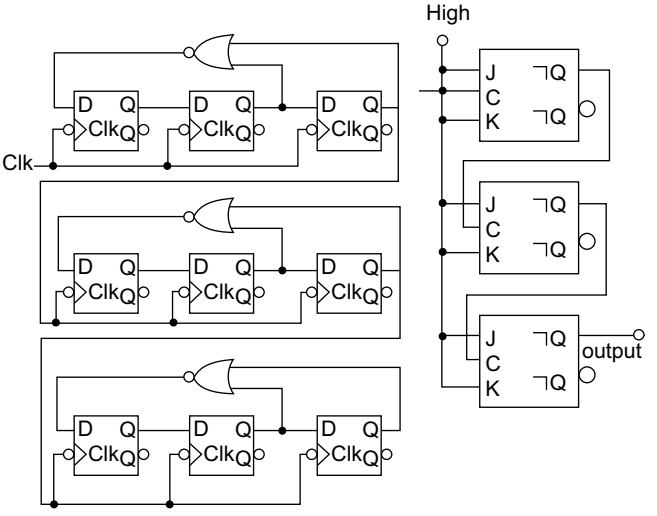
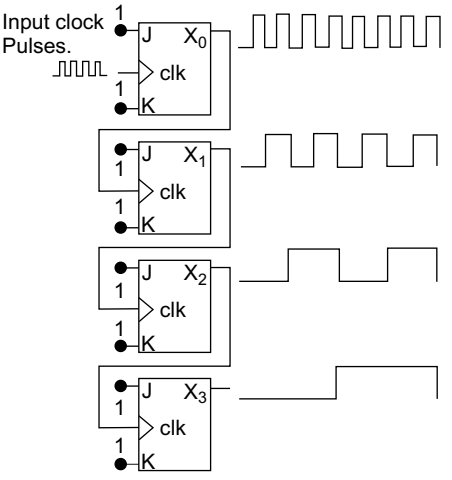


Figure 1.74 1/1000-frequency divider (duty cycle: 50%)

As mentioned, in Figure 1.74, on left there are three 1/5-frequency dividers cascaded here, and on the right, there is one 1/8-frenquency divider. The red, blue and green lines are connections between each level. However, the output of each level has a limitation: *there can only be one '1' and one '0' in each cycle of the output signal.* The duty cycle of the output signal namely the signal we obtained at last is depend on the output circuit of the last divider only.

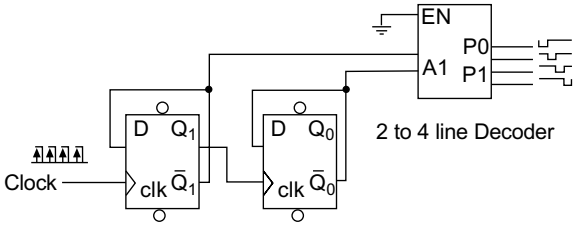
■ **Example** Explain the functionality of the following frequency divider.



The J and K inputs of each flip-flop are set to 1 to produce a toggle at each cycle of the clock input. For each two toggles of the first cell, a toggle is produced in the second cell, and so on down to the fourth cell. (This produces a binary number equal to the number of cycles of the input clock signal. This device is sometimes called a "ripple through" counter)

Thus, it is useful as a frequency divider which generates multiple frequencies. If we observe, output of first flip-flop follows clock pulse, while second flip-flop output will be having half the clock pulse and vice versa.

Process timer: Used in combination with a line decoder, a counting circuit like this could be used to implement a process stepper. As the D flip-flops counts down through a binary 4 count, the complement of the output is sent to the decoder which enables one and only one of the output lines (enabled Low). The truth table for the decoder is given on the table below. We can connect four devices each of the output lines such that every device will be selected on a round robin fashion.



A	A	P0	P1	P2	P3
1	0				
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

■ **Example** Explain how CRC calculation discussed in chapter on "Number System" can be implemented using shift registers.

■ **Answer:** We know that we will having a series of XOR operations while calculating CRC checksum. We advise readers to refer the previous chapter before continuing further. Here, we propose to employ a shift register, which we may denote as CRC which is of length r (the degree of CRC polynomial) bits. That is, shift register is made to have r flip-flops (In our case, we assume that flip-flops are D type Flip-Flops). When the subtractions (exclusive or's) are done, it is not necessary to represent the high-order bit, because the high-order bits of CRC polynomial and the quantity it is being subtracted from are both 1. The division process might be described informally as follows:

Initialise the CRC register to all 0-bits.

Get first/next message bit m.

If the high-order bit of CRC is 1.

Shift CRC register content and m together left 1 position, and XOR the result with the low-order r bits of CRC polynomial.

Otherwise,

Just shift CRC register content and m left 1 position. If there are more message bits, go back to get the next one.

It might seem that the subtraction should be done first, and then the shift. It would be done that way if the CRC register held the entire generator polynomial, which in bit form is bits. Instead, the CRC register holds only the low-order r bits of CRC polynomial, so the shift is done first, to align things properly.

Below is shown the contents of the CRC register for the generator CRC polynomial (G) = $x^3 + x + 1$ and the message M = $x^7 + x^6 + x^5 + x^2 + x$ Expressed in binary, G = 1011 and augmented M = 11100110.

000 Initial CRC contents. High-order bit is 0, so just shift in first message bit.

001 High-order bit is 0, so just shift in second message bit, giving:

011 High-order bit is 0 again, so just shift in third message bit, giving:

111 High-order bit is 1, so shift and then XOR with 011, giving:

101 High-order bit is 1, so shift and then XOR with 011, giving:

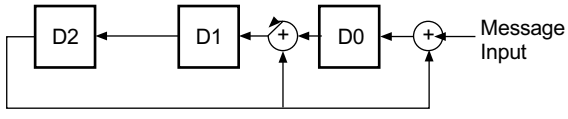
001 High-order bit is 0, so just shift in fifth message bit, giving:

011 High-order bit is 0, so just shift in sixth message bit, giving:

111 High-order bit is 1, so shift and then XOR with 011, giving:

101 There are no more message bits, so this is the remainder.

These steps can be implemented with the (simplified) circuit shown in the figure, which is known as a feedback shift register.



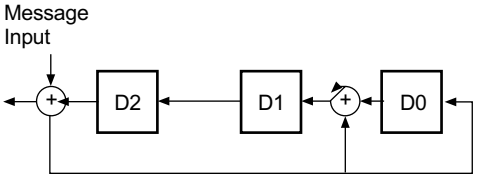
Polynomial division circuit for $G=x^3 + x+1$

The three boxes in the figure represent the three bits of the CRC register. When a message bit comes in, if the high-order bit (D2 box) is 0, simultaneously the message bit is shifted into the D0 box, the bit in D0 is shifted to D1, the bit in D1 is shifted to D2, and the bit in D2 is discarded. If the high-order bit of the CRC register is 1, then a 1 is present at the lower input of each of the two exclusive or gates. When a message bit comes in, the same shifting takes place but the three bits that wind up in the CRC register have been exclusive or'ed with binary 011. When all the message bits have been processed, the CRC holds M mod G.

If the circuit of figure that follows were used for the CRC calculation, then after processing the message, r (in this case 3) 0-bits would have to be fed in. Then the CRC register would have the desired checksum, But, there is a way to avoid this step with a simple rearrangement of the circuit.

Instead of feeding the message in at the right end, feed it in at the left end, r steps away, as shown in figure that follows .

This has the effect of pre-multiplying the input message M by xr. But pre-multiplying and post-multiplying are the same for polynomials. Therefore, as each message bit comes in, the CRC register contents are the remainder for the portion of the message processed, as if that portion had r 0-bits appended.



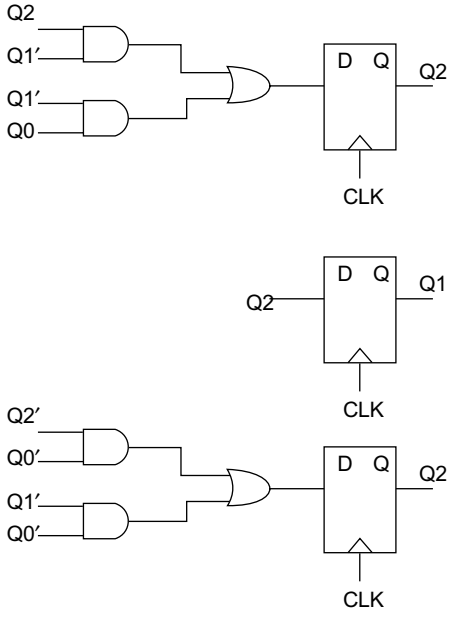
CRC circuit for $G=x^3 + x+1$

■ **Example** Write down at least two functions of a register.

■ **Answer:**

1. Registers are operating as a coherent unit to hold and generate data.
2. Registers functions also include configuration and start-up of certain features, especially during initialization, buffer *storage* e.g. video memory for graphics cards, input/output (I/O) of different kinds,

■ **Example** The following circuit is designed to generate the following sequence
000 001 100 110 011 000. Verify its validity.



N2 = Q2 Q1′ + Q1′ + Q0
N1 = Q2
N0 = Q2′Q0′ + Q1 Q0′

■ **Answer:** We consider N2, N1, N0 are the lines which drive or controls the D flip-flops. We know D flip-flop follows its D line. That is, if it's D line is 1, flip-flops stores 1, and vice versa. Thus, if we want circuit has to change from 000 to 010, we have to make N2, N1 and N0 as 010. However, in the problem description, the expected circuit behavior is not defined for states **010 101 111.** Thus, we take those states as donot cares. The following table explains flip-flop states what should be N2, N1 and N0 lines to go to next state.

Q2	Q1	Q0	N2	N1	N0
0	0	0	0	0	1
0	0	1	1	0	0
1	0	0	1	1	0
1	1	0	0	1	1
0	1	1	0	0	0
0	1	0	X	X	X
1	0	1	X	X	X
1	1	1	X	X	X

Now, we prepare K-Maps for N2, N1 and N0.
K-Map for N2 is given as:

Q2\Q1Q0	00	01	11	10
0	0	1	0	x
1	1	x	x	0

K-Map for N1 is given as:

Q2\Q1Q0	00	01	11	10
0	0	0	0	x
1	1	x	x	1

K-Map for N0 is given as:

Q2\Q1Q0	00	01	11	10
0	1	0	0	x
1	0	x	x	1

From the K-Maps, the driving equations for N2, N1 and N0 are given below which are same as the equations given in the problem specifications. Thus, circuit is a valid circuit.

N2 = Q2 Q1' + Q1' Q0

N1 = Q2

N0 = Q2' Q0' + Q1 Q0'

1.13 Number Representation and Computer Arithmetic (Fixed and Floating point)

1.13.1 Integer (Fixed) Representations

1. The range of unsigned numbers which can be stored in a n-bit register is **0 to 2^n-1**
2. The range of signed numbers which can be stored in an n-bit register is **-(2^{n-1} -1) to + (2^{n-1}-1)** (using sign magnitude representation).

3. The range of signed numbers which can be stored in an n-bit register is **$-(2^{n-1})$ to $+(2^{n-1}-1)$** (using 2's complement representation).
4. The signed 1's complement representation of a number -n is obtained by complementing all the bits of sign magnitude representation of +n. Whereas signed magnitude representation –n is obtained by simply inverting sign bit.
5. Both sign magnitude and signed 1's complement representation has two representations for 0 (+0, and –0).
6. In sign magnitude approach most significant bit is used as sign bit and remaining n-1 bits are used for magnitude.

Thus,

+15 in this approach in a 16 bit register is stored as: 0000000000001111

–15 in this approach in a 16 bit register is stored as: 1000000000001111

7. In 2's complement approach a positive numbers code is same as its sign magnitude version. Whereas, a negative numbers 2's complement is calculated by first finding its magnitude's sign magnitude representation, applying 1's complement for it and adding 1 to it.

■ **Example**

+15 in 2's complement approach in a 16 bit register is: 0000000000001111 (same as sign magnitude version).

-15's 2's complement is calculated by:

Find out +15's sign magnitude:	0000000000001111
1's complement	1111111111110000
Add 1	1
	1111111111110001

8. If a numbers 2's complement is given its most significant bit conveys whether the number is positive or negative. That is, it the msb is 0 positive else negative.

If sign is negative, then to find the magnitude, calculate 1's complement and add 1. Otherwise, magnitude is same as itself.

9. For example find out the number whose 2's complement is 1111111111110001.

■ **Answer:**

By seeing the msb we can conclude that the number is negative.

Magnitude is:	
Calculate 1's complement:	0000000000001110
Add 1	1
	0000000000001111

Therefore the number is : –15

10. The 2's complement can be formed by leaving all least significant 0's and the first 1 unchanged and inverting all remaining bits.
11. Two-s complement representation is having only one representation for zero unlike sign magnitude approach.
12. In two's complement approach there is no need of extra circuit for subtracting a number from the other.
13. The result of two n-bit numbers **addition** may occupy at most **n+1** bits.
14. The result of two n-bit numbers **product** may occupy at most **2n** bits.
15. Advantage of sign magnitude numbers is that a numbers negative value calculation is easy; i.e just invert sign bit.
16. Overflow is said to occur if the result of an arithmetic operation is too large to store; whereas underflow is said to be occurred if the result is too small to be stored. Some systems sends a signal when these things occurs whereas others store closest value. IEEE standard specifies a set of special representation (NaN: Not a Number).
17. While carrying out addition subtraction operations on two's complement numbers the overflow is said to be occurred if carry-in and carry-out are different to sign bit location.
18. In the case of unsigned number addition, an overflow is said to be occurred if carry out is detected at most significant bit position.
19. **Arithmetic overflow** occurs under a number of cases:
    (1) when two positive numbers are added and the result is negative
    (2) when two negative numbers are added and the result is positive
    (3) when a shift operation changes the sign bit of the result.
20. In the case of signed number addition where an negative number is assumed to be in 2's complement form, an overflow at most significant bit position does not indicate overflow.
21. In 2's complement addition/subtraction if we consider carry bit also then the result will be correct even we generally conclude overflow is occurred.
22. In 2's complement overflow only occurs when two numbers of the same sign are added. Similarly, overflow may occur when two numbers of different sign are subtracted.
23. For two n-bit non-negative unsigned numbers, the result of their difference will never be greater than 2^n-1. The danger here is that the result can be less than zero.

For example, 1-2 is calculated as:

1	00000001
-2	11111110
	011111111

Here, carry out is zero. The result conveys as 255, which is not correct.

Thus, we can conclude that for these numbers overflow occurs if carry out is zero.

24. The r's complement of a n-digit number (N) is given as: r^n -N.
25. The r's complement of a n-digit number (N) can be calculated by adding 1 to its (r-1)'s complement.
26. Multiplication of unsigned/signed integers can be carried out with shift-add method whereas division is carried out through shift-subtract.
27. BCD is signed notation. Here, we take a sign bit with usual convention for the sign and for every digit we take 4 bit code. Thus, -27 BCD code 1 0010 0111. Whereas +27 BCD code is 0 0010 0111.
28. In BCD addition, in two situations error occurs. Say 5+6=0101+1011=1011 (a invalid BCD digit). The second is a valid BCD digit but overflow occurs. For example 8+9=1000+1001=1 (carry) 0001.In either case by adding 6 to the result gives correct result.
29. BCD equivalent of one's complement is the nine's complement.
30. An n-bit Ripple carry adder (which does normal binary addition) is formed by cascading n full adders feeding the carry out of I'th full adder as carry in to I+1'th full adder and carry in for first full adder as zero. The computational complexity is Q(n) and size complexity also same.
31. A carry look ahead addition has complexity of O(lg n).
32. Using carry save addition three numbers addition can be done with O(lg n) itself.

1.13.1.1 Booths Algorithm

This algorithm gives a procedure for multiplying binary integers in signed-2's complement form. It operates on the fact that strings of 0's in the multiplier require no addition for the partial product but just shifting, and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as $2^{k+1} - 2^m$. For example, the binary number 001110 (+14) has a string of 1's from 2^3 to 2^1 (k = 3, m = 1). The number can be represented as $2^{k+1} - 2^m = 2^4 - 2^1$=14. Therefore, multiplication of any number M with 14 can be represented as $Mx2^4 - M2^1$. Thus, the product can be obtained by shifting the binary multiplicand four times to the left and subtracting M shifted left once.

While applying the following steps are applied

a. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier.
b. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous 1) in a string of 0's in the multiplier.
c. The partial product does not change when the multiplier bit identical to the previous multiplier bit.

Here, we assume Q, A, B are n-bit registers, Q_{n+1} is previous multipliers bit which is initially set as zero. A sequence counter (SC) is available to iterate the algorithm. Initially Q is assumed to be loaded with multiplier, A is 0's, B as with multiplicand. Finally, result obtained registers A&Q together.

■ **Example**

Take an example –9 × –13 = 10111 × 10011 = +117

B=10111 B′=01001 (subtracting is nothing but adding 2's complement)

Moreover, here all shifts are arithmetic shifts, shifting is done along with sign bit and after shifting sign bit sign will be same as the previous sign.

Q_n	Q_{n+1}		A	Q	Q_{n+1}	SC
		Initial	00000	10011	0	101
1	0	Subtract B	01001			
		(add B′)	01001			
		ashr	00100	11001	1	100
1	1	ashr	00010	01100	1	011
0	1	Add B	10111			
			11001			
		ashr	11100	10110	0	010
0	0	ashr	11110	01011	0	001
1	0	subtract B	01001			
		(add B′)	00111			
		ashr	00011	10101	1	000

Final result is 0001110101 = +117

■ **Example**

–3 × –5 =1101 × 1011 =+15

B =1011 B′=0101

Q =1101

	A	Q	Q_{n+1}	SC
Initial	0000	1101	0	100
Subtract B	0101			
	0101			
ashr	0010	1110	1	011

	A	Q	Q_{n+1}	SC
Add B	1011			
	1101			
ashr	1110	1111	0	010
Subtract B	0101			
	0011			
ashr	0001	1111	1	001
ashr	0000	1111	1	000

Final result is 00001111 = +15 (acceptable)

■ **Example** Perform 5 x -3 using Booth's Algorithm using a 5-bit representation for each operand. Show all your work. Also indicate how the results of the multiplication should be interpreted.

Multiplicand = 5 (00101) Multiplier = -3 (11101)
-5 = (11011)

product:	0 00000	11101 0
subt:	1 11011	11101
shift:	1 11101	11110 1
add:	0 00010	11110
shift:	0 00001	01111 0
subt:	1 11100	01111
shift:	1 11110	00111 1
shift:	1 11111	00011 1
shift:	1 11111	10001

product = 10001 = –15 in two's complement representation
upper 5 bits = 11111 (sign-extended lower 5 bits), therefore, no overflow.

■ **Example** Perform 5 x -4 using a 4-bit representation for each operand. Show all your work. Also indicate how the results of the multiplication should be interpreted.

Multiplicand = 5 (0101) Multiplier = -4 (1100)
-5 = (1011)

product:	0 0000	1100
shift:	0 0000	0110
shift:	0 0000	0011
add:	0 0101	0011
shift:	0 0010	1001
subt:	1 1101	1001
shift:	1 1110	1100

product = 1100
upper 4 bits = 1110 (not sign-extended lower 4 bits), therefore, OVERFLOW.
We would expect overflow here since -20 can't be represented in 4-bits using two's complement representation.

The real product is available in the low 6 bits of the final product register (10 1100) with the upper 2 bits of the final product register being the sign-extended product.

33. Theoretically any combinatorial circuit can be implemented by a ROM configured as a lookup table. For example, product of two 4 bit numbers can be stored in a ROM and their product can be simply read whenever needed. Using this one can calculate product of two integers very efficiently. This lookup tables are used in modern CPU's. The Pentium floating point bug is related to this. Though with this lookup tables multiplications can be done efficiently it needs more ROM. For example for 4-bit numbers products we need 256 bytes and for 8bit number 64Kx16 ROM.
34. Wallace trees are another useful combinatorial circuit used for efficient calculation of products. Though they are more complex than shift add multipliers, they produce result very quickly. Here, carry-save adders and parallel adders are used.

Carry save adder takes 3 (X,Y,Z) inputs and gives results (S) and carry (C). The sum Si is sum of Xi,Yi,Zi and Ci+1 is the carry. To get final product we have to bit vectors S and C.

To use a carry-save adder to perform product operation, we first calculate the partial products of the product then input them to the carry-save adder.

■ **Example**

P = 111
Q = 110
000 Partial Product 0
111 Partial Product 1
111 Partial Product 2

If we consider 5-bit carry-save adder then PP0, PP1, PP2 becomes 00000 (X), 01110 (Y), 11100(Z) respectively. Then the sum vector (S) becomes 011000 and carry (C) become 100010. Thus their addition becomes 101010=42. Note that we incorporate the leading and trailing zeroes to the partial products to align numbers properly.

Similarly, if we assume PP0, PP1, …PP7 are partial products of two 4-bit numbers product calculation then their product can be easily calculated by employing a hierarchy of carry-save adders and finally a parallel adder.

A wallace tree allow two n-bit numbers to be multiplied in $\Theta(\lg n)$ time using a circuit with $\Theta(n^2)$.

1.13.2 Excess–127 code

Excess-127 code is mentioned here because it is required while discussing the IEEE floating point standard. In general, we can consider an excess-M notation for any positive integer M. For an N-bit excess-M representation, the rules for conversion from binary to decimal are:

(1) Evaluate as an unsigned binary number
(2) Subtract M

To convert from decimal to binary, the rules are

(1) Add M
(2) Evaluate as an unsigned binary number

In considering excess notation, here we focus on eight-bit excess-127 notation. The range of values that can be stored is based on the range that can be stored in the plain eight-bit unsigned standard: 0 through 255. Remember that in excess-127 notation, to store an integer N we first form the number N + 127. The limits on the unsigned eight-bit storage require that $0 \le (N + 127) \le 255$, or $-127 \le N \le 128$. As an exercise, we note the eight-bit excess-127 representation of – 5, – 1, 0 and 4.

– 5 + 127 = 122.	Decimal 122 = 0111 1010 binary, the answer.
– 1 + 127 = 126.	Decimal 126 = 0111 1110 binary, the answer.
0 + 127 = 127.	Decimal 127 = 0111 1111 binary, the answer.
4 + 127 = 131	Decimal 131 = 1000 0011 binary, the answer.

1.13.3 Floating Point Numbers

Floating point representation is very similar to scientific notation with sign, fraction (significand or mantissa), and an exponent. Example, -1.232212×10^3. Main drawback of scientific notation is that the numbers can be represented in many ways. For example the above number can be even represented as -12.32212×10^2, or -0.1232212×10^4.
Thus, floating point numbers are normalised such that each numbers significand is a fraction with no leading zeros. However, the following exceptions can be noted.

a. For the number zero as normalisation cannot be done, a special value is assigned.
b. Also positive and negative infinity values are represented specially.
c. Illegal operations are represented as NaN (Not a Number). Say, ∞/∞ or taking the square root of negative number etc.

As each number is stored in normal form, it is implicit that the radix point is located to the left of the most significant bit of the significand and hence does not have to be stored explicitly.

While storing the exponent normally biased exponent is employed. That is some value is added to exponent such that the exponent is always positive and thus there is no need for sign bit for exponent. For example if 4 bits are used for exponent then the range of values one can store is –8 to 7. However, by considering bias value as 8, 8 is added such that exponent value is in between 0 to 15. The arithmetic algorithms will account for this when generating their results.

Precision characterises how precise a floating point value can be. It is defined as the number of bits in the significand. If the significand uses F bits, the precision is 1 part in 2^F. The fraction or significand value will be less than 1.

The gap is the difference between two adjacent values. For example, consider a floating point number with 8-bit significand and the value .10111010 x 2^3. Its adjacent values are .10111011 x 2^3 and .10111011 x 2^3, each of which produce a gap of .00000001 x $2^3 = 2^{-5}$. In general, the gap for floating point value is expressed as $2^{(\text{exponent-precision})}$.

The range of a floating point representation is bounded by smallest and largest possible values. For example, a floating point number with an 8-bit significand (assuming the leading 1 is explicitly stored in the significand register) and 4-bit exponent (-8 to +7) has a range of -.11111111 x 2^7 to $+.11111111 \times 2^7$.

Similarly, in single precision IEEE 754, floating point numbers are represented with an 23-bit significand (assuming the leading 1 is explicitly stored in the significand register) and 8-bit exponent with a bias of 128 can be having a range of $-.111\ 1111\ 1111\ 1111\ 1111\ 1111 \times 2^{127}$ to $+.111\ 1111\ 1111\ 1111\ 1111\ 1111 \times 2^{127}$. The gap is 0.1 x 2^{-128}.
Similarly, 48-bit float point representation with 12-bit exponent, 35-bit significand will have range = $+/- (1-2^{-35})$ x2^{+2047}.

The bias is an excess number added to the exponent so that internally all exponents become positive. As a consequence, the sign of the exponent is removed from being a separate entity. Moreover, when exponents are represented in biased fashion further arithmetic algorithms get simplified.

1.13.3.1 IEEE 754 Floating Point Representation

Number of bits allocated for significand and exponent for single precision and double precision according IEEE standards are given below and this is followed virtually on all CPU's which are having floating point capability.

The significand field also includes an implied 1 to the left of its radix point (exception for special values such as zero, +/- infinity, and de-normalised numbers as shown in the following table).

Normalized significands value in this representation fall in the range of 1<= significand<= 2..

Bias used here is 127 and exponent value for normalised number ranges from –126 to +127 .
The exponent values 00000000(-127) and 11111111(128) are used for special purpose.

■ **Example**
Show how the number 6.25 is stored in IEEE 754 representation in single precision representation.
Binary code of the number is =110.11

Normalized code such that there will 1 before point, i.e $=1.1001 \times 2^2$.
Therefore exponent value = 2 + 127 = 129 = 10000001
Significand = 1001 (as is assumes implied 1 to the left).

sign	exponent	significand
0	10000001	1001 0000 0000 0000 0000 000

■ **Example** How the number 128 can be stored in IEEE 754 single precision representation?

$128 = 2^7 = 10000000 = 1.0 \times 2^7$

Sign bit = 0

Exponent = 7 + 127 = 134 = 10000110

Significand = 0000 0000 0000 0000 000

sign	exponent	significand
0	10000110	0000 0000 0000 0000 0000 000

■ **Example** A real number's IEEE 754 representation is given as: 0100 0000 0110 0000 0000 0000 0000 0000? What is that number?

Sign = + as MSB bit is 0.

Exponent =10000000 -127 =1 (as bias is 127)

Significand bits = 110000000000000=11

Thus, actual Significand =1.11 (1 is implied)

Therefore, number = $1.11 \times 2^1 = 11.1 = 3.5$

■ **Example** Convert the following single-precision IEEE 754 number into a floating-point decimal value.

1 10000001 10110011001100110011010

1. **First, let us put the bits in three groups, sign, exponent and mantissa.**
2. **Now, look at the sign bit. As the sign bit is 1, number is negative number.**
3. **Get the exponent and the correct bias.** The exponent is $10000001_{bin} = 129_{ten}$ Remember that we will have to subtract a bias from this exponent to find the power of 2. Since this is a single-precision number, the bias is 127.
4. **Convert the fraction string into base ten.**

   This is the trickiest step. The binary string represents a fraction, so conversion is a little different.

   $0.10110011001100110011010_{bin} = \mathbf{1*2^{-1}} + 0*2^{-2} + \mathbf{1*2^{-3}} + \mathbf{1*2^{-4}} + 0*2^{-5} + 0 * 2^{-6} + ...$

   The fraction is about 0.7000000476837158.
5. **Now, we can put these numbers in the expression:**

   $(-1)^{\text{sign bit}} * (1+\text{fraction}) * 2^{\text{exponent - bias}}$

   $= (-1)^1 * (1.7000000476837158) * 2^{129-127}$

   $= \mathbf{-6.8}$

   The answer is approximately -6.8.

   Ofcourse, we can also do last two steps in one go as:

   $= (-1)^{\text{sign bit}} * (1.10110011001100110011010) * 2^{129-127} =$

   $= (-1)^{\text{sign bit}} * (1.10110011001100110011010) * 2^2$

   $= (-1)^{\text{sign bit}} * (110.11001100110011010)$ (shifting the binary code by two bits).

Now, we can calculate the number in decimal system.
The smallest positive number among the normalized ones is= 00000001 (sign) and 0000 0000 0000 0000 0000 000 (significand)=1.0×2^{-126}.
The additional assumed 1 in the mantissa (significand) allows additional bit or precision in the representation. But prevents the value 0 from being represented correctly. Thus, IEEE 0 is represented specially. Thus, when exponent field is 0 then leading bit of the mantissa is assumed as 0. This type of numbers are called as denormalised numbers and are used represent smaller nonzero values. The smallest positive representation for a single precision denormalised number is .000 0000 0000 0000 0000 0001 $x2^{-126} = 2^{-149}$.

■ **Example** The following is the IEEE 754 representation of a real number. What is the number?
1000 0000 0100 0000 0000 0000 0000 0000

Sign = - (as MSB bit is one)

Exponent =00000000

Significand=1000 0000 0000 0000 0000 000

As, exponent is 0 and signficand is non zero then the number is denormalised one.

Therefore number = $0.1 \times 2^{-126} = 2^{-127}$.

In summary, IEEE 754 standard:

The IEEE 754 specifies 3 basic formats, called *single*, *double* and *quad* formats.

Type	Single (number of bits)	Double (number of bits)	Quad (number of bits)
S= sign	1	1	1
E = exponent	8	11	15
L = leading bit	1	1	1
M = mantissa	23	52	111
Total width	32	64	128

Sign bit	0 = + 1 = -	0 = + 1 = -	0 = + 1 = -
Maximum E	255	2047	32767
Minimum E	0	0	0
Bias	127	1023	16383
Precision	1 part in 2^{24}	1 part in 2^{53}	1 part in 2^{112}
Range (Denormalized)	$\pm 2^{-149}$ to $(1 - 2^{-23}) \times 2^{-126}$	$\pm 2^{-1074}$ to $(1 - 2^{-52}) \times 2^{-1022}$	
Range (Normalized)	$\pm 2^{-126}$ to $(2\text{-}2^{-23}) \times 2^{127}$	$\pm 2^{-1022}$ to $(2\text{-}2^{-52}) \times 2^{1023}$	
Range(App. Decimal)	$\pm \sim 10^{-44.85}$ to $\sim 10^{38.53}$	$\pm \sim 10^{-323.3}$ to $\sim 10^{308.3}$	

Exponent Field	Fraction Field	Meaning
0	0	0
0	Not 0	+/-(0.fraction) $\times 2^{(1\text{-bias})}$[depending on sign bit]
Not 0, not all 1s	Any	+/–(1.fraction) $\times 2^{(\text{exponent-bias})}$[depending on sign bit]
All 1s	0	+/– infinity [depending on sign bit]
All 1s	Not 0	NaN

Special Numbers in single precision:

+ infinity	**0 11111111 00000000000000000000000**
- infinity	**1 11111111 00000000000000000000000**
NaN	**1 11111111 10000000000000000000000**

If s is the sign bit value, e is biased exponent, f is fraction value then the value of the number is = $(-1)^s 2^{e-127} \times (1.f)$. The most positive normalised number will have exponent as 254 and all significand bits as 1's. The same can be as: $1.1111\ 1111\ 1111\ 1111\ 1111\ 111 \times 2^{127} = (2\text{-}2^{-23}) \times 2^{127}$. Rounding is the process of fitting the results of arithmetic operations results into the significand bits.

Rounding to nearest or unbiased rounding

The Thogoal of this methods goal is to find a representation that is as close as possible to the actual desired value. For example, .1011 1010 to 4 bits yields the value .1100. This has the maximum error of +/– 1/2LSB, one-half of the value of the least significant bit of the rounded result.

Round to even

Here, values are rounded to the closest representable number such that the least significant digit of their result is even. For example,12.5 rounded to 12 and 13.5 rounded to 14.

Round towards 0

The extra bits are simply truncated. For example, .1011 1111 truncates last 4 bits (1111) if we assume significand uses 4 bits.

Round toward + ∞

This can be called as ceiling function. All values are rounded up to the next possible value. This results in negative values being truncated and positive values being rounded up to the next possible value. For example -.1011 1111 becomes -.1011, .1011 0000 rounded to .1011 and .1011 0001 will be rounded to .1100.

Round toward – ∞

This can be called as floor function. This results in negative values being rounded down and positive values truncated. For example -.1011 0001 becomes -.1100, .10110000 rounded to .1011 and .10111111 will be rounded to .1011.

Definition: The number machine epsilon, denoted ε_{mach}, is the distance between 1 and the smallest floating point number greater than 1. For the IEEE double precision floating point standard, $\varepsilon_{mach} = 2^{-52}$.

Be sure to understand many numbers below ε_{mach} are machine representable, even though adding them to 1 may be no effect.

Rounding Rounding is an important concept in scientific computing. Consider a positive decimal number x of 0..... with m digits to the right of the decimal point. One rounds x to n decimal place (n<m) in a manner that depends on the value of the (n+1)-th digit. If this digit is 0,1,2,3,or 4, then the n-th digit is not changed and all the following digits are discarded. If it is a 5,6,7,8 or 9, then the n-th digit is increased by one unit and the remaining digits are discarded. (The situation with 5 as the (n+1)-st digit can be handled in a variety of ways. For example, some choose to round up only when the previous digit is even, assuming that this happens about half time. For simplicity, we always choose to round up in this situation).

Here are some examples of seven-digit numbers being correctly rounded to four digits

$0.1735 \leftarrow 0.1735499$
$1.000 \leftarrow 0.9999500$
$0.4322 \leftarrow 0.4321609$

Note: If x is rounded so that $\tilde{x}$ is the n-digit approximation to it. Then

$$| x - \bar{x} | \leq \frac{1}{2} \times 10^{-n} \qquad (2)$$

In binary also, we have a similar way to do rounding.

IEEE Rounding to Nearest Rule

For double precision, if the 53rd bit to the right of the binary point is 0, then the round down (truncate after the 52nd bit). If the 53rd bit is 1, then round up (add 1 to 52 bit), unless all known bits to the right of the 1 are 0's, in which case 1 is added to bit 52 if and only if bit 52 is 1.

Definition: Denote the IEEE double precision floating point number associated to x, using Rounding to Nearest Rule, by fl(x).

Similar to the remark above, we have $\frac{|fl(x) - x|}{|x|} \leq \frac{1}{2} \varepsilon_{mach}$

1.13.3.2 Machine Representation

In this section will discuss a few more details about how a floating point representation is implemented on a computer. Each double precision floating point number is assigned an 8 byte word, or 64 bits, to store three parts. The sign is stored in the first bit, followed by 11 bits representing the exponent and the 52 bits following the decimal point representing the mantissa.

The sign bit s is 0 for a positive number and 1 for a negative number. The 11 bits represent the exponent come from the positive binary integer number resulting from adding 1023 to the exponent. For exponents between -1022 and 1023, this covers values of these 11 bits from 1 to 2046. The number 1023 is called the **exponent bias** of the double precision. The special exponent 2047 is used to represent infinity if the mantissa bit string is all zeros, and NaN, which stands for Not a Number, if the mantissa bit string is not all zeros. The special exponent 0, is interpreted as the non-normalized floating point form

$$\pm 0.b_1 b_2 b_3 \ldots\ldots b_{52} \times 2^{-1022}$$

These numbers are called subnormal floating point numbers. The smallest representable number in double precision is 2^{-1074}. The subnormal numbers includes + 0 and –0. + 0 has sign 0, exponent all zeros and mantissa 52 zeros. For –0, all is exactly same, except the sign bit is 1.

There are five distinct numerical ranges that single-precision floating-point numbers are **not** able to represent:

1. Negative numbers less than $-(2-2^{-23}) \times 2^{127}$ (*negative overflow*)
2. Negative numbers greater than -2^{-149} (*negative underflow*)
3. Zero
4. Positive numbers less than 2^{-149} (*positive underflow*)
5. Positive numbers greater than $(2 - 2^{-23}) \times 2^{127}$ (*positive overflow*)

Overflow means that values have grown too large for the representation, much in the same way that you can overflow integers. Underflow is a less serious problem because is just denotes a loss of precision, which is guaranteed to be closely approximated by zero.

Not A Number

The value NaN (**Not a Number**) is used to represent a value that does not represent a real number. NaN's are represented by a bit pattern with an exponent of all 1s and a non-zero fraction. There are two categories of NaN: QNaN (*Quiet NaN*) and SNaN (*Signalling NaN*).

A QNaN is a NaN with the most significant fraction bit set. QNaN's propagate freely through most arithmetic operations. These values pop out of an operation when the result is not mathematically defined.

An SNaN is a NaN with the most significant fraction bit clear. It is used to signal an exception when used in operations. SNaN's can be handy to assign to uninitialised variables to trap premature usage.

Semantically, QNaN's denote *indeterminate* operations, while SNaN's denote *invalid* operations.

Note on what happens while doing arithmetic Operations with IEEE 754 numbers

Operations on special numbers are well-defined by IEEE. In the simplest case, any operation with a NaN yields a NaN result. Other operations are as follows:

Operation	Result
n ÷ ± Infinity	0
± Infinity × ± Infinity	± Infinity
± nonzero ÷ 0	± Infinity
Infinity + Infinity	Infinity
± 0 ÷ ± 0	*NaN*
Infinity – Infinity	*NaN*
± Infinity ÷ ± Infinity	*NaN*
± Infinity × 0	*NaN*

Summary

To sum up, the following are the corresponding values for a given representation:

Float Values (b = bias)

Sign	Exponent (e)	Fraction (f)	Value
0	00 .. 00	00..00	+0
0	00 .. 00	00..01 : 11..11	Positive Denormalized Real $0.f \times 2^{(-b+1)}$
0	00..01 : 11 .. 10	XX..XX	Positive Normalized Real $1.f \times 2^{(e-b)}$
0	11 .. 11	00..00	+Infinity
0	11 .. 11	00..01: 01..11	SNaN
0	11 ..11	10..00 : 11..11	QNaN
1	00 .. 00	00..00	-0
1	00 .. 00	00..01: 11..11	Negative Denormalized Real $-0.f \times 2^{(-b+1)}$
1	00 .. 01 : 11..10	XX..XX	Negative Normalized Real $-1.f \times 2^{(e-b)}$
1	11 .. 11	00..00	-Infinity
1	11 .. 11	00..01: 01..11	SNaN
1	11 ..11	10..00 : 11.11	QNaN

1.13.4 Addition of Floating Point Number

Machine addition consists of lining up the decimal points of the two numbers to be added, adding them, and then storing the result again as a floating point number. The addition itself can be done in higher precision (with more than 52 bits) since the addition takes place in a register dedicated just to that purpose.

■ **Example** What will be the magnitude of relative errors in relation to magnitude of the number?

■ **Answer:** Relative error for representing small numbers is going to be high, while for large numbers the relative error is going to be small.

For example, for 256.786, rounding it off to 256.79 accounts for a round-off error of 256.786 – 256.79 = –0.004

The relative error in this case is

$$\varepsilon_t = \frac{-0.004}{256.786} \times 100$$
$$= -0.001558\%$$

For another number, 3.546, rounding it off to 3.55 accounts for the same round-off error of 3.546 – 3.55 = –0.004.

The relative error in this case is

$$\varepsilon_t = \frac{-0.004}{3.546} \times 100$$
$$= -0.11280\%.$$

Why do you need normalisation?

To get uniform relative error (while representing real numbers) independent of the magnitude of the number which is stored.

■ **Example** A machine stores floating-point numbers in a hypothetical 10-bit binary word. It employs the first bit for the sign of the number, the second one for the sign of the exponent, the next four for the exponent, and the last four for the magnitude of the mantissa.

(a) Find how 0.02832 will be represented in the floating-point 10-bit word.

(b) What is the decimal equivalent of the 10-bit word representation of part (a)?

■ **Answer:** (a) For the number, we have the integer part as 0 and the fractional part as 0.02832

Let us first find the binary equivalent of the integer part

$$\text{Integer part}(0)_{10} = (0)_2$$

Now we find the binary equivalent of the fractional part

Fractional part:	$.02832 \times 2$
	0.05664×2
	0.11328×2
	0.22656×2
	0.45312×2
	0.90624×2
	1.81248×2
	1.62496×2
	1.24992×2
	0.49984×2
	0.99968×2
	1.99936

Hence

$$(0.02832)_{10} \cong (0.00000111001)_2$$
$$= (1.11001)_2 \times 2^{-6}$$
$$\cong (1.1100)_2 \times 2^{-6}$$

The binary equivalent of exponent is

$$(6)_{10} = (110)_2$$

So

$$(0.2832)_{10} = (1.1100)_2 \times 2^{-(110)_2}$$
$$= (1.1100)_2 \times 2^{-(0110)_2}$$

The ten-bit representation bit by bit is

0	1	0	1	1	0	1	1	0	0

(b) Converting the above floating point representation from part (a) to base 10 by following gives

$$= (1.1100)_2 \times 2^{-(0110)_2}$$

$$= (1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 0 \times 2^{-4}) \times 2^{-(0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0)}$$

$$= (1.75)_{10} \times 2^{-(6)_{10}}$$

$$= 0.02734375$$

How do you determine the accuracy of a floating-point representation of a number?

■ **Answer:** The machine epsilon, $\in_{mach}$ is a measure of the accuracy of a floating point representation and is found by calculating the difference between 1 and the next number that can be represented. For example, assume a 10-bit hypothetical computer where the first bit is used for the sign of the number, the second bit for the sign of the exponent, the next four bits for the exponent and the next four for the mantissa.

We represent 1 as

0	0	0	0	0	0	0	0	0	0

and the next higher number that can be represented is

0	0	0	0	0	0	0	0	0	1

The difference between the two numbers is

$$(1.0001)_2 \times 2^{(0000)_2} - (1.0000)_2 \times 2^{(0000)_2}$$

$$(0.0001)_2$$

$$(1 \times 1^{-4})_{10}$$

$$(0.0625)_{10}.$$

The machine epsilon is

$$\in_{mach} = 0.0625.$$

The machine epsilon, $\in_{mach}$ is also simply calculated as two to the negative power of the number of bits used for mantissa. As far as determining accuracy, machine epsilon, $\in_{mach}$ is an upper bound of the magnitude of relative error that is created by the approximate representation of a number.

■ **Example** A machine stores floating-point numbers in a hypothetical 10-bit binary word. It employs the first bit for the sign of the number, the second one for the sign of the exponent, the next four for the exponent, and the last four for the magnitude of the mantissa. Confirm that the magnitude of the relative true error that results from approximate representation of 0.02832 in the 10-bit format (as found in previous example) is less than the machine epsilon.

■ **Answer:** From the previous example, the ten-bit representation of 0.02832 bit-by-bit is

0	1	0	1	1	0	1	1	0	0

Converting the above floating point representation to base-10 gives

$$= (0.2734375)_{10}$$

The absolute relative true error between the number 0.02832 and its approximate representation 0.02734375 is,

$$|\varepsilon_t| = \left|\frac{0.02832 - 0.02734375}{0.02832}\right|$$

$$= 0.034472$$

which is less than the machine epsilon for a computer that uses 4 bits for mantissa, that is,

$$\in_{mach} = 2^{-4}$$

$$= 0.0625$$

■ **Example** With x = 0100 0110 1101 1000 0000 0000 0000 0000 and y =1011 1110 1110 0000 0000 0000 0000 0000 representing single precision IEEE 754 floating point numbers, perform $x + y$ and show all work.

0	10111111	0111110 01000000 00000000
1	10111111	0111110 01000000 00000000

Same number but different sign, **the result is 0.**

■ **Example** The IEEE 754 floating point standard specifies 64 bit double precision with a 53 bit significand (including the implied 1) and an 11 bit exponent. IA 32 offers an extended precision option with a 64 bit significand and a 16 bit exponent.

i. Assuming extended precision is similar to single and double precision, what is the bias in the exponent?

■ **Answer:**

$$\mathbf{2^{16-1} - 1 = 32{,}767}$$

ii. What is the range of numbers that can be represented by the extended precision option?

■ **Answer:**

$$2^{16-1} = 32{,}768$$

$$\log_{10} 2^{32768} = \log_{10} a$$

$$32768(0.3) = \log_{10} a$$

$$9820 = \log_{10} a$$

$$a = 10^{9864}$$

Range of representation = $2.0_{10} \times 10^{-9864} \sim 2.0_{10} \times 10^{9864}$

1.13.5 Error Detection and Correction codes

In practice, there can be many reasons for data to get spoiled either during transmission or while it is stored in a storage disk. For example, while transmitting a sudden thunder followed by a giant lightening may disturb our satellite channel resulting bad data. Also, some memory devices

when exposed to high level of X rays or magnetic currents, the data available in them get spoiled. For example, suppose we store the bit sequence 0010 1100 on the disk, but due to a physical defect, what we read later is 0110 1100, which is erroneous. This could be really bad if the software is controlling an airplane, or a medical device, or financial transactions, etc.

Error detection is the process of identifying such a errors while error correction is the process of correcting the same.

In our daily life, we may find regular use of some or other form of numbers such as ISBN number, Visa card number, etc. In order to validate these numbers, there exists some fixed approaches which are akin to error detection methods. Evidently, humans may try to cheat automatic systems which uses the above numbers with pirated or fabricated numbers. Thus, the automatic systems are supposed to verify the authenticity of the numbers.

ISBN Check

Every published book gets assigned a unique ISBN (International Standard Book Number). For example, if a textbook's ISBN is 0-321-38701-5, the 0 means that it was published in the US. The 32 is the code for the publisher (in this case Addison-Wesley). The next 6 digits are the book number assigned by the publisher. The last character is a *check digit.* To verify that that an ISBN is valid, multiply the first digit by 10, the second by 9, the third by 8, … and the last by 1. Add up these products, and divide the sum by 11. That is, (5*1 + 1*2 + 0*3 + 7*4 + 8*5 + 3*6 + 1*7 + 2*8 + 3*9 + 0*10)%11 is zero (Here, we are following C language syntax where % refers to modulus operator). If the remainder is 0, then it is a valid ISBN.

■ **Example** Try to transpose 2 digits from the above – show how the check fails.

IBM Check

The "IBM check", which is used by MasterCard, VISA, and most other credit card companies, is an even/odd weighted code. The digits in the even positions (numbering from the right) are multiplied by 2, then reduced to a single digit (if > 9) by "casting out nines" (subtracting 9, which is equivalent to adding the digits). All digits are then summed and a check digit added to make the result evenly divisible by 10. For example, given the number

6 1 8 2 0 9 2 3 1 5 5 3

the leading 6 is doubled, giving 12, which is then reduced to 3 by adding the digits of 12 together; similarly, the 8 becomes 16 and then 7; the 0 is impervious to doubling; the 2 becomes 4; the 1 becomes 2; and the 5 in the second-last position becomes 10 and thus 1. Thus the check equation is

6 # 2 + 1 + 8 # 2 + 2 + 0 # 2 + 9 + 2 # 2 + 3 + 1 # 2 + 5 + 5 # 2 + 3 mod 10 = 0

where '#' represents multiplication with casting out nines, giving

3 + 1 + 7 + 2 + 0 + 9 + 4+ 3 + 2 + 5 + 1 + 3 mod 10 = 40 mod 10 = 0

This scheme catches all single errors and most adjacent transpositions, but not jump transpositions (such as 553 becoming 355) or 09 becoming 90. This procedure is used to validate MasterCard or VISA card.

Electronic Funds Transfer Routing Number Check

The check digit scheme used on routing numbers for Electronic Funds Transfer (EFT) between banks uses a 9-digit number with position weightings of 3, 7, and 1. The check equation for a number $a_1a_2a_3a_4a_5a_6a_7a_8a_9$ is

$3a_1 + 7a_2 + a_3 + 3a_4 + 7a_5 + a_6 + 3a_7 + 7\ a_8 + a_9 \bmod 10 = 0$

This scheme is based on the fact that multiplication modulo 10 yields a permutation of all 10 decimal digits if the multiplication factor is one of the digits 1, 3, 7, or 9, but only a subset of the decimal digits if the factor is 5 or an even digit, as illustrated in the following table:

Multiplication modulo 10

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	2	3	4	5	6	7	8	9
3	0	3	6	9	2	5	8	1	4	7
7	0	7	4	1	8	5	2	9	6	3
9	0	9	8	7	6	5	4	3	2	1

This scheme cannot detect adjacent transpositions of digits that differ by 5.

Consider an example code as 111000025. Now, we apply the above rule.

3(1 + 0 + 0) + 7(1 + 0 + 2) + (1 + 0 + 5) mod 10 is zero. Thus, the number 111000025 is valid.

Channel Encoding

In order to identify and correct errors in digital communication we add some extra bits (which may be called as parity bits or redundant bits). This process is known as channel

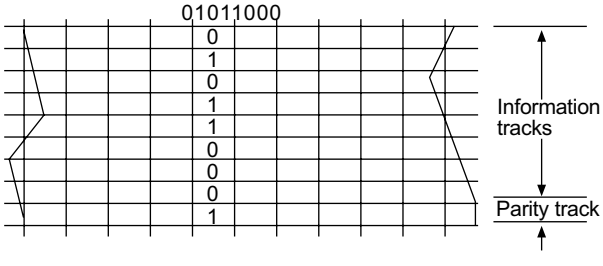


Figure 1.75 Parity bits in magnetic tapes

encoding. Similar bits are also used with data storage also. For instance, in magnetic tapes parity bits are added for every one word. Rather one channel is left for parity data as

shown in Figure 1.75. In Figure 1.75, we have shown parity bit as 1 for the data 01011000. Of course, here we have used even parity approach to calculate parity bit value. In the next pages, we shall be explaining about parity calculation.

- Channel coding allows bit errors introduced by transmission of a modulated signal through a wireless or wire line channel to be either detected or corrected by a decoder at the receiver end.
- The task of channel coding is to represent the source information in a manner that minimises the error probability in decoding.

Do not get confused between Encryption and Channel Coding

Encryption is mainly used to maintain privacy of the data either during communication or when it is stored.

Figure 1.76 demonstrates the use of channel encoding in a practical digital communication system. Detailed explanation of Figure 1.76 is beyond the scope of the book. However, in a nutshell while communicating signals such as voice first it is sampled and quantized with appropriate parameters before compression. The resulting bit sequence encoded and communicated over channels using various modulation methods. At the receiver end, the decoder reverses the operation and generates probably original analogue signal.

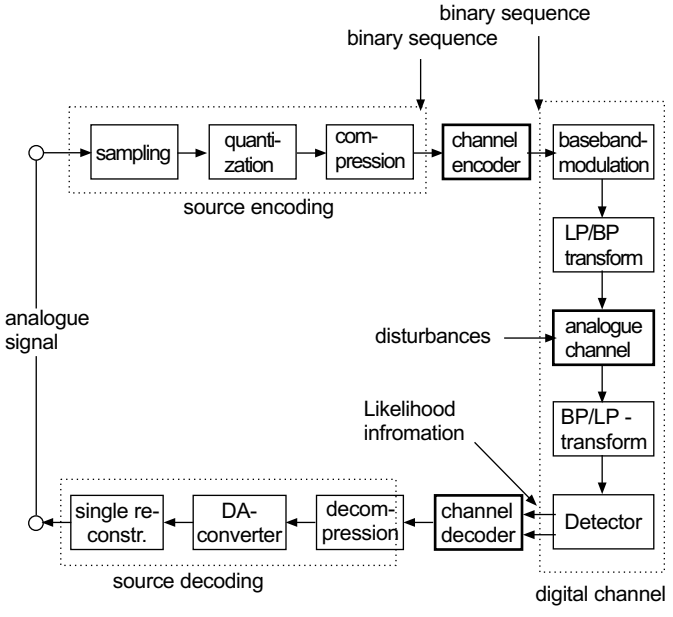


Figure 1.76 Channel Coding

1.13.5.1 Parity Bits

Parity bit is a simplistic approach to identify errors in the binary data. Here, we add a *parity bit* to sequences of data bits. For example, we could add a parity bit to every byte.

Even parity: add a 0 or 1 such that the total number of 1's is even

Odd parity: add a 0 or 1 such that the total number of 1's is odd

We do have some related terms such as mark parity, space parity and none parity. These are very widely used in serial data communication. Mark parity means that the parity bit is always set to the mark signal condition (logical 1) and likewise space parity always sends the parity bit in the space signal condition(logical 0). None parity indicates that no parity bit is included in the data at all. Detailed discussion on parity bits is beyond the scope of the book.

■ **Example** Suppose we want to save the sequence 0010 1100 on the disk. Let's add an even parity bit and write to the disk. Thus, the data on the disk will be 0010 1100 1

Now, if we read the same data from the disk and is read as 0110 1100 1; we know something went wrong at somewhere, because the parity is incorrect (there is an odd number of 1s). That is the idea here is that any single bit error can be detected with this single parity bit.

■ **Example** If we employ odd parity, if the data is 0101010 then the parity bit value should be 0 whereas if the data is 0101101 then the parity bit value should be 1.

Do remember that modern memory (RAM) includes parity bits extensively. We may still refer to bytes, but there are hidden parity bits that are used to make sure that the data that we read from RAM is the same as that which was written into RAM.

What's a weakness of this technique? What if multiple bits change because of noise?

What's a common error that people make when copying long numbers (such as phone numbers or SSNs (social security numbers))?

- transpose digits
- confuse double digits (e.g., 334 gets copied as 344)

How could we detect these kinds of errors? Let us look at an example.

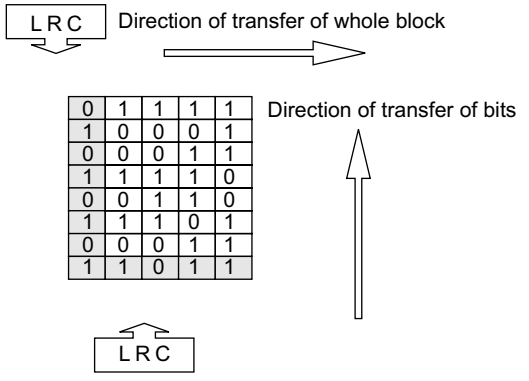
1.13.5.2 LRC(Longitudinal Redundancy Checking) and VRC(Vertical Redundancy Checking)

Single parity bit explained in above examples are used to check errors in single byte. LRC is applied to a "column" of bits within a message. That is, parity is applied to a single byte in the message, whereas LRC is applied to all of them. The LRC is a byte, initially set to 0xFF. Each byte is exclusive-or'ed (XOR) with the LRC and result is stored in LRC. After transmitting the bytes the LRC byte also transmitted. At the receiver also same operation is applied on the received byte including received LRC byte. If the resulting LRC value at the receiver is zero then data transmission

is considered as error free. (see the following example for LRC calculation. Here, even parity technique is employed).

LRC CHARACTER	1	1	1	0	0	0	1
Byte 1	0	1	0	0	1	1	0
Byte 2	1	1	1	0	0	0	1
Byte 3	0	1	0	1	1	1	0
Byte 4	1	1	1	0	0	0	1
Byte 5	0	0	0	0	1	1	0

In fact, error detection can be increased employing LRC and VRC (vertical redundancy checking). LRC is calculated in lateral fashion whereas VRC is calculated in vertical fashion. Rather VRC is nothing but single parity bit per byte as explained above. It is observed that by employing both VRC & LRC error detection increases two to four orders. However, this technique also may fail if an even number of bits changes in the same columns of an even number of bytes. Same columns of an even number of bytes.



1.13.5.3 Cyclic Redundancy Checking (CRC)

CRC algorithms are used to check errors both in memories (storage devices such as Hard disk, RAM) and also in data communication. Essentially, the data for CRC is a very long string of 1's and 0's (called as message); this is divided by a fixed binary string which is known as generator polynomial (see the Table 1.8 for widely used standard polynomials) using modulo-2 arithmetic's and the remainder of this division is called as CRC checksum. This is appended at the end while sending the message and at the receiver again CRC checksum is calculated using the same divisor polynomial. If the checksum is zero then the transmission is said to be error free. Main attraction with this technique is that it can detect single and double bit errors.

While practically calculating CRC checksum, first a series of 0's are appended to the message whose count is equal to the generator polynomial order (see the Table 1.8 for commonly employed polynomials)

Table 1.8 List of CRC polynomials.

	CRC12	CCITT /CRC-ITU or CRC-CCITT	CRC16	CRC32
Checksum size	12 bits	16 bits	16 bits	32 bits
Generator Polynomial	$X^{12} + x^{11} + x^3 + X^2 + x + 1$	$X^{16} + x^{12} + x^5 + 1$	$X^{16} + x^{15} + x^2 + 1$	$X^{32} + x^{26} + x^{23} + X^{22} + x^{16} + x^{12} + x^{11} + x^{10} + X^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

■ **Example** Consider the following example for understanding.

Message: 11100110

Generator Polynomial = 11001 (*i.e.* order is 4)

As mentioned above, we add four 0000s as we are taking polynomial 11001 for division, *i.e.*, $X^4 + x^3 + 1$.

11001)11100110 **0000**(10110110

```
 11001
 01011
  10111
  11001
  11100
  11001
  01010
   10100
   11001
   11010
   11001
   00110
```

Checksum 0110

As mentioned above, we add this checksum to the frame during transmission. Thus, transmitted Frame: 11100110 **0110**

At the receiver side, we simply divide with the same polynomial as worked out below. Of course, we don't add 0s now.

11001)11100110 0110(10110110

```
 11001
 01011
  10111
  11001
  11100
  11001
  01010
   10101
   11001
   11001
   11001
   00000
```

Checksum 0000

As reminder is zero, therefore no errors occurred during transmission.

Do remember that, in Internet communication also we use this technique for error detection.

1.13.5.4 Error Correcting Codes (Block Codes): Hamming Codes

Consider two binary sequences 011101 and 001111 that are selected for communication on a noisy channel. We need to find possibility of error in communication and also what is really sent by the sender. That is, we want to detect and correct the error. For example, if we assume that we have received 011111 and if we assume that at most one bit gets distorted during communication then we can certainly conclude that there is communication error as the received code is not matching with any of the two sequences. Also, we may receive 011111 if 2^{nd} bit gets spoiled in 011101 or 5^{th} bit gets spoiled in 001111. Thus, from the received 011111 we cannot conclude now which is really communicated, either 011101 or 001111. Now, consider that the two sequences which are communicated are 011101 and 001110 and the received sequence is still 011111. If we assume that at most only one bit gets spoiled during communication then we can conclude that the sequence which is sent is 011101 as with it only if one bit gets spoiled we may get 011111. This is not possible with the other sequence 001110. Thus, we are able to correct the communication error. That is, we can conclude that the actual message sent 011101. This way, we can communicate information in practical systems with errors.

We can analyse the above examples as follows with the help of Hamming distance, a similarity measure between two binary sequences. The ***Hamming distance*** is the number of bit positions where two codes (binary strings) of same length differ. If we say that **Hamming distance** between two sequences A and B is 1, then by changing one bit of B we can get sequence A or vice versa. Similarly, we have Hamming weight of a binary sequence or vector is defined as the number 1s in the sequence.

In the first case (or example used above), Hamming distance between 011101 and 001111 is 2 while with the second case hamming distance between 011101 and 001110 is 3. In the first case, hamming distance between received 011111 and the original sequences is same and is 1. Thus, we cannot decide which is really sent by the sender as one bit error with any of the two sequences may lead to 011111. Of course, we are able to conclude that there is an error in communication as received 011111 is not matching with either 011101 or 001111. However, in the case of second case, Hamming distance between received 011111 and 011101 is 1 while 011111 and 001110 is 2. Thus, in the second set we can conclude that the actual sequence which is sent from the sender as 011101 as it is the only one which can lead to 011111 if we assume during communication only one bit gets spoiled. This is the main concept in error correcting using Hamming codes. Imagine what happens if the two sequences which we have sent are having Hamming distance of 1? If we send one sequence, the receiver may receive a sequence of the other because of one bit error in communication. This is very serious situation compared to the others what we have discussed above. In practice, we never want such a situation at all. Thus, in a nutshell if we take two sequences with a minimum hamming distance of three then we can detect and correct the error at the receiver end. This is the theme in Hamming code based error detection and correction.

In a nutshell, the above discussion explains the folliwing point.

1. If we take two binary sequences of m-bits length with Hamming distance of 1 then there is danger of misinterpretation at the received end even single bits gets spoiled during communication. Hamming distance of 1 between them indicates that they differ only at one bit. If that bit gets distorted then we may wrongly conclude that we have received the other message which is not correct and acceptable.
2. If we take two binary sequence of m-bits length with Hamming distance of 2 then any single bit communication error leads to reception of m-bit code which is not matching with either of the codes. Thus, we can conclude that there exists communication error during communication. That is, we can detect the communication error.
3. If we take two binary sequences of m-bits length with Hamming distance of 3 bits then we can detect and correct single bit communication error. We know that Hamming distance of 3 means that the two sequences differs at three bit locations in their codes. Assuming that channel induces single bit errors and if we send one of the two sequences then the sequence what we will receive will be having least Hamming distance with the sequence what we have really sent compared to the other sequence. In no case Hamming distance between the received and code and the two selected codes will be same. Thus, if we have two codes with Hamming distance of 3 then we can detect and correct the single bit error.

Consider a practical situation where we have eight symbols (say A to H) to be stored or communicated. If we think of using the binary sequence numbers 000 to 111 as the codes

to symbols A to H, then there is a un-avoidable danger of one symbol getting misinterpreted as some other symbol even if one bit gets disturbed or spoiled during communication.

We want to have some codes (of more number bits than three bits length) to be assigned for each symbol (A-H) such that even if one (or some) bit gets spoiled during communication yet we can know what is really communicated. Consider the following table having codes for our symbols. In this case, the minimum Hamming distance is 3. This is an important attribute of this set of codes. If one bit is spoiled in any of the symbols code, we don't misinterpret the same as another symbol as it will not be matching with any other codes. Of course, we will be interested in which symbol is really sent. This can be decided by calculating Hamming distance with all the symbols and the received symbol. Then, we classify the received symbol as the symbol for which distance is minimum.

Symbol	Code
A	000000
B	001111
C	010011
D	011100
E	100110
F	101001
G	110101
H	111010

Suppose we receive the code 111100, what symbol does it represent which is transmitted originally? Well, first we calculate it's Hamming distance from the original codes of all the symbols as shown below:

Symbol	Code	Distance
A	000000	4
B	001111	4
C	010011	5
D	011100	1
E	100110	3
F	101001	3
G	110101	2
H	111010	2

In this case, the nearest code is the one for the letter D. Because we know that the smallest Hamming distance between any 2 correct codes is 3, we can assume that that the code that was transmitted was 011100.

Thus, the main theme behind block (Hamming) coding is to add some redundant bits to data bits and communicate (Figure 1.77). That is, if we assume source sends m-bit messages then we will be having 2^m possible messages. We map these messages into another set of messages of larger length say n-bits where n is greater than m. That is, we know in n-bits we will be really having 2^n possible messages, however only 2^m of them are sent by the channel encoder. If we happened to select these 2^m messages of length n-bits such that they are sufficiently apart in terms of Hamming distance. When an error occurs during communication, we can detect and correct the same as explained above. This is explained as below. However, most important aspect here is selecting 2^m code words (valid code words) of length n-bits!.

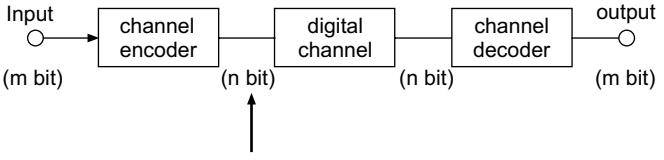


Figure 1.77 The main theme of Hamming Coding

According to Hamming coding theory, if we assume that the number of data bits are m, number of redundant (can be called as parity bits) bits are p, then to detect and correct single bit errors we have to select p such that the following condition has to be satisfied.

$m + p + 1 <= 2^p$

We are sure readers have understood our emphasis of selecting the codes having minimum hamming distance of 3 to detect and correct one bit error. However, most important aspect which we have not emphasised is: How could we build one of these codes? Let's say that we wanted to transmit 4 bits and we wanted to be sure that we could correct every 1 bit error that occurs.

From the above theorem we can find that it takes 3 parity bits to ensure that we detect 1-bit errors in 4 bits of data. However, as we increase the number of parity bits, the number of data bits we can encode increases quickly as shown in the following table.

Number of Parity Bits	Number of Data Bits
4	11
5	26
6	57
7	120
8	247
9	502
10	1013

Note that the above table contains number of parity bits that can only correct 1-bit errors. How it could be possible to correct more number of bit errors? Answer is to develop a code with larger Hamming distance between the code values. For example, if the minimum Hamming distance is 5, then we can correct 2 bit errors.

■ **Example** Calculate minimum Hamming distance for the following code vectors and comment on the same

Index	codewords					Index	codewords				
1	1	1	0	0	0	6	0	1	0	1	0
2	1	0	1	0	0	7	0	1	0	0	1
3	1	0	0	1	0	8	0	0	1	1	0
4	1	0	0	0	1	9	0	0	1	0	1
5	0	1	1	0	0	10	0	0	0	1	1

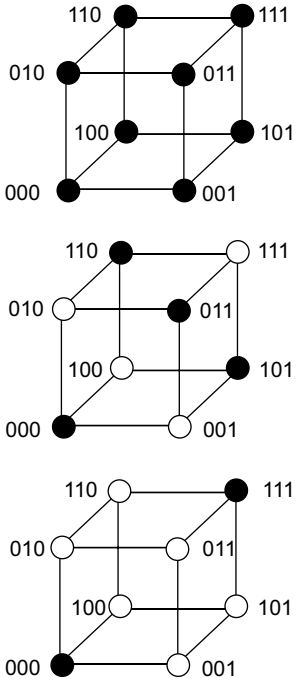
■ **Answer:**

See the following table which contains the Hamming distance between each of the code vectors. We may find that the minimum Hamming distance between the code vectors is 2. Thus, we can only detect single bit errors.

Index	1	2	3	4	5	6	7	8	9	10
1	0	2	2	2	2	2	2	4	4	4
2	2	0	2	2	2	4	4	2	2	4
3	2	2	0	2	4	2	4	2	4	2
4	2	2	2	0	4	4	2	4	2	4
5	2	2	4	4	0	2	2	2	2	4
6	2	4	2	4	2	0	2	2	4	2
7	2	4	4	2	2	2	0	4	2	2
8	4	2	2	4	2	2	4	0	2	2
9	4	2	4	2	2	4	2	2	0	2
10	4	4	2	2	4	2	2	2	2	0

→ Minimum (Hamming) distance: h_{min} =2

■ **Example** See the following figure which contains code vectors that are shown as nodes of a cube. Calculate h_{min} and comment on the error detection and correction abilities of the codes.



■ **Answer :**

First Case:

h_{min} value is 1. Thus, we cannot detect or correct errors even of single bit.

Second Case:

h_{min} value is 2. Thus, we can detect single bit error but cannot correct single bit errors.

Third Case:

h_{min} value is 3. Thus, we can detect and correct single bit errors.

Calculating the Hamming Code (Bit Stuffing)

This Hamming coding is also popularly called as parity bit stuffing. This is also akin to the method used above; however it is little more convenient for computer implementation. Here, m message bits are stuffed with r check bits and frame is prepared for communication. In the frame, bits are numbered from 1 at the left and bits 1,2,4,8 etc (which are powers of 2) are called as parity or check bits and bits 3,5,6,7, etc are the data bits. In order to calculate hamming bits expressing all bits positions whose value is one in binary number and XOR operation is applied for them. At the receiver end, to determine bit position where error has occurred, extract the Hamming bits and XOR them with the binary code for each data bit positions that contain a 1. The resulting binary code indicates the bit position where error is occurred and to correct simply flip the bit. This method is used only to correct single bit errors.

■ **Example** Assume message bits: 010101011101

		0		1	0	1		0	1	0	1	1	1	0		1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

The above figure contains data bits of the message in their respective cells of the frame while Hamming bits in cells 1, 2, 4, 8 and 16 are empty. To calculate Hamming bit values, we simply apply XOR for the cell indexes of the data bits whose values are 1s as shown below.

Bit Position where data bit value is 1	**XOR operation**
	00101
	00111
5	00010
7	01010
	01000
10	01100
	00100
12	01101
	01001
13	01110
	00111
14	10001
	10110 Hamming bits (last 0 goes to
17	first cell as first Hamming bit and vice versa).

Frame stuffed with Hamming bits that can detect and correct single bit errors is given below.

0	1	0	1	1	0	1	0	0	1	0	1	1	1	0	1	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

■ **Example** If our 4 bit sequence is 1011, then find out the hamming code after adding 3 parity bits to recover from 1 bit errors.

■ **Answer:** 1, 2, 4 are party bits while data bits with value 1 are: 3,6, 7. By applying XOR for 3, 6 and 7, we calculate parity bits:

011
110
101(XOR result)
111
010(XOR result)

Position 1 parity is: 0

Position 2 parity is: 1

Position 4 parity is: 0

This gives us a resulting Hamming code as: **011**0011.

■ **Example** Calculate Hamming code for a byte of data: 10011010

■ **Answer:** Indexes of the data bits whose values are 1 = 3,7,9,11

We may need 4 parity bits. To find out the parity bits, we apply XOR operation for 3, 7, 9, and 11 as shown below.

0011
0111
0100(XOR result)
1001
1101(XOR result)
1011
0110(XOR result)

Thus, final Hamming code word parity bits stuffed becomes: **011**1001**0**1010.

We advise readers to visit http://candle.ctit.utwente.nl/wp5/tel-sys/exercises/datalinkp2p/hamming74demo.html and experiment the available online Hamming code demo.

At the receiver, all bit position values there is 1 are XORed (data bit only) along with the Hamming code that is received and this result is called the (SYNDROME). The syndrome value represent one of the following situations:

1. If the syndrome is zero, no error has been detected.
2. If the syndrome contains one and only one bit set to 1, then an error has occurred in one of the check bits, no correction is required.
3. If the syndrome contains more than one bit set to 1 then the numerical value of the syndrome indicates the position of the data bit in error. This data bits is inverted for correction.

First consider an n block for data 001110010 using Hamming code encoding.

8 bit data require 4 check bit and distributed as follows:

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Distribution	D8	D7	D6	D5	C4	D4	D3	D2	C3	D1	C2	C1
	0	0	1	1		1	0	0		1		

Then D6, D5, D4, and D1 position number is XORed to find check bits.

Check bits = D6 XOR D5 XOR D4 XOR D1

= 1010 XOR 1001 XOR 0111 XOR 0011 = 0111

$C_1 = 1, C_2 = 1, C_3 = 1, C_4 = 0$

Now we assume that for the above example the received message is 001101101111. We find the syndrome and correct message if possible. As usual, we identify data bits and their cell indexes as shown below.

D8	**D7**	**D6**	**D5**	**C4**	**D4**	**D3**	**D2**	**C3**	**D1**	**C2**	**C1**
0	0	1	1	0	1	1	0	1	1	1	1
1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001

Thus,

SYNDROME = XOR(D6, D5, D4, D3, D1, 0111)

= XOR (1010, 1001, 0111, 0110, 0011, 0111) = 0110

Then this indicates that bit position (0110) which is 6^{th} in error and correct message is

001101001111 removing check bits

00111001 = data

Single error correcting, double error detecting code

If an overall parity check is included at position 0, then the Hamming code word extended by this bit becomes a *single error correcting, double error detecting* code. The following 4 cases cover all possibilities for 2 or fewer errors:

1. no parity error, no Hamming error ⇒ no error detected
2. no parity error, Hamming error ⇒ double error detected
3. parity error, no Hamming error ⇒ parity bit in error
4. parity error, Hamming error ⇒ correctable error detected

Notice that

- If there are no errors, there are no parity errors for any of the checks and no error correction is needed. This is the "*no parity error, no Hamming error*" case.
- If 2 bits are in error in the overall code word, then the overall parity will be unaffected; ie., the overall parity check will find no error. On the other hand, since at least one of the errant bits is in the Hamming code word, the Hamming parity checks will flag an error. This is the "*no parity error, Hamming error*" case, and flags occurrence of a double error. In this case error correction no longer applies, since there is no way to determine which 2 bits are in error, even if one of them happens to be the parity bit, but the double error has been detected.
- If a single bit is in error then an overall parity error will be flagged. If the bit is the parity bit, then the Hamming code word generates no errors. This is the "*parity error, no Hamming error*" case, and the parity error can be corrected by changing the parity bit (so single error correction remains in effect).
- If a single bit is in error and it is in the Hamming code word, then the Hamming parity checks locate the position of the bit. This is the "*parity error, Hamming error*" case, and the error can be corrected using the Hamming decoding technique.

This covers all possibilities of 0, 1, or 2 errors being present. If more than two errors are present, one of these cases will occur, but the result will be erroneous.

In general, Hamming code is designed to correct single bit error. However, table below list the number of check or Hamming bits required for various data lengths for single error correction, and double error correction.

Data bits	Check bits (single error correction)	Check bits (single error correction) (double error correction)
8	4	5
16	5	6
32	6	7
64	7	8
128	8	9
256	9	10

1.13.6 A Note on Weighted and non-Weighted codes

The binary code, hexadecimal code, etc., are called as weighted codes as for each digit there will be an associated positional value. We advise readers to refresh first unit of this chapter where we have represented a number as a polynomial with some weights. If the weights are positives then the number is said to be positively weighted code number else negatively weighted code number. Those number which does not have no such weights associated with each digit are called as non-weighted codes. Gray code, Excess-3 code are the best examples of this category.

Sequential Code is the one in which each successive number will be having a difference of one such that mathematical manipulations becomes easy.

Self Complimenting Code is the in which 9s complement of a number N (i.e 9-N) can be achieved by simply complementing bits of N. Conventionall binary code (8421) is not self complimenting. However, Excess-3 code is self complimenting. For example, 3's binary code in Excess-3 is 0110 while 6 is 1001. Nines complement of 3, i.e 9-3 = 6, whose Excess-3 code is 1001 which we can achieve by complementing 0110. Thus, Excess-3 code is self complimenting.

Cyclic codes are the ones in which each successive code word differs at one bit position. Good example of this category is gray code. They are also called as reflected codes. Digital systems are usually designed to process data in discrete form only. Many physical systems supply continous data. These data should be conveted into digital or discrete form before applying to digital system. Here, these codes are very much used.

1.13.7 A Note on Gray Codes

Gray codes are cyclic and self complimenting type which were used in electromechanical devices. Nowadays, they are used in many application areas such as terrestrial communication, TV communication, Karnaugh maps, Genetic algorithms, Neural Networks, etc. Evidently, successive codes differs in single bit position. The following table contains 4-bit gray codes. Of course, to verify the self-complementary property, we can neglect MSB bit.

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111

Decimal	Binary	Gray
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

To convert binary code to gray code, we can take MSB of binary code as MSB of gray code and subsequent bits starting from MSB to LSB can be found by XORing successive bits and retaining the subsequent bits of resultant gray code. For example: 010s gray code can be found as: $0 \quad 0 \oplus 1 \quad 1 \oplus 0$. That is, 011. We can get in the same way binary code given the gray code. The following Figure 1.78 summarises this operation pictorially.

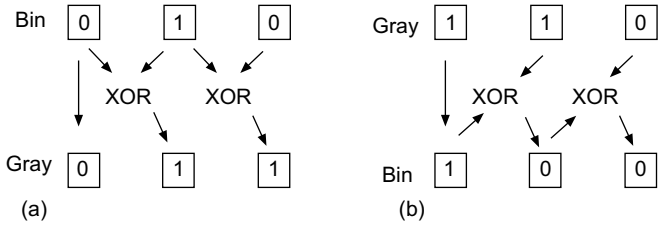


Figure 1.78 Binary to gray and gray to binary conversion of a 3 bits number

To get gray code, we use the following recursive approach. To get n + 1 bit gray code from n-bit gray code,

1. prefix n-bit representation with 0
2. append to this n-bit representation in reverse order prefixed by 1.

For example, see the following work out having details about calculating 2 bit gray code for 1 bit, 3 bit gray code from 2 bit gray code and vice versa.

1-bit	2-bit	3-bit	4-bit
0	00	000	0000
1	01	001	0001
	11	011	0011
	10	010	0010
		110	0110
		111	0111
		101	0101
		100	0100
			1100

1-bit	2-bit	3-bit	4-bit
			1101
			1111
			1110
			1010
			1011
			1001
			1000

■ **Example** If we assume base of a number system as 1 what are possible digits?
Or
Why we cannot consider base of a number system as 1?

■ **Answer:** If base of a number system is 1 then possible symbols or digits are only 0. Thus, we can represent 0 only in this system.

■ **Example** Give an example of communication where error correcting code works better than error detection and an example, where neither error detection nor error correction is needed.

■ **Answer:** Error correcting code works better in the case of communication between a space shuttle and its land control station. Due to long distance, the error rates are high and so is the end-to-end latency. Data retransmission can be avoided by including error-correcting codes.

In interactive continuous media application, the maximum acceptable end-to-end latency is in the order of tens of milliseconds. Data retransmission is ruled out, as late data is useless in audio/video applications. Since, human eye and ear have a "built-in" error correction mechanism small gaps in data stream would not be noticeable thus eliminating the need for error correcting codes.

■ **Example** Assuming that the messages are required to be sent to receiver:

0	**0000**	**1**	**0001**	**2**	**0010**	**3**	**0011**
4	0100	5	0101	6	0110	7	0111
8	1000	9	1001	A	1010	B	1011
C	1100	D	1101	E	1110	F	1111

There are 16 different messages. These 4-bits messages are mapped to the following Sixteen Valid Code words

0	**0000000**	**8**	**1001011**
1	0000111	**9**	1001100
2	0011001	**A**	1010010
3	0011110	**B**	1010101
4	0101010	**C**	1100001
5	0101101	**D**	1100110
6	0110011	**E**	1111000
7	0110100	**F**	1111111

Verify whether these 16 code words are sufficient to detect and correct 1 bit errors or not?

■ **Answer:**
If we observe, minimum Hamming distance between any two words is 3. Thus, it can correct and detect single bit errors. Also, it satisfies Hamming condition. Checking the hamming equation:
Data bits d=4, parity bits p=3 and
4+3+1 is less than 23. Thus, the code words are perfect.

■ **Example** Assume that a communication system used the above 7 bit Hamming codes to send 4 bit data. Assuming that the received code word is 0000001, calculate the Hamming distance between this received code word and all the code words and then recover the actual code word sent along with the data bits sent.
The following table contains the Hamming distance between each of the code word and 0000001.

Code word	Hamming Distance	Code Word	Hamming Distance
0 0 0 0 0 0 0	**1**	1 0 0 1 0 1 1	3
0 0 0 0 1 1 1	2	1 0 0 1 1 0 0	4
0 0 1 1 0 0 1	2	1 0 1 0 0 1 0	4
0 0 1 1 1 1 0	5	1 0 1 0 1 0 1	3
0 1 0 1 0 1 0	4	1 1 0 0 0 0 1	2
0 1 0 1 1 0 1	3	1 1 0 0 1 1 0	5
0 1 1 0 0 1 1	3	1 1 1 1 0 0 0	5
0 1 1 0 1 0 0	4	1 1 1 1 1 1 1	6

As the minimum Hamming distance is 1 with 0000000, the receiver concludes that the actual transmitted code is 0000000, which is the correct. The actual data bits are 0000.

■ **Example** Is 8421 code is self complimentary? Explain by taking 3 as an example.

■ **Answer:** The 8421 is same as normal binary code in which weights are 8,4, 2, 1 from MSB to LSB. In this code, 3 can be represented as 0011. Number 6, which is 3s complement (9-3) code is 0110. As 1s complement of 0011 is not 0110. This system is not self-complimenting type.

■ **Example** Is 4221 code is self complimentary?. Explain by taking 3 as an example.

■ **Answer:** Code of 3 in 4221 representation is 0011. Number 6, which is 3s complement (9-3) in 4221 is 1100 which is same as 1s complement of 0011, we can say that 4221 code is self complimenting.

■ **Example** Prove 7421 code is not self complimentary type.

■ **Example** What is $84\overline{21}$ coding?. What are the weights for this?. Explain. Is it self-complimenting type?.

■ **Answer:** In this representation, first two bits weights are separately dealt. If these bits values are ones, respective digit weights are subtracted. For example, number 9s code in this representation is 1111 which we can verify with by expanding: 8 + 4 – 2 – 1 = 9.
This code is self complimentary. For example, 3s code in this representation is 0101. Number 6 is 3s complement (9–3) whose representation in this code is 1010 which we get by complimenting 0101. Thus, this code is self-complimenting type.

■ **Example** A communication channel is reported to be having error rate of 1 in 10^6 bits. Find out the probability of a bit getting spoiled out of 8 bits.

■ **Answer:** The error rate 1 in 10^6 bits indicates that if we send a data unit of size 1000000 bits then certainly 1 bit gets spoiled. If we assume that this 1000000 bits data unit is split into 8-bit frames only one frame gets spoiled remaining are not. Thus, probability of a bit getting spoiled out if 8-bits = 8/1000000.

■ **Example** Assuming that 5-bit code words are proposed such that they exactly contains two 1s. During communication if any bit gets spoiled, we can detect the error in communication by simply counting the number of 1s in the received data. Assume that the following codes are proposed for each of the 10 digits. Comment on the error detection procedure.

Digit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Code	00011	00101	01001	10001	00110	01010	10010	01100	10100	11000

■ **Answer:** It can detect single bit errors and also multiple errors in adjacent bits.

■ **Example** Consider the following code C:
{0101010, 1001100, 0011001, 1110000, 0100101, 1000011, 0010110 }
What is the Hamming distance of code C?. Which types of error can code C correct? . What code word can you add into the code C yet retaining the same Hamming distance?

■ **Answer:**
Let us find out minimum hamming distance for the code.

	0101010	1001100	0011001	1110000	0100101	1000011	0010110
0101010	0	4	4	4	4	4	4
1001100		0	4	4	4	4	4
0011001			0	4	4	4	4
1110000				0	4	4	4
0100101					0	4	4
1000011						0	4
0010110							0

Minimum Hamming distance of the above code is:4 (lowest value of the above matrix)
Thus, we can correct 1 error.
If we add 1111111 to the code, the Hamming distance between other code vectors can be given as: 4. If we observe all the code vectors, we find four zeros in each of them thus. Thus, Hamming distance between 1111111 and each of the code vectors becomes 4. Therefore, minimum Hamming distance of the code will still be 4.

■ **Example** Recover the correct data word from the received 7-bit code word 0111001, given that odd parity is used.

■ **Answer:** We know 1st, 2nd, and 4th bits are parity bits in the received data(0111001).
Now, we expand the indexes (3, 7) of the data bits which are ones (0111001). That is,

1 + 2

1 + 2 + 4

As we are using odd parity, according to received data parity bits should be 110. However, received parity bits are 011. That is, at 1st and 4th bits parity values are not matching. Thus, we can conclude that 5th bit(1 + 4) is the one spoiled bit. Therefore, corrected data frame can be 1110101 in which actual data is 1101.

■ **Example** For 01010101_2 give the parity bit that will produce odd parity

■ **Answer:** 1
There is an even number of 1s in the data word, so that the parity bit must be 1 to obtain an odd weight for the complete code work (data word plus parity bit)

■ **Example** For 01010101_2 give the parity bit that will produce even parity.

■ **Answer:** 0
There is an even number of 1s in the data word, so that the parity bit must be 0 to maintain an even weight for the complete code work (data word plus parity bit)

■ **Example** For 111111_2 give the parity bit that will produce odd parity

■ **Answer:** 1
There is an even number of 1s in the data word, so that the parity bit must be 1 to obtain an odd weight for the complete code work (data word plus parity bit)

■ **Example** Find the transmitted message for 00110101,01101010111001 1,01010101 and 0101010101010101

■ **Example** Find corrected message for each of the following.
001000110111,101010111110,010111111100001101011

■ **Example** Explain the use of most significant bit as a sign bit.

■ **Answer:** MSB is used to indicate the sign of an integer number. In sign-magnitude approach, MSB 0 indicates it is a positive integer, otherwise it is negative integer. In 2s complement approach also, sign bit is used.

■ **Example** In the following table, we have gives different coding schemes A, B and C for 0-7. Which is gray code? Explain why?

Value	Code A	Code B	Code C
0	000	111	000
1	001	011	001
2	011	010	011
3	010	110	010
4	100	100	011
5	101	000	111
6	111	001	101
7	110	101	100

■ **Answer:** Code A is not valid. Two bits change from 3 → 4. Code B is Valid. We may find only one bit change between any two adjacent codes including 7 to 0 also. Code C is also not valid as the code "011" appears twice.

1.14 Instruction Set Architecture

A machine language or assembly language programmer views the machine at the level of abstraction provided by the ISA. The ISA defines the personality of a processor and

indirectly influences the overall system design. It specifies how a processor functions: what instructions it executes and what interpretation is given to these instructions.

One of the characteristics that influence the ISA is the number of addresses used in the instructions. Since typical operations require two operands, we need three addresses: two source addresses to specify the two input operands and a destination address to indicate where the result should be stored. Most processors specify three addresses. We can reduce the number of addresses to two by using one address to specify a source address as well as the destination address. The Pentium uses two-address format instructions. It is also possible to have instructions that use one or even zero address. The one-address machines are called accumulator machines and the zero-address machines are called stack machines.

The number of addresses used in instructions partly influences the number of data registers and their use. For example, stack machines do not require any data registers. However, as noted, part of the stack is kept internal to the processor. This part of the stack serves the same purpose that registers do. In three- and two-address machines, there is no need for the internal data registers. However, as we have demonstrated before, having some internal registers improves performance by cutting down the number of memory accesses.

1.14.1 Design Decisions for Instruction Sets

- Short Instructions are typically better.
- Instructions of a fixed length are easier to decode but waste space.
- Memory organisation affects instruction format.
- How many different types of addressing modes we want?

■ **Example** Implement the following instruction in three address, two address, one address and zero address machines. Give number of memory operations with each of them.

A = B + C * D – E + F + A

Three address implementation (T is assumed as temporary variable):

mult T, C, D ; T = C*D
add T, T, B ; T = B + C*D
sub T, T, E ; T = B + C*D – E
add T, T, F ; T = B + C*D – E + F
add A, T, A ; A = B + C*D – E + F + A

Two address implementation (T is assumed as temporary variable):

load T, C ; T = C
mult T, D ; T = C*D
add T, B ; T = B + C*D
sub T, E ; T = B + C*D – E
add T, F ; T = B + C*D – E + F
add A, T ; A = B + C*D – E + F + A

One address or accumulator machine implementation:

load C ; load C into the accumulator
mult D ; accumulator = C*D
add B ; accumulator = C*D + B
sub E ; accumulator = C*D + B – E
add F ; accumulator = C*D + B – E + F
add A ; accumulator = C*D + B – E + F + A
store A ; store the accumulator contents in A

Stack machine implementation:

push E ; <E>
push C ; <C, E>
push D ; <D, C, E>
mult ; <C*D, E>
push B ; <B, C*D, E>
add ; <B + C*D, E>
sub ; <B + C*D – E>
push F ; <F, B + D*C – E>
add ; <F + B + D*C – E>
push A ; <A, F + B + D*C – E>
add ; <A + F + B + D*C – E>
pop A ; < >

In the three-address machine, each instruction takes four memory accesses: one access to read the instruction itself, two for getting the two input operands, and a final one to write the result back in memory. Since there are five instructions, this machine generates a total of 20 memory accesses.

In the two-address machine, each arithmetic instruction still takes four accesses as in the three-address machine. Remember that we are using one address to double as a source and destination address. Thus, the five arithmetic instructions require 20 memory accesses. In addition, we have the load instruction that requires three accesses. Thus, it takes a total of 23 memory accesses.

The count for the accumulator machine is better as the accumulator is a register and reading or writing to it, therefore, does not require a memory access. In this machine, each instruction requires just two accesses. Since there are seven instructions, this machine generates 14 memory accesses.

Finally, if we assume that the stack depth is sufficiently large so that all our push and pop operations do not exceed this value, the stack machine takes 19 accesses. This count

is obtained by noting that each push or pop instruction takes two memory accesses, whereas the five arithmetic instructions take one memory access each.

This comparison leads us to believe that the accumulator machine is the fastest. The comparison between the accumulator and stack machines is fair because both machines assume the presence of registers. However, we cannot say the same for the other two machines. In particular, in our calculation, we assumed that there are no registers on the three- and two-address machines. If we assume that these two machines have a single register to hold the temporary T, the count for the three-address machine comes down to 12 memory accesses. The corresponding number for the two-address machine is 13 memory accesses. As you can see from this simple example, we tend to increase the number of memory accesses as we reduce the number of addresses.

1.15 Addressing Modes

Instruction consists of the op-code that tells the process what instruction to perform and, the operand or address field which tells the processor where to find that data to be operated upon. This address is known as the Effective Address (EA).

To determine the EA, the processor uses one of a number of addressing modes that are defined by the operand field of the instruction. Getting the EA from the addressing mode may be quite simple (e.g., the operand is [the contents of] a data register) or complex (e.g., the operand is in memory, the address of which is contained in an address register).An assembly language instruction may use one of the following addressing modes to generate this address.

The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced. Variety of addressing modes are supported on or both of the following reasons.

a. To give programming versatility to the user by providing such facilities as pointer to memory, counters for loop control, indexing of data, especially when manipulating data structures such as arrays, and for program relocation. For example, in 8086 the Base-indexed addressing mode lets you set BX and SI (registers) to the row and column offsets, respectively, of any element in the table such that the elements can be accessed easily.
b. To reduce the number of bits in the addressing field of the instruction.

In some computers the addressing mode of the instruction is specified with a distinct binary code, just like operation code. Other computers use a single binary code that designates both the operation and the addressing mode of the operand.

The following are the commonly used addressing modes. A microprocessor's instruction set use some or all of these modes, depending on its design. In these examples, we have used LDAC instruction; the LDAC instruction loads data from memory into the microprocessor's AC (accumulator) register.

Direct Mode

In this mode, the instruction includes a memory address(s) as operands; the CPU accesses that location in memory. For example, the instruction LDAC 5 instruction reads the data from memory location, 5 and stores the data in the CPU's accumulator. This mode is typically used to load operands and values of variables into the CPU.

Indirect Mode

Indirect mode starts off very much like direct mode, but then performs a second memory access. The address specified in the instruction is not the address of the operand; it is the address of a memory location that contains the address of the operand. For example, the LDAC 5 instruction in indirect mode, often denoted as LDAC @5 first retrieves the contents of location 5, say 10. Then the data into the CPU. This addressing mode is used by compilers and operating systems with re-locatable code and data.

Register Direct & Register Indirect Modes

Register modes work the same as direct and indirect modes, except they do not specify a memory address; instead, they specify a register as operand. If register R contains the value 5, the LDAC R instruction copies value 5 from register R into the CPU's accumulator (AC). The register indirect instruction, LDAC @R, performs just as the as the above mode. Here, we assume that the Register contains memory address. Thus, the instruction copies the value (address) from the register R to AC. By doing now a memory access, actual operand is read from the memory. Here the indirection comes from reading the address from the register instead of the first memory access. The advantage of register indirect mode instruction is that the address field of the instruction uses fewer bits to select a register than would have been required to specify a memory location.

Autoincrement or Autodecrement Mode

This is similar to the register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory. This helps to access table data in memory efficiently.

Immediate Mode

The immediate mode, the operand specified is not an address; it is actual data to be used in the instruction. The instruction LDAC #5 moves the data 5 into the accumulator. Normally, literals are used in this fashion.

Implicit Mode

Implicit mode does not explicitly specify an operand; the instruction implicitly specifies the operand because it always applies to a specific Register. This isn't usually used for load instructions. It is more commonly found in such instructions as CLAC, which clears the accumulator (sets its value to zero). Similarly, some instructions such as ISZ (increment and skip if zero), CMP (Complement) No operand is loaded because this instruction always refers to the accumulator. This mode is used in CPUs that use a stack to store data. They do not have to specify an operand, since it is implicit that the operand must come from the stack. These instructions which runs on the stack are also called as zero-address instructions since the operands are not explicitly available in the instructions. Rather they are implied to be on top of the stack.

Relative Mode

This mode, the operand supplied is an offset, not the actual address. It is added to the contents of the CPU's program counter register to generate the required address. The program counter contains Address of the current instruction being executed, so the same relative instruction will produce different addresses at different locations in the program. The address part of the instruction can be a signed number. When this is added to PC, the result produces EA whose position in memory is relative to the address of the next instruction.

The relative mode instruction LDAC $5 is located at memory location 10 (and assume PC value is 10), and it takes up two memory locations, the next instruction is at location 12. The instruction reads data from location (12 + 5 =) 17 and stores it in the accumulator. This mode is particularly useful for short jumps and re-locatable code.

To further clarify, let PC value is 825 and the address part of the instruction is 24. The instruction at location 825 is read during the fetch phase and the PC is then incremented to 826. The EA for the relative address mode is 826+40=850. This is 24 memory location forward from the address of the next instruction.

Relative addressing is often used with branch instructions when the branch address is in the area surrounding the instruction word itself. It results in shorter address field in the instruction format since the relative address can be specified with a smaller number of bits compared to the number of bits required to designate the entire memory address.

Index Mode & Base Address Mode

Index mode works like relative mode, except the address supplied by the instructions is added to the contents of an index register instead of program counter. 1f the index register contains the value 10, the instruction LDAC (5) reads data from location (5 + 10 =) 15 and stores it in the accumulator.

Base address mode works exactly the same as index mode, except at the index register is replaced by a base address register. In the index mode instructions supply the base address and the index register supplies the offset to that base address. Base address mode instructions supply the offset to the base address stored in the base address register. In practice, only one of the two modes is used, usually Index mode.

1.15.1 A Glimpse of 8086 Addressing Modes

Programming Model for the 8086

Accumulators AH AL (AX), BH BL (BX), CH, CL(CX), DH, DL (DX),

Index Registers BP, SP, SI, DI

Segment Registers CS, SS, DS, ES

Status and Control IP, D I T S Z A P C Flags

The Data Registers, Four registers, named data registers or general purpose registers (GPR), can be used to hold working variables, constants and counters for use in arithmetic and logical calculations. Although they are 16 bits in size they can be used for operations on 8-bit BYTE or 16-bit WORD data.

AX (accumulator) AX is called the accumulator register because it is favoured by the CPU for arithmetic operations. Other operations are also slightly more efficient when performed using AX.

BX (base) In addition to the usual GPR functions, BX has special addressing abilities. It can hold a memory address that points to another variable. Three other registers with this ability are SI, DI, and BP. When using the processor in 32 bit mode, any of the 8 GPRs can be used to hold addresses.

CS (counter) This acts as a counter for repeating or looping instructions. Such instructions automatically repeat and decrement CX and quit when it equals 0.

DX (data) This has a special role in multiply and divide operations. E.g. in multiply it holds the high 16 bits of the product.

The Segment Registers The CPU contains four segment registers, used as base locations for program instructions, data, and stack. In fact, all references to memory on the PC involve a segment register used as a base location.

CS (code segment) Base location of all executable instructions (code) in a program.

DS (data segment) Default base location for variables.

SS (stack segment) Contains the base location of the stack.

ES (extra segment) An additional base location for memory variables.

The Index Registers Index registers contain the offsets of variables. The term offset refers to the distance of a variable, label, or instruction from its base segment. Index registers speed up the processing of strings, arrays, and other data structures containing multiple elements.

SI (source index) Takes its name from the string movement instructions in which the source string is pointed to by the SI register. It usually contains an offset value from the DS register, but it can address any variable.

DI (destination index) This acts as the destination for string movement instructions. It usually contains an offset value form the ES register, but it can address any variable.

BP (base pointer) Contains an asumed offset from the SS register as does the stack pointer. BP is often used by a subroutine to locate variables that were passed on the stack by a calling program.

Special Registers The IP and SP registers are grouped together here, since they do not fit into any of the previous categories.

IP (instruction pointer) IP always contains the offset of the next instruction to be executed. CS and IP combine to form the complete address of the next instruction to be executed.

SP (stack pointer) SP contains the offset, or distance from the beginning of the stack segment to the top of stack. SS and SP combine to form the complete top-of-stack address.

Flags Registers The Flags register is a special 16-bit register with individual bit positions assigned to show the status of the CPU or the results of arithmetic operations. Each relevant bit position is given a name; other positions are undefined:

Bit position

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	x	x	O	D	I	T	S	Z	x	A	x	P	x	C

0 = Overflow S = Sign
D = Direction Z = Zero
I = Interrupt A = Auxiliary Carry
T = Trap P = Parity
x = Undefined C = Carry

The 8086 supports a number of addressing modes, shown by the following addressing mode examples. In the table, a displacement is either a number or the offset of a variable. The EA of an operand refers to the offset (distance) of the data from the beginning of a segment. BX and BP are base registers, and SI and DI are index registers. In many instructions the operand can only be specified by certain addressing modes.

As discussed earlier the use of a register operand, implies the register AM. In 8086, the

Register Operand A register operand may be any 8-bit or 16-bit register. In general, this AM is the most efficient because registers are part of the CPU and no memory access is required. Some examples using the MOV instruction are:

```
mov   ax,bx
mov   cl,al
mov   si,ax
```

Immediate Operand An immediate operand is a constant expression, such as a number, a character, or an arithmetic expression. The assembler must be able to determine the value of an immediate operand at assembly time. Its value is inserted directly into the machine instruction.

Direct Operand A direct operand refers to the contents of memory at the offset of a variable. The assembler keeps track of every label, making it possible to calculate the effective address of any direct operand. In the following example, the contents of memory location count are moved into AL:

```
count       db    20
.
.
mov   al,count
```

OFFSET Operator When it is necessary to move the offset of a label into a register or variable, the OFFSET operator does the trick. Since the assembler knows the offset of every label as the program is being assembled, it simply substitutes the offset value into the instruction. Assuming that the offset of the variable aWord in the following example is 0200h; the MOV instruction would move 200h directly into BX:

```
aWord       dw    1234
.
.
mov   bx,offset aWord
; the above assembles as: mov BX,0200
```

Indirect Operand When the offset of a variable is placed in a base or index register, the register becomes a pointer to the label. For variable containing a single element this is of little benefit but for a list of elements a pointer may be incremented – within a loop, say – to point to each element.

■ **Example** If we create a string in memory at location 0200h and set BX to the base offset of the string, we can access any element in the string by adding its index to BX. The letter 'F' is at index 5 in the following example:

```
;indices are:          0123456
aString     db    "ABCDEFG"
.
.
mov   bx,offset aString     ; BX = 200
add   bx,5                  ; BX = 205
mov   dl,[bx]               ; DL = 'F'
```

Segment Defaults If BX, SI, or DI is used, the EA is by default an offset from the DS (data segment) register. BP, on the other hand, is an offset for the SS (stack segment) register. Assuming that the stack segment and data segment are at different locations, the following two statements would have different effects even if the SI and BP registers contained the same values:

```
mov dl,[si]  ; look in the data segment
mov dl,[bp]  ; look in the stack segment
```

If one really must use BP in the data segment, a segment override operator forces the issue:

```
mov dl,[si]          ; look in the data segment
mov dl,ds:[bp]       ; ditto
```

Based and Indexed Operands based and indexed operands are basically the same: A register is added to a displacement to generate an EA. The register must be SI, DI, BX or BP. A displacement is either a number or a label whose offset is known at assembly time. The notation may take several equivalent forms:

Register added to an Offset

```
mov dx,array[bx]
mov dx,[bx+array]
mov dx,[array+bx]
```

Register added to a Constant

```
mov ax,2[si]
mov ax,[si+2]
mov dx,-2[bp]
mov dx,[bp-2]
```

■ **Example** If we create an array of bytes in memory at location 0200h and set BX to 5, BX can then be used to access the 6th element of the array (note: array indices start at 0).

```
;indices:     0 1 2 3 4 5 6 7  8
array db 00,02,04,08,16,32,64,128,256
.
.
mov   bx,5
mov   al,array[bx]                    ; AL = 32
```

Based-Indexed Operand An operand's EA is formed by combining a base register with anindex register. Suppose BX = 202h and SI =6; then the following instuction would calculate an EA of 208h:

```
mov   al,[bx+si]
```

This technique is often useful for two-dimensional arrays, where BX can address the row and SO the column:

```
array       db 10h,20h,30h,40h,50h
db 60h,70h,80h,90h,A0h
db B0h,C0h,D0h,E0h,F0h
..
mov   bx,offset array ; point to array
add   bx,5          ; select 2nd row
mov   si,2          ; select 3rd col
mov   al,[bx+si]    ; get element
```

Two base registers or two index registers cannot be combined, so the following would be incorrect:

```
mov   al,[bx+bp]    ; error: 2 base regs
mov   dx,[si+di]    ; error: 2 index regs
```

Base-Indexed with Displacement An operands effective address is formed by combining a base register, an index register, and a displacement.

Using the previous two-dimensional array example, we no longer have to set BX to the beginning of the array – we just set BX to the address of the second row relative to the beginning of the table. This makes the code simpler:

```
array       db 10h,20h,30h,40h,50h
db 60h,70h,80h,90h,A0h
db B0h,C0h,D0h,E0h,F0h
..
mov   bx,5          ; select 2nd row
mov   si,2          ; select 3rd col
mov   al,array[bx+si] ; get element
```

A DEBUG example : The following example program shows how a variety of addressing modes may be used when accessing elements of an array. The array is located at offset 150, and the sum will be stored at offset 153:

```
Statement              Comment
A 150                  Assemble data at offset 150
db 10,20,30,0          1st 3 bytes are array, last is sum
<ENTER> ends assembly
A 100                  Assembly code at offset 100h
mov bx,150             BX points to the array
mov si,2               SI will be an index
mov al,[bx]            Indirect operand
add al,[bx+1]          Base-offset operand
add al,[bx+si]         Base-indexed operand
mov [153],al           Direct operand
int 20                 End program
<ENTER> ends assembly
T                      Trace each instruction
.
.
D 150,153              Dump array and sum
```

PROBLEM QUESTIONS

1. An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R1 contains the number 200. Evaluate effective address if the addressing mode of the instruction is

   A. Direct B. Indirect
   C. Immediate D. Relative
   E. Register
   F. Register with R1 as index register

   Answer:

   A. Direct - 400
   B. Indirect - value at memory location 400
   C. Immediate - No concept effective address
   D. Relative - 400 + 302
   E. Register - No concept of effective address
   F. Indexed - 200+400

2. The following information is available in a computers memory. Explain what will be operand values if different addressing modes are used. Assume PC=200, R1(processor register)=400, XR (index register)=100.

Address	Memory Content
200	LDAC with AM bits
201	500
399	450
400	700
500	800
600	900
702	325
800	300

   Answer:

Addressing Mode	Effective Address	Content of AC
Direct Address	500	800
Immediate Address	201	500
Indirect Address	800	300
Relative Address	702	325
Indexed Address	600	900
REGISTER	---	400
Register Indirect	400	700
Autoincrement	400	700
Autodecrement	399	450

3. A two-word instruction is stored in memory at an address designated by symbol W. The address field of the instruction (stored at W+1) is designated by the operand Y. The operand used during the execution of the instruction is stored at an address symbolised by Z. An index register contains the value X. State how Z is calculated from the other addresses if the addressing mode of the instruction is

   A. direct
   B. indirect
   C. relative
   D. indexed

   Answer: (Assuming M(X) indicates value of memory at X).

   A. Z= M(Y) value at the address Y
   B. Z= M(M(Y))
   C. Z=M(PC+2+Y) (assuming instruction is occupying 2 words).
   D. Z=M(X+Y)

1.16 Memory Organisation

Integrated RAM IC's are used while building up the required size RAM. The capacity of the RAM depends on two parameters; the number of words and the number of bits per word. An increase in the number of words requires that the number of bits in address will be increased. Every bit added to the length of the address doubles the number of words in memory.

Consider the possibility of constructing 256Kx8 RAM using 64Kx8 RAM chips. Here, to make 256K words, we have used 4 IC's of 64K. Number of address lines for 64K IC is 16 bits where as for the required system we may need 18bits out of which 16 bits (low order) for address lines of each chip where as last two bits (most significant) for chip select. We employ decoders to select chips.

A Memory-design problem that the computer architect may encounter is the following: Given that N x w-bit RAM IC denoted as MN,W are available, design an N'x w'-bit RAM, where N′> N and/or w′ > w′. A general approach is to construct a pxq array of the M_{N},w ICs, where p=⌈N′/N⌉, q =⌈w′/w⌉ , and [x] denotes the smallest integer greater than or equal to x. In this IC array each row stores N words (except possibly the last row), while each column stores a fixed set of w bits from every word (except possibly the last column). For example, to construct a 1GB RAM using 64M x 1-bit RAM ICs requires p = 16, q= 8, and a total of pq =128 copies of the 64Mb RAM. When N′ > N, additional external address decoding circuitry is usually required.

1.16.1 High Order Interleaving & Low Order Interleaving

If two 8 x 2 chips could be configured as illustrated in Fig.1.79 memory locations of first chip becomes 0 to 7 (0000 to 0111) and the lower chip as locations 8 to 15 (1000 to 1111). For the upper chip a1ways has A3 = 0 and the lower chip has A3 =: 1. This difference is to select one of the two chips. When A3 = 0, the upper chip is enabled and the lower chip is disabled. This configuration uses high-order interleaving. All memory locations within a chip are contiguous within systems memory. Consider the other configuration shown in Fig.1.79, which uses low-order interleaving. The upper chip is enabled when A0 = 0, or by addresses XXXO, in this case words 0, 2, 4, 6, 8, l0, 12, and 14 belongs to this chip. The lower chip is enabled when A0 = l, which is true for addresses 1, 3, 5, 7, 9, 11, 13, and 15. Both look the same the CPU, but low-order interleaving can offer some speed advantage for pipelined memory access.

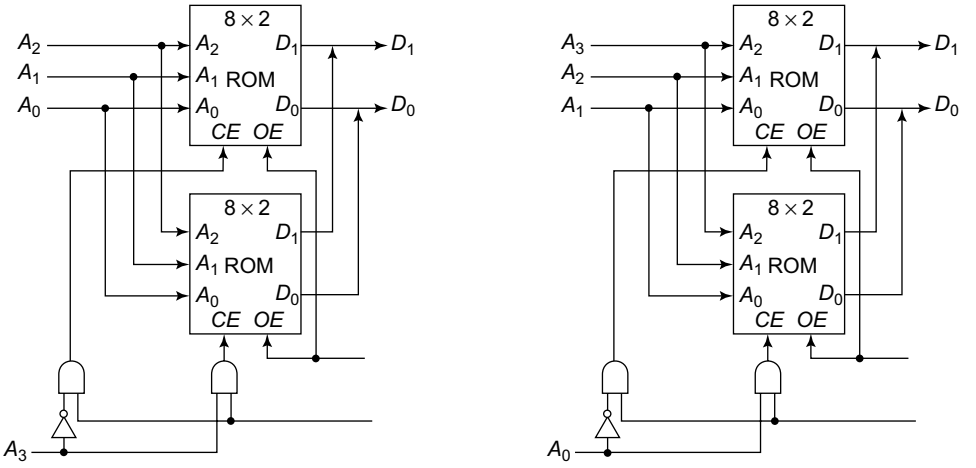


Figure 1.79 "Memory Interleaving

Coincident Decoding

Main drawback of above organisations which are also called as linear organisations is that they need large decoders.

Inside a RAM chip, the decoder with k inputs and 2^k outputs requires 2^k AND gates with k inputs per gate if a straightforward design approach is used. In addition if the number of words is large, and all bits for one bit position in the word contained in a single RAM bit slice, the number of RAM cells sharing the read and write circuits is also large. The electrical properties resulting from both of these situations cause the access and write cycle times of the RAM to become long, which is undesirable.

Here the m-bit address of the word is divided into two parts, X and Y, consisting of mx and my bits, respectively. The cells are arranged in a rectangular array of Nx<= 2^{mx}, rows and Ny <= 2^{my} columns, so the total number of cells is N = Nx.Ny. A cell is selected by the coincidence of signals applied to its X and Y address lines. The 2-D organisation requires much less access circuitry than a 1-D organisation for the same storage capacity. For example, if Nx = Ny = sqrt(N), the number of address drivers needed is 2 sqrt(N), whereas the 1-D RAM we may require N address drivers. In addition, the 2-D organization is a good match for the inherently two-dimensional layout structures allowed by VLS1 technology.

Total number of decoder gates, the number of inputs per gate and the number of RAM cells per bit slice can all be reduced by employing two decoders a coincident selection scheme. In one possible configuration, two k/2-input decoders are used instead of one k-input decoder. One decoder controls the row selection and the other selects column selection scheme. If the RAM chip has m words with 1 bit per word, the scheme selects the RAM cell at the intersection of the Word Select row and the Bit Select column. Since the Word Select is no longer strictly selecting words, its name is changed to Row Select. An output from the added decoder that selects one or more bit slices is referred to as a Column Select.

Commercial DRAM chips uses this organisation very commonly. Moreover, in order to reduce the no of pins first row select address is sent on address lines and followed by column select address in DRAM's.

1.16.2 Isolated I/O & Memory Mapped I/O Devices

In the isolated I/O CPU has distinct instructions for memory read/write and I/O read/write. The addresses what we

use for accessing a device may be same as one of the address in the memory however separate lines are used to indicate it is I/O device address (IO/M). After loading the address by enabling this line processor indicates that the address is for external interface registers. (In 8085 processor address lines are 16. However, I/O ports have 8-bit addresses only).

In the case of memory mapped devices; their interface registers are memory mapped on to the address space of memory. Thus the interface register is treated as being part of the memory. However, assigned addresses can not be used for memory words. Thus, reduces the memory address range available. The CPU can manipulate I/O date residing the interface registers with the same instructions what it uses for normal memory words.

Problems

1. Design a 16x4 memory subsystem with high-order interleaving using 8x2 memory chips for a computer system with an 8-bit address bus.

Answer:

No of RAM chips needed = 16x4/8x2=4

RAM chips organized as 2x2 array such that word size becomes 4 bits and number of words becomes 16.

Actual No of address lines required = log2(16)=4

∴ Out of the eight address lines only 4 are used and as higher order interleaving is needed, least significant 3 bits are common lines for each chip where as 4'th bit is used for chip select.

2. Design a 32x8 memory subsystem with high-order interleaving using 16x2 memory chips for a computer system with an 8-bit address bus.

Answer:

No of RAM chips needed = 32x8/16x2 = 8

RAM chips organized as 2x4 array such that word size becomes 8 bits and number of words becomes 32.

Actual no. of address lines required = log2(32) = 5

∴ Out of the eight address lines only 5 are used and as higher order interleaving is needed, least significant 4 bits are common lines for each chip where as 5'th bit is used for chip select.

3. Design a 16x4 memory subsystem with low-order interleaving using 8x2 memory chips for a computer system with an 8-bit address bus.

Answer:

No of RAM chips needed = 16x4/8x2=4

RAM chips organized as 2x2 array such that word size becomes 4 bits and number of words becomes 16.

Actual no. of address lines required = log2(16)=4

∴ Out of the eight address lines only 4 are used and as low order interleaving is needed, least significant 1 bit is used for chip select and next 3 bits are used as common bits.

4. Design a 32x8 memory subsystem with low-order interleaving using 16x2 memory chips for a computer system with an 8-bit address bus.

Answer:

No of RAM chips needed = 32x8/16x2=8

RAM chips organized as 2x4 array such that word size becomes 5 bits and no. of words becomes 32.

Actual no. of address lines required = log2(32)=5

∴ Out of the eight address lines only 5 are used and as low order interleaving is needed, least significant 1 bit is used for chip select and next 4 bits are used as common bits.

5. Design a 32x8 memory subsystem with split-order interleaving (one high order bit and one low order bit are interleaved) using 8x4 memory chips for a computer system with an 8-bit address bus.

Answer:

No. of RAM chips needed = 32x8/8x4=8

RAM chips organized as 2x4 array such that word size becomes 5 bits and no. of words becomes 32.

Actual no. of address lines required = log2(32)=5

∴ Out of the eight address lines only 5 are used and as split-order interleaving is needed, least significant 1 bit is used for selecting first 2x2 chips versus next 2x2 chips, next 3 bits are common for all chips, the next bit (high order bit) is used to select either chips in top row or bottom row.

6. How many 128x8 RAM chips are needed to provide a memory capacity of 2048 bytes. How many lines of the address bus must be used to access 2048 bytes of memory? How many of these lines will be common to all chips. How many lines must be decoded for chip select? Specify the size of the decoders.

Answer:

No. of address lines are=log2(2048) = 11 bits

No. of chips needed are =2048/128=16

No. of lines common for all chips = 7

No. of lines required for chip select = 11–7=4

7. A computer uses RAM chips of 1024x1 capacity. How many chips are needed, and how should their address lines be connected to provide a memory capacity of 1024 bytes? Similarly, find out how many chips are needed to provide a memory of capacity of 16K bytes?

Answer:

To make 1024 bytes:

No. of chips needed are = 8 (they are connected horizontally such that all the address lines should be common to all chips)

To make 16K bytes:

We need = 16x8=128 chips

Address lines are = 14

Out of which common lines are=10

Chip select lines are = 4

8. A memory system is made up of 4096 bytes of RAM (with 1Kx8 byte RAM IC's) and 4096 bytes of ROM with two 2048Kx8 byte ROM IC's. Draw the memory map table.

Answer:

IC Name	Address range in Hex system
RAM Chip 1	0000-03FF
RAM Chip 2	0400-07FF
RAM Chip 3	0800-0BFF
RAM Chip 4	0C00-0FFF
ROM Chip 1	1000-17FF
RAM Chip 4	1800-1FFF

9. A computer employs RAM chips of 512x8 and ROM chips of 2048x8. The computer system needs 2K bytes of RAM, 4K bytes of ROM, and four interface units, each with four registers. A memory mapped I/O configuration is used. The two highest order bits of the address bus are assigned 00 for RAM, 01 & 10 for ROM and 11 for interface registers. How many RAM, and ROM chips are needed?. Give the address range in Hex for RAM, ROM and interface.

Answer:

Total Memory size required =2K+4K+registers = ~6K

No. of address lines needed are = 13 bits

Chip	Binary Address Range	Hex Address
RAM Chip 1	0 0000 0000 0000-0 0001 1111 1111	0000-01FF
RAM Chip 2	0 0010 0000 0000-0 0011 1111 1111	0200-03FF
RAM Chip 3	0 0100 0000 0000-0 0101 1111 1111	0400-05FF
RAM Chip 4	0 0110 0000 0000-0 0111 1111 1111	0600-07FF
ROM Chip 1	0 1000 0000 0000-0 1111 1111 1111	0800-0FFF
ROM Chip 2	1 0000 0000 0000-1 0111 1111 1111	1000-17FF
Interface registers	Can have any address starts with 11	

10. An 8-bit computer has a 16-bit address bus. The first 15 lines of the address are used to select a bank of 32K bytes of memory. The high order bit of the address is used to select a register which receives the contents of the data bus. Explain how this configuration can be used to extend the memory capacity of the system to eight banks of 32K bytes each, for a total of 256K bytes of memory.

Answer:

Memory banks are used to increase memory capacity of the computer. Selection ports and bank switching registers are used to select the appropriate bank. This is combined SW and HW solution used in early micro computers (Please refer Dogulus V Hall).

11. A computer system is having 512 bytes of RAM (four 128 × 8bit) and 512 byte ROM. Find how many address lines are needed. Also, prepare Memory address map.

Answer:

As total memory (RAM + ROM) is 1024 bytes, we need 10 address lines.

IC	Hexadecimal Address
RAM Chip 1	0000-007F
RAM Chip 2	0080-00FF
RAM Chip 3	0100-017F
RAM Chip 4	0180-01FF
ROM chip	0200-03FF

12. The following memories are specified by the number of word times the number of bits per word. How many address lines and input-output data lines are needed in each case?

(a) 8kx32 (b) 256Kx64

(c) 32Mx32 (d) 4Gx8

Answer:

(a) Address Lines = 13 Data Lines = 32

(b) Address Lines = 18 Data Lines = 64

(c) Address Lines = 15 Data Lines = 32

(d) Address Lines = 32 Data Lines = 8

13. How many 32Kx8 RAM chips are needed to provide a memory capacity of 1M bytes? How many address lines are needed?

Answer: 32 Address lines are=20

14. How many lines must be decoded for the chip select inputs for the problem 13?. Specify the size of the decoder.

Answer: As number of chips of size 32Kx8 needed are 32 for making 1M, we need 5 bits for chip select. Thus, we need 5x32 decoder for chip select.

15. Using the 64Kx8RAM construct the 256Kx32 RAM.

Answer: No of chips required of size 32Kx8 are = 256Kx32 /(64Kx8)=4x4=16

Chip select bits = log2(16) = 4

Decoder size needed = 4x16

16. What will be the size of the ROM which maintains truth table of square of 3 bit numbers?

Answer: 8x4 (square of largest 3 bits number 111 occupies 4 bits only. Thus 4 bits are sufficient to store square of any 3-bit number is ROM such that it can read instead of calculating).

17. What will be the size of the ROM which maintains the truth table for product of two 8 bit numbers?

Answer: Product of two 8 bit numbers requires at most 16 bits. Total number of possibilities are 256x256 and each product requires 16 bits. Thus, 64Kx16 bits.

18. A 32Kx8 RAM chip uses coincident decoding by splitting the internal decoder into row and column select. Assuming that the RAM cell array is square (almost), what is the size of each decoder, and how many AND gates are needed for decoding and address?

Answer: Actual address lines needed = 15bits

If we organize the RAM as 256x128 bit slice processors we may need 8 bits for row selection and 7 bits for column selection.

19. A DRAM has 12 address pins and its row address is 1 bit longer than its column address. How many addresses total does the DRAM have?

Answer: 12+11=23

20. 256Mb DRAM uses 4-bit data and has equal length row and column addresses. How many address pins does the DRAM have?

Answer: 12 bits each.

21. A 1K word RAM is made up of memory modules having 256 words. Draw and determine the memory bank/module address (lower order interleaving(LOI)) and the address of the word 301H in the bank/module.

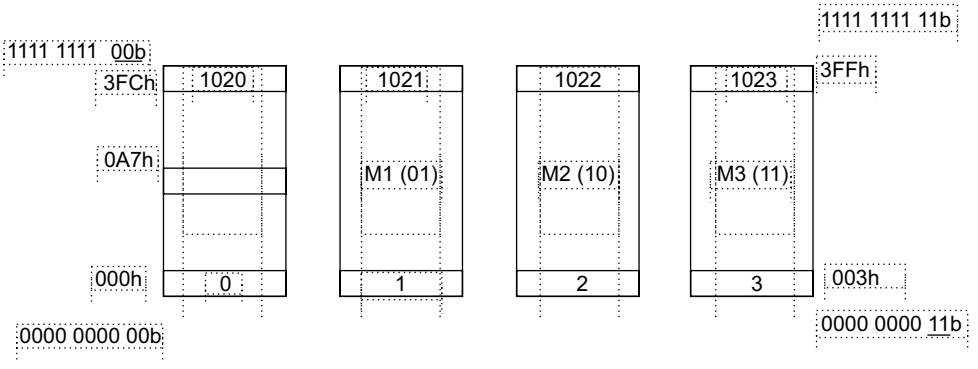
Answer:

- Memory capacity = 1Kwords = 2^{10} = 10 bits for main memory address
- Module/bank size(capacity) = 256 = 2^8 = 8 bits for word in bank/module
- Total number of bank/module = memory capacity/ module(bank) size = $2^{10}/2^8$ = 4 module/bank
- There are 4 memory banks/modules, 2^2, thus 2 bit for the banks/modules address

Memory address = 301H = 1010 0111 00

Memory bank/module address = **00**

Address of the word in the bank/module = **1010 0110 = A7h**

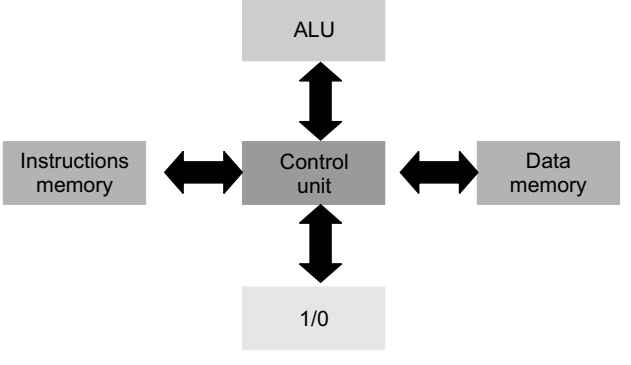


Note on Virtual Memory: Readers are advised to refer to the *Operating Systems* module for page replacement algorithms, page table, etc.

1.17 Cache Memory

1. Cache addresses Von Neumann bottleneck and gives fast, single access to its memory.

2. In Harvard architecture, separate memory is available for instructions and data. Where as in Von Neumann architecture, both instructions and data is stored in the only one available memory unit.



3. The speed difference (latency) between memory and CPU is handled by the cache.
4. Memory, virtual memory transfers are taken care by OS. This is transparent to application programmers but visible for system programmers.
5. Cache, memory transfers are transparent to both application and system programmers.
6. Cache and memory are directly accessible by CPU but not paging disk.
7. Cache coherence or cache in-consistency is seen in Uni-processor's also.
8. Write back or copy back copies cache content to RAM only the selected cache word for replacement is dirty, i.e., modified one.
9. In write through caching, for each write operation on cache updating will be done in RAM.
10. For example for the following code **for (i=1;i<100;i++) x=x+i+1.** When we employ write through caching then 100 times memory is accessed for updating x; where as in the case of write back only one memory access is made while updating information about x.
11. When write miss occurs two approaches are possible : (1) **write allocate** (2) **write–no allocate**. In the case of write allocate first a location in cache is allocated then the required word is read from memory then writing is done in the cache memory. Where as in the write-no allocate, writing will be done directly into memory. Write back employs write allocate policy where as write through uses write-no allocate policy.
12. **Write buffer** is employed to take care of write misses in write through caching. A write buffer stores the data while waiting to be written into memory. After writing into cache, and buffer the processor continues execution. When the data is written into memory it is removed from buffer. If the buffer is full processor must stall. If write occurs in bursts then stalls may increase. To reduce stalls, buffers with more depth is employed. DEC3100 uses buffer with four words deep.
13. A victim cache is a cache used to hold blocks evicted from a CPU cache upon replacement. The victim cache lies between the main cache and its refill path, and only holds blocks that were evicted from the main cache. The victim cache is usually fully associative, and is intended to reduce the number of conflict misses. Many commonly used programs do not require an associative mapping for all the accesses. In fact, only a small fraction of the memory accesses of the program require high associativity. The victim cache exploits this property by providing high associativity to only these accesses.
14. **Trace cache:** One of the more extreme examples of cache specialisation is the trace cache found in the Intel Pentium 4 microprocessors. A trace cache is a mechanism for increasing the instruction fetch bandwidth and decreasing power consumption (in the case of the Pentium 4) by storing traces of instructions that have already been fetched and decoded.
15. **Exclusive versus inclusive:** Multi-level caches introduce new design decisions. For instance, in some processors, all data in the L1 cache must also be somewhere in the L2 cache. These caches are called strictly inclusive. Other processors (like the AMD Athlon) have exclusive caches — data is guaranteed to be in at most one of the L1 and L2 caches, never in both. Still other processors (like the Intel Pentium II, III, and 4), do not require that data in the L1 cache also reside in the L2 cache, although it may often do so.
16. Prominently two type of caching is employed with respect to cache, memory, processor and their physical connectivity with bus. They are **look aside** and **look though**. In the case of look aside all the things are connected to same I/O bus. Where as in the case of look through caching, cache and memory are connected to special (local) bus whose width can be wider than I/O bus width.
17. **Split cache** are I-cache and D-cache (Instruction cache, Data cache)
18. If main memory size increases even more cache levels may be desirable.
19. All PowerPC models have an LRU block (line) replacement policy with either 32 or 64 bytes line sizes. Write back policy on cache miss is employed.
20. Empirical studies indicated that by doubling cache size hit ratio is raised by 30%.
21. Increasing the associativity of a cache, hit rate is increased while conflict misses are reduced.
22. Assuming cache size fixed and if we increase the length of the line then the no of sets or lines reduces and thus conflict misses increases. Associativity reduces.
23. Caches are also dividable into two types based on what address they are caching. They are (1) **physical address caches** and (2) **virtual address caches**.
24. **Aliasing:** Different logically addressed data have the same index/tag in the cache (Example multiple processes with the same range of virtual address spaces).

25. **Physical address caches :** No aliasing is allowed so that the address is uniquely transferred; no need of cache coherence; fewer cache bugs in OS kernels. Bus watching may be needed for proper functioning without frequent cache flushes.
26. Virtual address cache is used then cache flushing is required for each context switch, before I/O writes, after I/O reads.
27. Caches can be divided into 4 types, based on whether the index or tag correspond to physical or virtual addresses:
    - **Physically Indexed, Physically Tagged** (PIPT) caches use the physical address for both the index and the tag. While this is simple and avoids problems with aliasing, it is also slow, as the physical address must be looked up (which could involve a TLB miss and access to main memory) before that address can be looked up in the cache.
    - **Virtually Indexed, Virtually Tagged** (VIVT) caches use the virtual address for both the index and the tag. This caching scheme can result in much faster lookups, since the MMU doesn't need to be consulted first to determine the physical address for a given virtual address. However, VIVT suffers from aliasing problems, where several different virtual addresses may refer to the same physical address. The result is that such addresses would be cached separately despite referring to the same memory, causing coherency problems. Another problem is homonyms, where the same virtual address maps to several different physical addresses. It is not possible to distinguish these mappings by only looking at the virtual index, though potential solutions include: flushing the cache after a context switch, forcing address spaces to be non-overlapping, tagging the virtual address with an address space ID (ASID), or using physical tags. Additionally, there is a problem that virtual-to-physical mappings can change, which would require flushing cache lines, as the VAs would no longer be valid.
    - **Virtually Indexed, Physically Tagged** (VIPT) caches use the virtual address for the index and the physical address in the tag. The advantage over PIPT is lower latency, as the cache line can be looked up in parallel with the TLB translation, however the tag can't be compared until the physical address is available. The advantage over VIVT is that since the tag has the physical address, the cache can detect homonyms. VIPT requires more tag bits, as the index bits no longer represent the same address.
    - **Physically Indexed, Virtually Tagged** (PIVT) caches are only theoretical as they would basically be useless.
28. Aliasing with kernel and user data is major problem with virtual address caches.
29. Flushing will not overcome the aliasing problem with shared memory, with memory mapped files and copy-on-write Unix systems. Every system call forces a cache flush.
30. VAC may lead to poor performance unlike most processes are compute bound.
31. With frequently flushed cache debugging becomes difficult.
32. There exists optimum block size for cache.
33. As cache size increases (very large) many words are fetched which may never used.
34. Hit ratio approaches zero if block size become cache size.
35. Hit ratio may decrease as the no of sets increases for a fixed cache capacity.
36. **Sector mapping cache:** Here both RAM, cache is partitioned into fixed size sectors. Then a fully associative search is performed. That is each sector can be placed in any of the available sector frames.
37. Cache directory is a old terminology which indicates address, data pairs.
38. Cache directory is implemented as an implicit lookup or explicit lookup table.
39. Implicit lookup indicates simultaneous searching of address tags & fetching of corresponding data.
40. Explicit lookup indicates first address tag is searched and then data is fetched.
41. When a page fault occurs context switching may take place as context switching overhead is much smaller than page replacement overhead.
42. Cache miss service time is much less than page fault service time.
43. Cache misses are frequent than page faults.
44. A cache read miss from an instruction cache generally causes the most delay, because the processor, or at least the thread of execution, has to wait (stall) until the instruction is fetched from main memory.
45. A cache read miss from a data cache usually causes less delay, because instructions not dependent on the cache read can be issued and continue execution until the data is returned from main memory, and the dependent instructions can resume execution.
46. A cache write miss to a data cache generally causes the least delay, because the write can be queued and there are few limitations on the execution of subsequent instructions. The processor can continue until the queue is full.

47. Three Cs

- *Compulsory misses* are those misses caused by the first reference to a location in memory. Cache size and associativity make no difference to the number of compulsory misses. Pre-fetching can help here, as can larger cache block sizes (which are a form of prefetching). Compulsory misses are sometimes referred to as *cold misses.*
- *Capacity misses* are those misses that occur regardless of associativity or block size, solely due to the finite size of the cache. The curve of capacity miss rate versus cache size gives some measure of the temporal locality of a particular reference stream. Note that there is no useful notion of a cache being "full" or "empty" or "near capacity": CPU caches almost always have nearly every line filled with a copy of some line in main memory, and nearly every allocation of a new line requires the eviction of an old line.
- *Conflict misses* are those misses that could have been avoided, had the cache not evicted an entry earlier. Conflict misses can be further broken down into *mapping misses*, that are unavoidable given a particular amount of associativity, and *replacement misses*, which are due to the particular victim choice of the replacement policy.

	Compulsory Misses	**Conflict Misses**	**Capacity Misses**
Double the associativity (capacity and line size constant) (halves # of sets)	No effect If the data wasn't ever in the cache, increasing associativity with the constraints won't change that.	Decrease Typically higher associativity reduces conflict misses because there are more places to put the same element.	No effect Capacity was given as a constant.
Halving the line size (associativity and # sets constant) (halves capacity)	Increase Shorter lines mean less "prefetching" for shorter lines. It reduces the cache's ability to exploit spatial locality.	No effect Same # of sets and associativity.	Increase Capacity has been cut in half
Doubling the number of sets (capacity and line size constant) (halves associativity)	No effect If the data wasn't ever in the cache, increasing the number of sets with the constraints won't change that.	Increase Less associativity	No effect Capacity is still constant

48. Process do not context switch on a cache miss.

49. Context switching will invariably make a process to make initial cache misses in an attempt to restore its "locality set".

50. A unified cache is called as homogeneous.

51. A cache is said to be multiported if two or more requests can be made to cache concurrently (a priority algorithm is mandatory).

52. A cache is always pipelined in mainframes. For example the following stages are possible in the pipelined cache: Priority selection, TLB access, Cache access, replacement status update.

53. If a cache supports multiple contexts then context switch may increase "cold-start" miss ratio.

54. "Warm-Start" – means steady state miss ratio.

55. To improve cache performance which supports multiple contexts use main memory to save a process context when context switching takes place and load en masse when selected next.

56. Cache coherence may occur even in Uni-processor also when I/O devices request for the memory and the word in cache and memory are not same.

57. Bit selection algorithm is used in set associative caching.

58. No of sets and blocks per set are inversely proportional.

59. For a given cache size the miss ratio improves with increase in block size and increase in block size captures spatial locality; but it may deteriorate temporal locality.

60. For a given block size, a cache increase improves miss ratio because of temporal locality.

61. Accessing larger caches will be increasing cache word access times. Thus though hit rates increases with increase in cache size but the real benefit of average access time will not be so substantial. Rather if we increase cache size radically we may find worse average access times.

62. Several copies of same physical block may exists in cache with different names. It is known as **synonym problem** and leads to coherence problem.

63. Translation look aside buffer (TLB) used to speed up virtual-to-physical address translation for both executable instructions and data.

64. **Inverse Translation Buffer** (ITB) is accessed on a physical address and indicates all the virtual addresses associated with that physical address in the cache.

65. It is observed that significant fraction of misses is due to task switching for execution of supervisory tasks. To solve this employ use user cache and supervisory cache. Supervisory cache has high-miss ratio because of large working set.

66. Snoopy cache is inconsistent state of the cache.

67. **Disk Cache**

A disk cache is a portion of system memory used to cache reads and writes to the hard disk. In some ways this is the most important type of cache on the PC, because the greatest differential in speed between the layers mentioned here is between the system RAM and the hard disk. While the system RAM is slightly slower than the level 1 or level 2 cache, the hard disk is *much* slower than the system RAM.

Unlike the level 1 and level 2 cache memory, which are entirely devoted to caching, system RAM is used partially for caching but of course for other purposes as well. Disk caches are usually implemented using software (like DOS's SmartDrive).

68. **Peripheral Cache**

Much like the hard disk, other devices can be cached using the system RAM as well. CD-ROMs are the most common device cached other than hard disks, particularly due to their very slow initial access time, measured in the tens to hundreds of milliseconds (which is an eternity to a computer). In fact, in some cases CD-ROM drives are cached to the hard disk, since the hard disk, despite its slow speed, is still much faster than a CD-ROM drive is.

69. A cache controller sends a memory request both to cache and main memory. If satisfied (cache hit) in cache itself data is transferred to CPU while aborting main memory request.

70. Cache misses are (1). Compulsory miss (also known as cold miss) occurs the first time a location is used. (2). Capacity miss is caused by a too large working set. (3). Conflict miss happens when two locations map to the same location in the cache.

71. If h_1, h_2, h_3 are cache hit ratios in a 3 level cache schema and T_{L1}, T_{L2}, T_{L3} are their access times then the average memory access time is given as assuming Tmain is the main memory access time and hit rates till that level.

$T_{av} = h_1 * T_{L1} + (h_2 - h_1) * (T_{L1} + T_{L2}) + (h_3 - h_2 - h_1)*(T_{L1}+T_{L2}+T_{L3})+(1-h_1-h_2-h_3)*(T_{main}+ T_{L1} + T_{L2} + T_{L3})$

72. The best case memory access is same as cache access time where as worst case is same as main memory access time in a memory system with caching.

73. The set associative cache gives more hit rates than the direct mapped.

74. The set associative cache is slower than direct mapped but the higher hit rates will compensate the same.

75. Analysing a piece of code for conflict misses is easy in direct mapped cache than set associative. This analysis is very much needed in embedded program design.

76. **CPU stall**

The time taken to fetch one cache line from memory (read miss or read latency) makes CPU to run out of things to do while waiting for the cache line. This state is called as a CPU stall.

As CPUs become faster, stalls due to cache misses hampers execution speed. Various techniques have been employed to keep the CPU busy during this time.

Out-of-order CPUs (Pentium Pro and later Intel designs, for example) attempt to execute independent instructions after the instruction that is waiting for the cache miss data.

Another technology, used by many processors, is simultaneous multithreading (SMT), or in Intel's terminology, hyper-threading (HT), which allows an alternate thread to use the CPU core while a first thread waits for data to come from main memory.

77. A uni-processor system uses split cache (instruction, data) with hit ratio's h_i and h_d. Any cache access time takes c cycles, block transfer time is b cycles. Among all memory references f_i is the percentage of instructions, among blocks replaced fdir is percentage of dirty block. Assuming write back policy the effective memory access time = $f_i(h_ic+(1-h_i)(b+c)) + (1-f_i)((h_d)c+ (1-h_d)((b+c)(1-f_{dir})+(2b+c)f_{dir}))$

78. Loop tiling breaks up a loop into a set of nested loops with each inner loop performing the operations on a subset of data. See the following versions of a loop fragments.

Fragement1 :

```
for(I=0;I<n;I++)
for(j=0;j<n;j++) c[I]=a[I][j]*b[I]
```

Fragment2 :

```
for(I=0;I<n;I+=2)
for(j=0;j<n;j+=2)
for(II=I;II<min(I+2,n);II++)
for(jj=j;jj<min(j+2,N);jj++) c[II]=a[II][jj]*b[II]
```

Loop tiling may change the order of access of elements of the array. This will effect the cache performance.

79. Consider the following code fragment. The arrays a, b base addresses are 1024 and 4099. Let they are mapped to same block. Let every access brings 4 words. Thus there is a danger of repeated cache misses in a alternative manner. To reduce this padding can be used.

```
for(I=0;I<n;I++)
for(j=0;j<n;j++)a{j][I]=b[j][I]*c
```

80. Given the following memory characteristics, draw the memory configuration for a direct mapped cache, 2-way set-associative cache, 4-way set-associative, and 8-way set-associative.

Cache memory: 256 bytes

Main memory: 1024 bytes

Cache line size/block size: 4 bytes

Include in the description of each memory system the fields of which the main memory addresses are composed.

Specifically indicate the cache blocks to which the following main memory addresses are mapped:

Main memory address 0

Main memory address 100

Main memory address 256

Answer:

Direct-mapped:

Main memory address 0 ⇒ 0/4 = 0 = main memory block number

0 % 64 = 0 = cache block number

Main memory address 100 ⇒ 100/4 = 25 = main memory block number

25 % 64 = 25 = cache block number

Main memory address 256 = 256/4 = 64 = main memory block number

64 % 64 = 0 = cache block number

Main memory address (10 bits)

Tag	Index	Offset
2	6	2

2-way set-associative:

Main memory address 0 ⇒ 0/4 = 0 = main memory block number

0 % 32 = 0 = cache set number; cache block numbers 0, 1

Main memory address 100 ⇒ 100/4 = 25 = main memory block number

25 % 32 = 25 = cache set number; cache block numbers 50, 51

Main memory address 256 = 256/4 = 64 = main memory block number

64 % 32 = 0 = cache set number; cache block numbers 0, 1

Main memory address (10 bits)

Tag	Index	Offset
3	5	2

4-way set-associative:

Main memory address 0 ⇒ 0/4 = 0 = main memory block number

0 % 16 = 0 = cache set number; cache block numbers 0,1,2,3

Main memory address 100 ⇒ 100/4 = 25 = main memory block number

25 % 16 = 9 = cache set number; cache block numbers 36, 37, 38, 39

Main memory address 256 = 256/4 = 64 = main memory block number

64 % 16 = 0 = cache set number; cache block numbers 0,1,2,3

Main memory address (10 bits)

Tag	Index	Offset
4	4	2

8-way set-associative:

Main memory address 0 ⇒ 0/4 = 0 = main memory block number

0 % 8 = 0 = cache set number; cache block numbers 0,1,2,3,4,5,6,7

Main memory address 100 ⇒ 100/4 = 25 = main memory block number

25 % 8 = 1 = cache set number; cache block numbers 8,9,10,11,12,13,14,15

Main memory address 256 = 256/4 = 64 = main memory block number

64 % 8 = 0 = cache set number; cache block numbers 0,1,2,3,4,5,6,7

Main memory address (10 bits)

Tag	Index	Offset
5	3	2

81. When a CPU writes to the cache, both the item in the cache and the corresponding item in the memory must be updated. If data is not in the cache, it must be fetched from memory and loaded in the cache. If t_l is the time taken to reload the cache on a miss, then the effective average access time of the memory system is given by:

$t_{ave} = ht_c + (1 - h)t_m + (1 - h)t_l$.

EXERCISE

1. Only read access is seen in

A. Instruction cache B. Data cache

C. TLB D. L1 cache

2. First hardware cache that is used in a computer system is

A. Instruction cache B. Data cache

C. TLB D. L1 cache

3. Execution time for R1<– R2 + R3, R4<– R1 + R3, R5<– R6 + R3 is ___cycles. (Assume three stages with last stage being execution and write back).

Answer: 6 cycles. If we observe both second third instructions have read after write dependency on first instruction. Thus, they can be executed only after first instruction completing its third stage. Thus, second instruction has to wait (stall) for one cycle. That is, first completes by 3 cycles, while second instruction completes in 5^{th} cycle, while third instruction completes in 6^{th} cycle.

4. A computer system has a cache with access time 10ns, a hit ratio of 80% and average memory access time is 24ns. Then what is the access time for physical memory?

Answer: 24=0.8*10 +0.2(x+10)

Where x is memory access time

X = 70ns

5. A computer has a cache and a disk used for virtual memory. If a referenced word is in the cache, 5ns are required to access it. If it is in main memory, but not in cache, 60ns are needed to load it into cache and the reference is made in the cache. If the word is not in main memory, 15 ms are needed to fetch the word from disk, followed by 60 ns to copy it to the cache, and then the reference is made in the cache. The cache hit ratio is 93 % and the main memory hit ratio is 0.65. What is the average time required to access a referenced word on this system?

Answer: 0.93 * 5 ns + (0.07 * 0.65 * (60ns + 5ns)) + (0.07 * 0.35 * (15ms + 60 ns + 5 ns))

= 4.65 ns + 2.9575 ns + 1.96ns = 9.5675 ns

6. The one which takes more latency

A. Cache read miss in instruction cache

B. Cache read miss in data cache

C. Cache write miss in a data cache

7. We have two caches A and B. In which tag field length is more? Assume physical address is 32 bits.

Cache A: 8KB 4 way set associative with 64 bytes cache blocks.

Cache B: 256KB 8 way set associative with 128 bytes cache blocks.

Answer: In cache A, there are 8KB/64 = 128 cache blocks. As it's 4-way set associative, it contains 128/4=32 sets (and hence 2^5 = 32 different indices). There are 64=2^6 possible offsets. Since the CPU address is 32 bits, this implies physical address 32 bits is decomposed into 21 tag, 5 bits index and 6 bits offsets. In the case of cache B, the same 32 physical address implies 32=17+8+7, and where 17 bits is tag field. Thus, cache A tag field is more than cache B.

8. What is the average memory access time of a m/c whose hit rate is 93%, with cache access time of 5ns and main memory access time of 80ns?

Answer: Average access time = 0.93*5ns + 0.07 *(80ns + 5ns) =11.5ns

9. If we want an average memory access time of 6.5ns, our cache access time is 5ns, and our main memory access time is 80ns, what cache hit rate must we achieve?

Answer: Let required hit rate = x

6.5ns = x*5ns + (1 – x)*(80 + 5)

x = hit rate required = .98 = 98.12%

10. Assume that a system has two level cache with hit ratios 90%, 97% and access times 4ns and 15ns and main memory access time is 80ns. What is the average memory access time?

Answer: Average access time = 0.9*4ns + (1–0.9)*0.97 * (15ns + 4ns) + (1–0.9–(1–0.9)*0.97) * (80ns + 4 + 15)

11. Consider a direct mapped cache with 64 blocks and a block size of 16 bytes. What block number does byte address 1200 map to?

Answer: Binary code of the address 1200

= 1**001011**0000

Block number = 001011 = 11

12. If a cache has 64 byte cache lines how long does it takes to fetch a cache line if main memory takes 20ns cycles to respond to each memory request and returns 2 bytes of data in response to each request?. What will be the same if page mode DRAM is used with 20 cycles for first 2 bytes and 10 cycles for subsequent each 2 bytes?

Answer:

a. 64/2*20 = 640 cycles

b. 20 + (64–2)/2*10 = 330 cycles

13. In a 16KB direct mapped cache with line length of 32 bytes how many bits are needed to determine the byte that a memory operation references within a cache line, and how many bits are used to select the line in the cache that may contain data.

Answer:

Log_2 32=5 bits

16KB/32=512=9bits

14. A certain memory has four levels with hit ratios 0.8, 0.95, 0.99 and 1.0 respectively. A program makes 3000 references to this memory system. Calculate the exact number of references made and that are satisfied at each level.

First level =0.8*3000=2400 references

Second level = (1–0.8)*0.95*3000= 570 references

Third level = (1– 0.8 – (1–0.8)*0.95) *.97 * 3000=29.7

Fourth level =0.3

15. A two level memory has access times of 10–8 and 10–3. What should be the hit ratio in order for the access efficiency to be at least 65 percent of its maximum possible value.

Answer: Cache access time =10ns

Memory access time =1ms

Required average memory access time =0.65ms

Let hit ratio = x

0.65ms = x*10ns + (1– x)*1ms

x = 0.035 = 3.5%

16. How many total bits are needed for a direct mapped cache with 64KB of data and one word blocks, assuming a 32bit addresses and computer is a 32 bit computer.

Address lines=32bits

Instruction size=32bits

No of locations=64KB/4bytes=16K words=2^{14}

Therefore, tag bits=32–14–2=16 (here 2 bits is for byte offset)

Valid bit=1

Therefore, total cache size = 2^{14} *(32+16+1) = 784 k bits = 98 KB

17. Assume direct mapped cache with 4 locations. Find out the final content of cache if the sequence of requests are 001, 010, 011, 100, 101 and 111. Repeat the same for two way set-associative cache. Assume with both LRU replacement policy. The content of the memory is as follows. Assume initially cache is empty.

Address	Data
000	0101
001	1111
010	0000
011	0110
100	1000
101	0001
110	1010
111	0100

Answer: For Direct Caching

Block	Tag	Data
00	1	1000
01	1	0001
10	0	0000
11	1	0100

For Set associative with 4 sets

Set	Block0 Tag	Block0 Data	Block1 Tag	Block1 Data
00	1	1000		
01	0	1111	1	0001
10	0	0000		
11	0	0110	1	0100

For set associative with 2 sets

Set	Block0 Tag	Block0 Data	Block1 Tag	Block1 Data
0	01	0000	10	1000
1	10	0111	11	0100

18. Consider the following program

```
main(){
int a[128],b[128],c[128],I;
for(I=0;I<128;I++)a[I]=b[I]+c[I];
}
```

Suppose the program is running on m/c with 32-KB and 16KB data and instruction caches. Each cache is direct mapped with 128 byte cache line and integers are 4 bytes.

a. Does the system's choice of where to place the instructions of the program in memory affect the hit rate of either cache? (Assume only one program is running, ignore OS).

b. Is it possible place the arrays such that there are no conflict or capacity misses in the data cache. If so, what are the constraints on how three arrays are placed in memory?

Answers:

a. No effect. Only if we consider OS then there is a possibility of conflict misses.

b. No possibility of capacity misses as the total memory of arrays are 1.5KB which is very less compared to cache size. To eliminate conflicts we have to make sure that three arrays are mapped to different groups of sets in cache.

19. For a cache with 128-bytes cache lines the address of the first word in the line containing address 0xa23847ef is 0xa238480.

Answer:

As the cache line is 128 bytes, the low 7 bits of the address indicate which byte within the line an address refers. Since lines are aligned the address of the first word in the line can be found by setting low order 7 bits in the given address to 0's.

20. A cache has 64KB capacity, 128-byte lines, and is 4-way set-associative. The system containing the cache uses 32 bit addresses. Then calculate number of lines, sets, entries in the tag array, number of bits in the tag. If the cache is write-through how many bits are needed for each entry in the tag array, and how much total storage is required for the tag array if an LRU replacement is used? What is the cache is write-back?

Answer:

No of lines = 64KB/128bytes = 512

$\therefore$ No of tag entries = 512

No of sets = 512/4=128

As there are 128 sets, 7 bits are needed to select the set.

As the line size is 128 bytes other 7 bits are needed for selecting word in a line

$\therefore$ Tag bits = 32 – 7 – 7 = 18

As LRU replacement is used and cache is 4-way associative 2 bits are needed to hold the age of the line. Additional 1 bit is needed as valid bit. Thus 18+2+1=21 bits are needed in the tag. As there are 512 lines, total memory of the tag array = 512*21=10752 bits.

Write-back policy if used additional dirty bit is needed. Thus tag array contains 512*22 bits, i.e. 11264 bits.

21. Suppose a given cache has an access time (cache hit latency) of 10ns and miss rate 5 percent. A given change to the cache will decrease the miss rate to 3 percent, but it will increase the cache hit latency to 15ns. Under what conditions does this change result in greater performance (lower average memory access time)?

Answer:

$(15\ ns*0.97 + T_{miss} * 0.03) < (10ns*0.95 + T_{miss} *0.05)$

Solving for T_{miss}, $T_{miss} = 252.5$. As long as T_{miss} penalty is greater than this value the reduction in the cache miss frequency will be more significant than the increase in the cache hit time.

22. A cache has hit rate of 95%, 128 byte lines and a cache hit latency of 5ns. The main memory takes 100ns to return the first 32 bits of a line and 10ns to return to subsequent word. What is the miss penalty for this cache (Assume that the cache waits until the line has been fetched into the cache and then reexcutes the memory operation, resulting in a cache hit. Neglect the time required to write the line into the cache once it has been fetched from the main memory. Also assume that the cache takes the same amount of time to detect that a miss has occurred as to handle a cache hit).

If doubling the cache line length reduces the miss rate of 3 percent, does it reduce the average memory access time?

Answer:

As the cache lines are 128 bytes and each word is 32 bits, thus the time required when cache miss occurs = 100ns + 31*10ns = 410ns. As the cache the re-executes the operation that caused the miss taking 5 ns. Then total cache miss time becomes 410ns + 2*5ns = 420ns.

Average memory access time = 0.95*5ns + 0.05*420ns = 25.75

If we double the cache line length cache miss time becomes = 100ns + 63 * 10ns + 2*5ns = 740ns.

$\therefore$ Average memory access time = 0.95*5ns + 0.03 * 740ns = 27.1ns.

$\therefore$ Average memory access time increases.

23. A system initially has split caches of 16KB. They are replaced with unified cache with 48KB. 50 percent more than the total capacity of the two caches, but a given program sustains more total cache misses (counting both instruction and data references) after the change than before.

a. If the program takes 10KB of memory references 64KB of data, what is the most likely explanation for the increase in misses?

b. Suppose the program takes up 10KB of memory and references 15KB of data. What would the most likely explanation for the increase in cache misses be?

Answer:

a. Miss rate may increase as both instructions and as well as data references may compete for the cache locations. Thus hit rate may increase.

b. Probably the location of program and data in the memory is reason.

24. Suppose we have a processor with a base CPI (cycles per instruction) of 1.0, assuming all references hit in the primary cache and clock rate of 500MHz. Assume a main memory access time 200ns, including all the miss handling. Suppose the miss rate per instruction at the primary cache is 5%. How much faster will the machine be if we add a secondary cache that has a 20ns access time for either a hit or a miss and large enough to reduce the miss rate to main memory to 2%.

Answer:

The miss penalty to main memory = 200ns/2 ns/clock cycle = 100 clock cycles

Total CPI= base CPI + Memory stall cycles per instruction

For the m/c with one level caching Total CPI = 1. + 5%100 = 6

With two levels of caching, the miss penality = 20ns / 2ns/clock cycle =10 clock cycles

Total CPI= 1+ Primary stall per instruction + Secondary stalls per instruction

=1+(5% – 2%) of 10 + 2% of (100 + 10) = 3.5

Thus the machine with secondary cache is faster by =6.0/3.5 = 1.7

25. Consider details of two cache memories. Which is faster in accessing memory? In each case, t_c is the access time of the cache memory, tm is the access time of the main store, and h is the hit ratio.

Cache A. t_m = 70 ns, t_c = 10 ns, h = 0.9

Cache B. t_m = 60 ns, t_c = 5 ns, h = 0.9

Answer:

Cache A: effective memory access time = 0.9*10ns + 0.1(70+10)ns = 17ns

Cache B: Effective memory access time = 0.9*5ns + 0.1(60+5)ns = 11ns

Therefore, cache B is better than cache A.

26. For the following system, calculate the hit ratio h required to achieve the stated speedup ratio S compared to the system with only memory. Use following details:

t_m = 50 ns, t_c = 10 ns, S = 2.0

Answer: 2*(h*10ns+(1-h)*(50+10)ns)=50ns

10h + 60 – 60h = 25

50h = 35

h = 35/50 = 0.7 = 70%.

27. A computer performs both internal operations and memory accesses. The average time to execute an instruction is

$t_{ave} = F_{internal}.t_{cyc} + F_{memory} . [h.t_c + (1 – h)(t_c + t_d)].t_{cyc}$

$F_{internal}$ = fraction of time doing internal operations

F_{memory} = fraction of time spent doing memory accesses operations

t_{cyc} = clock cycle time

t_c = cache access time in clock cycles

t_d = additional penalty paid when accessing main memory

For the following systems calculate the average cycle time

a. $F_{internal}$ = 20%, t_{cyc} = 20 ns, t_c= 1, t_d= 3, h = 0.95

b. $F_{internal}$ = 50%, t_{cyc} = 20 ns, t_c= 1, t_d= 3, h = 0.9

Answer: 0.2*20 + 0.8 [0.95*1 + 0.05(1+3)]*20=4 + 16(9.5+2) = 188ns

b. 0.5*20 + 0.5[0.9*1 + 0.1*(1+3)]*20 =10+10 (9+0.4)=104ns

28. Cache can be accessed in parallel or serial with main memory. In a parallel access mode both the cache and the main store are accessed simultaneously. If a hit occurs, the access to the main store is aborted. In a serial access, the cache is first examined and, if a miss occurs, the main store is accessed. Assume that the hit-ratio is *h* and that the ratio of cache memory access time to main store access time is *k* (k < 1). Derive expressions for the speedup ratio of both a parallel access cache and a serial access cache

Answer: Assuming T_c, T_m are cache and memory access times. Also $k = T_c/T_m$

Effective memory access time in the case of parallel access = $h*T_c + (1–h)T_m$

Effective memory access time in the case of serial access = $h*T_c + (1–h)(T_c+T_m)$

Speed up = $(h*T_c + (1–h)T_m)/(h*T_c + (1–h)(T_c+T_m))$

Taking $T_c = kT_m$

Speed up = $(hkT_m + (1–h)T_m) / (hkT_m + (1–h)(k+1) T_m)$

= $(hk – h +1)/ (hk + k+1–hk–h)$

= $(h(k–1) +1)) / (k+1–h)$

29. Given a 2 Kbytes two-way set associative cache with 16 byte lines and the following code:

for (int i = 0; i < 1000; i++) A[i] = 40 * B[i];

Compute the overall miss rate (assume array elements takes 4 bytes). What kind of cache locality is being exploited?

Answer: Each array contains 1000 elements. Each cache line contains 4 words. Since each array element will be accessed only once, all misses shall be compulsory misses. Since every 4 words will be loaded or written back simultaneously in a cache, the miss rate is 25% since every 4th access misses.

Spatial locality is being exploited.

30. Consider a cache with the following characteristics:

32-byte blocks

8-way set associative

256 sets

32-bit addresses

Write back policy

LRU replacement policy

How many bytes of data storage are there?

$256 \times 8 \times 32 = 218 = 64$ KB

How many tag bits per set?

32- log2256-log232=32-8-5=19 bits per set

What operation is needed upon a read-miss (the program wants to read from a memory location that is not in the cache)?

a. Find the LRU cache line to replace. If it is dirty, write the block to next level cache / memory.

b. Fetch 32-byte memory data from next level cache/ memory of that memory address and other data from the same block/line;

c. Update the tag bit of that cache line.

What operation is needed upon a write-miss (the program wants to write to a memory location that is not in the cache)?

a. Find the LRU cache line to replace. If it is dirty, write the block to next level cache / memory.

b. Fetch 32-byte memory data from next level cache/ memory of that memory address and other data from the same block/line;

c. Write to the memory location in that cache line. Mark the cache line as dirty. Update the tag bit of that cache line.

31. Consider a 3-way set associative cache. A, B, C, D are memory addresses that have the same index bits but different tag bits from each other. In a program, the reference sequence is as follows:

A, B, A, C, D, A, D, C, A, C

What is the miss rate if the cache is using LRU replacement policy?

40%

What is the miss rate if the cache is using MRU replacement policy?

50%

Assuming the memory addresses being accessed are still A, B, C and D, provide a case of memory reference sequence with length=10, in which MRU performs better than LRU. Show the reference sequence and the miss rate of LRU and MRU policy.

For example: A,B,C,D,A,B,C,D,A,B

32. A cache has a 90% hit rate. It also equipped with a predictor which predicts correctly 75% of the time if there is a cache hit. A cache hit with a correct prediction will take 2 cycles, a cache hit with an incorrect prediction will take 4 cycles, and a cache miss (way-prediction irrelevant) will take 60 cycles. Compute the average memory access time of the cache with way-prediction.

Without prediction (the original 2-way set associative cache) has the same hit rate and miss time, but a 3 cycle hit time. By how many cycles does prediction improve the average memory access time?

Note: It is important to notice that 60 cycles is the miss time, not the miss penalty (miss time = miss penalty + hit time).

Answer:

Average memory access time= $(0.9)((0.75)2 + (0.25)4) + (0.1)60 = (0.9)(1.5 + 1) + 6 = (0.9)(2.5) + 6 = 8.25$

Average memory access time (without WP) = $(0.9)(3) + (0.1)(60) = 2.7 + 6 = 8.7$

Improvement: $8.7 - 8.25 = 0.45$

33. Examine the code given below to compute the average of an array:

```
total = 0;
for(j=0; j < k; j++) {
    sub_total = 0;
    /* Nested loops to avoid overflow */
    for(i=0; i < N; i++) {
        sub_total += A[j*N + i];
    }
    total += sub_total/N;
}
average = total/k;
```

When designing a cache to run this application, given a constant cache capacity and associativity, will you want a larger or smaller block size? Why?

Answer: Examining the code it is good to see that the program accesses consecutive addresses and never reuses any of them. Because of this you will want to leverage its spatial locality by using a larger blocker size to reduce compulsory misses

34. Examine the code given below (note it is slightly different than previous question):

```
total = 0;
for(i=0; i < N; i++) {
    sub_total = 0;
    /* Nested loops to avoid overflow */
    for(j=0; j < k; j++) {
        sub_total += A[j*N + i];
    }
    total += sub_total/k;
}
```

average = total/N;

Generally, how will the size of the array and the cache capacity impact the choice of line size for good performance? Why?

Answer: Here we have to explain how to pick a line size given a cache capacity and array size. Changing the size of the array or the cache were not options.

Basically you want to be able to fit at least one iteration of the outer loop in the cache with a block size ≥ 2 words. To do this, you want at least two "columns" of A to be able to fit in the cache at a time, so the you will get some hits from spatial locality. Without this, the program will suffer 100% misses. If the block size is two and en entire column can fit, one iteration of the outer loop result in all misses, but the next iteration will be all hits.

As the size of the array gets smaller relative to the cache capacity, the "columns" are "shorter," so you will want a larger line size to reduce compulsory misses.

As the size of the array gets larger relative to the cache capacity, the "columns" will get "taller." You will need to make the line size smaller to get more lines in the cache to reduce conflict misses so it can still hold an entire iteration of the outer loop.

35. A block direct mapping cache has lines/slot that contains 4 words of data. The cache size is 64Kline and the main memory capacity is 16Mbytes.

a. Given the following main memory address 3AF-F80h, find the values for tag, line/slot and word field in hexadecimal.

b. Given the following information, find the main memory address in hexadecimal.

tag	line/slot	word 11	word 10	word 01	word 00
3Fh	9DA3h		**Data**		

Answer:

Physical address length: 16Mbyte = $2^4 \times 2^{20}$ = 24 bits

Number of bits required to access individual for words in cache=4 bytes = 2^2 = 2 bits

Number of bits required to access all for lines/slots in cache= 64Kline = $2^6 \times 2^{10}$ = 16bits

Therefore, Tag= 24-16-2 = 6bits , line (index) = 16bits, word = 2bits

a. Tag line word

0011 10 10 1111 1111 1000 00 00

Tag = 00 1110 = 0Eh, Line = 10 1111 1111 1000 000 =BFE0h, word =00 = 0h

b. Tag line word

11 1111 1001 1101 1010 0011 10 = FE768Eh

36. Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

a. How is a 16-bit memory address divided into tag, line number, and byte number? Tag-8, (index) line-5,word-3

b. Into what line would bytes with each of the following address be stored?

0001 0001 **0001 1**011-- 03

1100 0011 **0011 0**100 -- 06

1101 0000 **0001 1**101 -- 03

1010 1010 **1010 1**010 – 15

As line is 5 bits long, we can find the answer by considering 3rd to 7th bits of the addresses from LSB (shown in bold and underlined). The lines in which the addresses gets mapped are: 3,6,3, and 15.

c. Suppose the byte with address 0001 1010 0001 1011 (1A1B) is stored in the cache. What are the addresses of the other bytes stored along with it?

As the block size is 8 bytes, along with the required byte other bytes whose addresses given below are stored.

0001 1010 0001 1000 (1A18)

0001 1010 0001 1001 (1A19)

0001 1010 0001 1011 (1A1A)

0001 1010 0001 1100 (1A1C)

0001 1010 0001 1101 (1A1D)

0001 1010 0001 1110 (1A1E)

0001 1010 0001 1111 (1A1F)

d. How many total bytes of memory can be stored in the cache?

For each line, no of bits = tag + (no of word* word size) = 8 + (8 *8) = 72 bits

For 32 lines , total no of bits = 32 * 72= 2304 bits

$= \frac{2304}{8} = 288$ bytes

37. A set associative cache size of 16Kline divided into 2-line sets (2-way) with block a line of 4 words. Main memory capacity is 32Mbyte.

a. What is cache address (**in hexadecimal**) for the following main memory address 133AC0Bh?

b. What is main memory address (**in hexadecimal**) for the following cache data?

Tag	Set	word 11	word 10	word 01	word 00
34Ah	1B5Ch			**Data**	

Answer:

Number of how many bits are used for main memory address= 25 bits

Number of modules (sets) in cache = 16K/2 = 8K

Number of bits for set numbers=13 bits

Number of bits for block words= 2 bits

Thus, Tag: 25-13-2 =10bits, set- 13 bits, word 2 bits

a. Address 133AC0Bh can be decomposed into tag, set and word by representing the same in binary code. Middle bits of this binary code indicates the cache address, which is 2B02.

01001100111 **<u>01011 00000010</u>** 11 ----- 2B02h

b. Tag, set, word 10bit 13 bits 2 bits

= 11 1000 1010 1 1101 0101 1100 01

Grouping 4bits from left : the required address becomes: 1C57571h

38. A set associative cache size of 64Kline divided into 4-line sets (4-way) with block of 4 words. Main memory capacity is 64 Mbyte.

e. How many bits are used for main memory address?

64M = $2^6 \times 2^{20}$ =26 bits

f. How many modules (set) in cache?

$= \frac{\text{lines}}{\text{noofways}} = \frac{64K}{4} = 16\,\text{Ksets}$

g. How many bits for set numbers? =16K = $2^4 \times 2^{10}$ = 14 bits

h. How many bits for block words? = 4 = 2^2 = 2bits

i. Show the format of main memory addresses with tag, set and word bits.

Tag = 26-14-2=10 bits, set =14 bits,word=2 bits

j. What is cache address (in hexadecimal) for the following main memory address 377AC01h?

11 0111 0111 10<u>10 11</u>00 0<u>000 00</u>01 = 2B00h

k. What is line size (cache word/cache block size)?

= tag + (no of word per line * word size)

= 10 +(4 * 8) = 42 bits for each line

1. What is main memory address (in hexadecimal) for the following cache data?

Tag	Set	word 11	word 10	word 01	word 00
34Ah	3B5Ch	**Data**			

11 0100 1010 11 1101 0101 1100 11 = 34AF573h

39. A machine has a two-level caches L1 and L2. Further, out of every 100 memory references, 5 misses are seen at the L1 cache and 2 misses at the L2 cache. Calculate various miss rates, local and global. Using the miss rates, what is the average memory access time assuming the L2 miss penalty is 200 cycles, the L2 hit time is 20 cycles, and the L1 hit time is 1 cycle

Answer: various miss rates are:

L1 global: 5%

L1 Local: 5%

L2 global: 2%

L2 Local : 40%

Average Memory access time= 0.95*1 + 0.02*(1+20)+ 0.03(1+20+200)=0.95+4.2+ 6.63= 11.78 cycles.

40. Effective Access Time Example: A computer has a single cache (off-chip) with a 2 ns hit time and a 98% hit rate. Main memory has a 40 ns access time. What is the computer's effective access time? If we add an on-chip cache with a .5 ns hit time and a 94% hit rate, what is the computer's effective access time? How much of a speedup does the on-chip cache give the computer?

Answers:

Effective memory access time in the first case= 0.98*2 ns + .02 * (40+2) ns = 2.8 ns.

Effective memory access time with on-chip cache= .5 ns + .06 * (2 ns + .02 * 40 ns) = .668 ns.

Therefore, speedup is 2.8 / .668 = 4.2.

41. Virtual memory problem: Assume a computer has on-chip and off-chip caches, main memory and virtual memory. Assume the following hit rates and access times: on-chip cache 95%, 1 ns, off-chip cache 99%, 10 ns, main memory: X%, 50 ns, virtual memory: 100%, 2,500,000 ns. Notice that the on-chip access time is 1 ns. We do now want our effective memory access time to increase much beyond 1 ns. Assume that an acceptance effective access time is 1.6 ns. What should X be (the percentage of page faults) to ensure that EMAT is no worse than 1.6 ns?

Answer: EMAT = 1ns + .05 * (10 ns + .01 * (50 ns + (1 – X) * 2,500,000 ns)). Since we want EMAT to be no more than 1.6 ns, we solve for X with 1.6 ns = 1ns + .05 * (10 ns + .01 * (50 ns + (1 – X) * 2,500,000 ns)). X = 1 – ((((((1.25 ns – 1 ns) / .05) – 10 ns) / .01) – 50 ns) / 2,500,000). X = 0.99994 = 99.994%. Our miss rate for virtual memory must be no worse than .006%.

42. Cache/Memory Layout: A computer has an 8 GB memory with 64 bit word sizes. Each block of memory stores 16 words. The computer has a direct-mapped cache of 128 blocks. The computer uses word level addressing. What is the address format? If we change the cache to a 4-way set associative cache, what is the new address format?

Answers:

With 8 GB and a 64 bit word size, there are 8 GB / (8 bytes / word) = 1 GW of memory. This requires 30 bits for an address. Of the 30 bits, we need 4 bits for the word on the line and 7 bits for the block number, leaving 30 – (7 + 4) = 19 bits for the tag. So the address format is 19 – 7 – 4.

If we have a 4-way set associative cache instead, then there will be 4 sets with 128 / 4 = 32 blocks per set. So we would only need 5 bits for the block number, leaving 30 – (5 + 4) = 21 bits for the tag. So the address format is 21 – 5 – 4.

43. Direct Mapping Question: Assume a computer has 32 bit addresses. Each block stores 16 words. A direct-mapped cache has 256 blocks. In which block (line) of the cache would we look for each of the following addresses? Addresses are given in hexadecimal for convenience.

a. 1A2BC012

b. FFFF00FF

c. 12345678

d. C109D532

Answers: Of the 32 bit address, the last four bits denote the word on the line. Since four bits is used for one hex digit, the last digit of the address is the word on the line. With 256 blocks in the cache, we need 8 bits to denote the block number. This would be the third to last and second to last hex digit.

a. this would be block 01, which is block 1

b. this would be 0F which is block 15

c. this would be 67 which is block 103 (remember, 67 is a hex value)

d. this would be 53 which is block 83.

44. A computer has a simple singlelevel paged virtual memory system (with no segments), a hardware loaded TLB (that is, the hardware consults the page table stored in memory directly on a cache miss), and no cache, give a symbolic formula for the average memory access time, as a function of:

PT = probability of a TLB miss

PP = probability of a page fault, given a TLB miss occurs

TT = time to access TLB

TM = time to access memory

TD = time to transfer a page to/from disk

PD = probability page is dirty when replaced

Answer: The average access latency in a simple single-level paged virtual memory system with hardware-loaded TLB, no cache, probability pt of a TLB miss, probability pp of a page fault given a TLB miss occurs, probability pd that a page is dirty when replaced, time tt to access the TLB, time tm to access memory, and time td to access the disk is:

tt + tm + pt (tt + tm + pp (td + pd td)).

This formula quantifies the following: the TLB and memory are always accessed; whenever there is a TLB miss, the TLB and memory are accessed a second time; whenever the TLB miss is a page fault, the disk is accessed; whenever the page fault replaces a dirty page, the disk is accessed a second time.

1.18 I/O Management

1. By making the I/O instructions illegal to execute when not in kernel or supervisor mode, user programs can be prevented from accessing the devices directly.

2. Once OS initiates an operation on the device it must poll continuously since the OS does not know when the device will actually respond and want to initiate transfer.

3. A CPU that sustains 300 million instructions per second and average 50000 instructions in the OS per I/O operation.

A memory backplane bus capable of sustaining a transfer rate of 100 MB/sec. SCSI-2 controllers with a transfer rate of 20MB/sec and accommodation up to seven disks. Disk drives with read/write bandwidth of 5MB/sec and an average seek plus rotational latency of 10ms.

If the workload consists of 64-KB reads and the user program needs 100000 instructions per I/O operation find the max sustainable I/O rate and the no of disks and SCSI controllers required. Assume that the reads can always be done on an idle disk if one exists.

Max I/O rate of CPU= $300*10^6 /(50+100)*10^3 = 2000$

Each I/O Transfer takes 64KB, so Max I/O rate of bus = $100*10^6/(64000)=1562$

Thus the bus is the bottleneck.

Time per I/O at disk = Seek time + Transfer time = 10ms + 64KB/5MB= 22.8ms

Number of I./O's per second = 43.9

Thus the no of disks req=1562/43.9 = 36

4. Early systems used programmed I/O. What impact does that have on multi-programming?

   Programmed I/O is where the process performs a busy waiting check until the I/O operation has finished. For example,

   ```
   perform some I/O;
   while (I/O not finished)
   {
     check if I/O has finished;
   }
   ```

   This type of busy waiting consumes a large amount of CPU time. One of the primary aims of multi-programming is to interleave CPU execution and I/O. However with multi-programming the CPU execution that occurs during I/O is NOT USEFUL.

   Using interrupt driven I/O, the process that asks for some I/O operation to be performed, will be taken off the CPU until it the I/O operation is complete. During that time the CPU can be given to another process that will carry out some useful work.

5. With DMA, intelligent peripheral devices read data from, or write data to, main memory without the intervention of the CPU. DMA is nearly always done to operating system buffers, and not to buffers within a process. In a couple of paragraphs, describe the various reasons why operating system buffers are used with DMA.

   Here, buffers meant buffers owned by operating system in main memory.

   DMA is generally done for whole blocks, e.g from disk devices. Processes generally don't request I/O in whole-block sizes: they might ask for any amount of data e.g., 10 bytes, 50000 bytes etc. Processes of course don't know the actual block size.

   Therefore the OS must buffer the incoming block, and then copy part of the block into the process' memory space. Ditto if the process asks for data spanning 2 blocks: the OS must get both blocks and then do the copying.

   Another reason for buffering in the OS: the OS can cache recently requested blocks, and give faster access to the data if the blocks are heavily used. Finally, some I/O isn't done specifically for processes. For example, pageouts are not done at a process' behest, but are caused by the OS.

6. Synchronous data transfer can employed between processor and peripherals which are located in the same computer or which are in proximity. They can share common clock.

7. Average seek times of the disks are advertised. However, depending on the application the actual seek time will be 25% to 33% of the advertised one because of locality of disk references.

8. Asynchronous data transfer does not demand common clock.

9. Polling can be called as processor initiated data transmission where processor sends a request to device to transfer the data. The device processes this request and sets device ready signal till which point the processor will be waiting or blocked.

10. Polling also wastes CPU time.

11. Wait states are used to obviate polling. For example a disk drive might assert the wait signal until it is positioned its R/W head and then it de-asserts such that CPU can transfer data. This though do not stop CPU waiting but it makes programmers life simpler.

12. Interrupts are two types 1. HW and 2. SW (signals). HW ones arrive from devices where as signals may arrive from users, programs.

13. HW interrupts further can be classified as External and internal. External interrupts arrive from external I/O devices. Whereas internal ones occur entirely within CPU (Example timer which is used to allocate CPU to different tasks). Devide by zero exception, arithmetic overflow page faults, invalid instruction codes all comes under this internal interrupts category.

14. Interrupt service routine is executed only after completion of the current instruction. Otherwise we have to save many registers as well as state information in control unit. Thus only after completion of current instruction service routine is started when an interrupt arrives in which case only PC value has to be saved.

15. Moreover, when a task is running a interrupt is arrived after t1 sec of starting time slice b sec and interrupt routine takes t2 secs then after completion of service routine this task is going run for b-t1-t2 sec.

16. Vectored interrupts are the ones which send address (either partial or full) of service routine along with interrupt; whereas non-vectored ones will not send any address (here CPU service routines are available are fixed locations).

17. Service routine address is sent to CPU via systems data bus.

18. Very commonly handler routine first disables any further interrupts and clears current interrupt (In some m/c's CPU itself clears this while handler routine is accessed). Before loading previously saved PC value

from stack to PC the handler routine does restored other state information and enables previously disabled interrupts.

19. Daisy chaining is used to make a set of device with some priority among themselves to share interrupt. SCSI devices uses this concept. Here, irrespective of which device has sent interrupt request the high priority device will be given preference.

20. The vector address what every device sends to CPU may not be same.

21. Main drawback of DMA are 1. Yet CPU has to do some work for I/O. 2. Multiple transfers require separate DMA transfers. 3. Some times data has to be processed which DMA can not do. These shortcomings are addressed by I/O processors which are also called as I/O controllers, channel controllers, peripheral processing units (PPU's).

22. An I/O processor manages multiple devices which are connected to separate I/O bus.

23. The I/O instructions given to I/O processor are called as commands. These commands can be classified as block transfer commands, processing commands which does operations such as arithmetic, logical on data before transferring data to cpu, and control commands.

24. USB port supports 127 devices connected to a single port. USB devices require addresses to differentiate them.

25. In USB, token packets are used to initiate data transfer.

26. USB transmits data in packets.

27. USB version 1.1 supports 1.5Mbps where as 2.0 supports 480Mbps.

28. USB communication can be considered as serial communication.

29. RS-232-C supports nine signals, RTS, CTS, TD, DTR, DSR, RD, CD, RI, and ground.

30. RI (ring indicator) is needed in modems which are set to answer incoming calls.

31. Both RS-232-C, RS-422 uses differential voltage while sending data.

32. RS-232 terminals uses 25 pin connector employs serial communication.

33. Memory mapped terminals will not communicate to computer via serial lines; they are integral part of the computer itself. Typically in IBM PC addresses 0xB0000 for monochrome and 0xB8000 for color displays.

34. Each character occupies more than one byte in video RAM as attributes are also stored along with the character code. For example in some m/c's 2 bytes are used for every character in video RAM.

35. In Bit-mapped terminal every bit in the video RAM controls one pixel in the screen.

36. Each I/O bus controller may have DMA processor.

37. If the system uses caches then DMA transfer may becomes little efficient as processor can proceed without accessing memory all the time during which period DMA can use available memory bandwidth.

38. Cache in the system also leads to stale data problem when we use DMA as DMA directly transfer data to memory.

39. Another main difficulty in virtual memory systems with DMA is that whether DMA should work at virtual address level or physical address level?.

40. Overlapped seeks are done by controllers or device driver SW in which seeks on two or more drives are carried out at the same time. While writing on one drive reading will be continued from the other drive here.

41. No need of seek or rotational optimization in RAM disks.

42. RAM disks are especially used to store frequently used programs.

43. Device mounting can not be done in MSDOS.

44. Memory mapped I/O all can use either polling or interrupt strategy.

45. An individual process will execute sequentially across the processor and the I/O device.

46. In polling device driver itself does all the necessary task.

47. If API among different device drivers is same then application programmer life becomes easy. However, each driver implementation is specific to the device.

48. Main motivation of interrupts in HW is to eliminate the need for the device driver to constantly poll the controller status register.

49. When a process makes an I/O request context switching takes place. Only after satisfying I/O request the process waits. If it is not implemented like this, there is a danger of the process working on trash or old value of the object (say x in the following code).

```
x=10;
read(deviceX, "%d", x);
Y=f(x)
```

50. The average time to execute a process is much less with interrupts than it will polling.

51. In memory mapping of devices, the devices are associated with logical primary memory addresses (don't confuse virtual addresses) instead of having specialised device addresses.

52. Memory mapped I/O reduces the no of instruction types in the processor.

53. Buffering is keep slower I/O devices during times when a process is not requiring I/O operation. Also, buffering is to explicitly overlap a process's use of the CPU and its devices.

54. In some disks, homing command is available with which the head can quickly return to track 0 or last track. Such that Scan or cscan disk arm scheduling algorithms can perform better.

EXERCISE

1. Assume that the number of clock cycles for a polling operation — including transferring to the polling routine, accessing device, and restarting the user program, is 400 cycles, and that the processor executes with a 500 MHz clock. Determine the fraction of CPU consumed for the following three cases assuming that you poll often enough so that no data is ever lost and assuming that the devices are potentially always busy.

a. The mouse must be polled 30 times per second.

b. The FDD which transfers 16-bit data at data rate of 50KB/sec. No data transfer can be missed.

c. The HDD which transfers 4-word chunks at 4MB/sec. Again no transfer can be missed.

Answer:

a. Clock cycles for polling = 30*400=12000 cycles per second

Fraction of the processor cycles consumed = 12000/500*1000000=0.002%

b. For FDD, the rate at which we must poll=50KB/2bytes = 25Kpolling accesses/sec

Cycles for second polling = 25K*400

Fraction of the processor cycles consumed=25K*400/(500*1000000) = 2%

c. For HDD, the rate at which we must poll = 4MB/16bytes = 250K polling accesses/sec.

Cycles for second polling = 250K*400

Fraction of the processor cycles consumed=250K*400/(500*1000000) = 20%

In the first two cases polling overhead is not much, whereas in the third one 20% cycles of the processor is spent on polling.

2. Consider a system with 500 MHz clock. The HDD which transfers 4-word chunks at 4MB/sec. No transfer can be missed. Interrupt driven I/O is used. The overhead for each transfer including the interrupt is 500 clock cycles. Find the fraction of the processor consumed if the HDD is only transferring 5% of the time.

Answer:

For HDD, the rate at which we must transfer = 4MB/16bytes=250K polling accesses/sec.

Fraction of CPU consumed = 250K*500/(500*1000000) = 25%

Cycles for second polling=250K*500

Assuming that the disk transferring data 5% , then the fraction of the processor consumed on average=25 % of 5%=1.25%

3. Suppose we have processor with 500MHz clock rate and HD as above. Assume that the initial setup of a DMA transfer takes 1000 clock cycles for the processor and assume the handling of the interrupt at DMA completion takes 500 clock cycles for the processor. The HD transfer rate is 4MB/sec and uses DMA. If the average transfer from the disk is 8KB what fraction of the 500MHz processor is consumed if the disk is actively transferring 100% of the time. Ignore any impact from bus contention between the processor and DMA.

Answer:

Each DMA transfer takes = 8KB/4MB/sec=$2 * 10^{-3}$

So if the disk is constantly transferring, it requires

$(1000 + 500)/(2*10^{-3})=750*10^3$

Since the processor runs at 500MHZ,

Fraction of processor consumed $750*10^3/(500*10^6)$ = 0.2%

4. In virtually all systems that include DMA modules, DMA access to main memory is given higher priority than processor access ot main memory. Why?

Or

Cycle stealing occurs when the I/O processor and the CPU try to access the same memory module or the same bus simultaneously. Why does the I/O processor normally get priority?

Answer : A major aim of multi-programming is to achieve efficient use of the computer systems

resources. This means you want the CPU executing instructions, the printer printing documents, the display displaying output and the disk drive storing information. That means you want interleaved I/O and CPU execution.

I/O devices are very, very slow when compared to the CPU. The CPU will want to access memory and the bus much more regularly than I/O devices of any description.

I/O is given precedence because you want the I/O device to use the memory or the bus and then to start doing its I/O operation. Once serviced the I/O device will take a long time before asking for the bus again. During this period the CPU can be executing.

5. Suppose a process makes a request to transfer 500 sectors from a disk. Consider two possible approaches: a DMA controller that reads one sector and interrupts the CPU after reading it; and an I/O processor that handles the operations for all 500 sectors.

Suppose it takes 0.5ms to respond to an interrupt and issue another read. How much time does the CPU save if the second approach is used? How many instructions could it have executed in this time? Assume the CPU can execute one instruction in 500 nanoseconds.

1,000,000 nano-seconds = 1 milli-second

Using the DMA controller the following will happen 500 times

I/O operation occurs

interrupt is generated

interrupt must be handled and process ask for more I/O

Answer:

The first two steps will not take any CPU time and the last step will consume 0.5ms. That means using the DMA controller will result in the following amount of CPU time being used

500 x 0.5ms = 250ms

Using the I/O processor this 250ms of CPU time is not used. Given that the CPU can execute one instruction every 500 nanoseconds. This means that the CPU can execute 2000 instructions for every millisecond which means the CPU could execute 500,000 instructions in 250ms

6. Checking for interrupts is normally part of the instruction cycle. What is the advantage of doing it this way instead of using a separate machine language instruction?

Answer:

Most computers use the same basic instruction execution cycle.

This cycle is built into the hardware and as a result goes relatively quick.

What would happen if the check for an interrupt was a separate machine language instruction?

That would imply that the programmer could control when interrupts are handled. What does this imply?

- if the user decides not to do it, it won't get done
- which implies the handling of interrupts will slow down a great deal
- also checking interrupts as a separate instruction may be slower than having it handled automatically by the hardware

7. Explain the motivation behind disk arm scheduling, and why the elevator algorithm is often used. To which software layer in I/O does disk arm scheduling belong? Justify your answer in a couple of paragraphs.

Answer: Motivation: to minimize arm movement which is very slow. Achieved by reordering pending arm motion requests so minimize overall seeks. This depends on there being a queue which can be reordered.

The elevator algorithm is one such queue reordering algorithm. The queue is reordered so that the head starts at one end of the disk, and moves towards the other end, servicing requests in that order, until there are no more requests in that direction. The arm then reverses direction, and services requests the other way. One nice property is, given any collection of requests, the upper bound on the motion is fixed at exactly 2 * the number of tracks.

The algorithm is only dependent on the number of tracks/cylinders on a disk. The algorithm doesn't need to know the disks commands in to reorder the queue. Therefore, it is possible to keep the elevator algorithm in the device-independent part of the I/O stack. This would allow it to be shared across all disk devices.

8. Why is it important to try to balance file system I/O among the disks and controllers on system in a multitasking environment?

Answer: By spreading disk activity out amongst multiple disks and controllers you can have many different disk accesses all being served at once. A single disk/controller can only handle one request at a time. Spreading disk activity out should improve efficiency.

9. What is the average time to read a 512-byte sector for a typical disk rotating at 3600 RPM? The advertised average seek time is 12ms, the transfer rate is 5MB/sec and the controller overhead is 2ms. Assume that the disk is idle so that there is no waiting time. Repeat the same for disk rotating speeds of 5400RPM and 7200RPM.

Answer: Average disk access time = average seek time + average rotational delay + transfer time + controller overhead

= 12 ms + 0.5 rotation/3600RPM + 0.5KB/5MB/sec + 2ms= 12ms + 8.3 ms + 0.1ms + 2ms=22.4ms

For 5400 RPM:

= 12 ms + 0.5 rotation/5400RPM + 0.5KB/5MB/sec + 2ms= 12ms + 5.6 ms + 0.1ms + 2 ms=19.7ms

For 7200 RPM

=12 ms + 0.5 rotation/7200RPM + 0.5KB/5MB/sec + 2ms=12ms + 4.15ms + 0.1ms + 2ms=18.25 4ms

1.18.1 Direct Memory Access

Disks

Standard floppy and hard disks use magnetic recording technology. The disk surfaces are divided into concentric *tracks* (hundreds per surface on a hard disk), and each track is divided into sectors (typically a few tens per track). The position of tracks and sectors are marked on the disk by magnetically-recorded *formatting* information identifying each sector with its track and sector number. Magnetically-recorded data is written to or read from the data portion in each sector, and a sector will hold a fixed number of bytes, usually 512.

The *access time* of a disk is the time it takes the head to get into position above the required sector, and is made up of two parts, the track seek time (the time it takes to move the head to the required track), plus the rotational latency to get to the required sector. Seek times are typically a few milliseconds for hard disks, and 100's ms for floppies. Floppies usually rotate at 360 rpm, and hard disks typically at 3600, 5400 or 7200 rpm, so the *average* rotational latency is 83ms for a floppy and, at 3600 rpm, 8.3ms for a hard disk. Data transfer rates are largely determined by how closely the bits can be packed on the disk surface (the *recording density*) and are typically 500 kbit/sec for a floppy and 10 Mbit/sec or more for a hard disk.

Disk Controllers

All disk drives require controllers, to interface the disk electronics to something the rest of the computer can understand. Hard drives usually contain quite sophisticated disk controllers built into the disk drive unit. The two common standards for communication with these controllers are *IDE*, and *SCSI* : the IDE standard is suitable where the disk is inside the computer case, while SCSI is rather more flexible, and is intended to be used for internal and external drives. An IDE bus is very similar in structure to the processor's own memory bus it has data, address and control signals so very little hardware is required to interface the processor to the IDE bus. In the case of SCSI, a SCSI bus controller is needed, through which the CPU operates the SCSI bus. The CPU interface of a floppy or IDE disk controller is the usual I/O controller bank of registers, typically eight, mapped into the CPU I/O bus address space (if the CPU has a separate I/O bus), or memory-mapped. The registers are used to transfer data bytes or words, and control and status information, such as desired track and sector number, current track and sector number, etc. The individual functions carried out by the disk controller for the CPU are quite complex, and one register, written by the CPU, is usually designated the *command register* the byte written to this register determines which complex sequence the controller carries out next.

Typical commands could be:

Seek n Move the head to track n

Read Sector m Wait for sector m to rotate under head, then read the data from it

Write Sector m Wait for sector m, then write new data into it

Format Track Initialize the track, writing the sector marks and identifying information

Using a disk controller

If the CPU wishes to read the data from a particular sector (the filing system will provide the mapping from parts of files to physical disk sectors), it must first issue a *seek* command to the required track. 10's or 100's of ms later, the seek will complete, and the disk controller will interrupt the CPU. Next the CPU issues a *read sector* command for the required sector.

Again, there may be a long delay for the rotational latency, so the controller will interrupt again when the head is above the requested sector. Now the data must be transferred, a byte at a time, and at floppy rates (*eg* 500kbit/sec), this is one byte per 16 *micro* seconds. Assuming a status register in the disk controller (at address fdstat) contains a bit indicating that the next byte is ready, and the disk controller data register is at address fddata, R2000 code to do the transfer for a *read sector* command might be (with register $4 pointing on entry to the base of the 512 byte buffer for the data in memory):

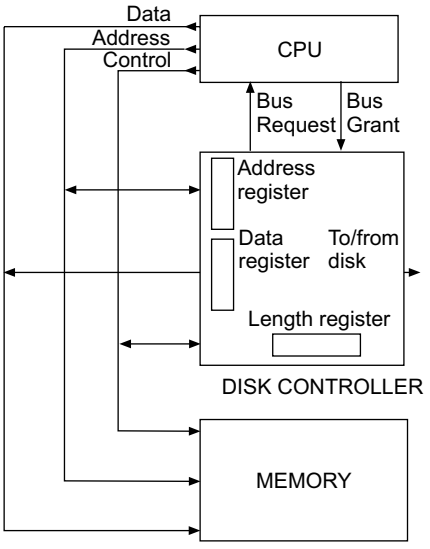
```
addui $8,$4,512 # $8 points just after end of buffer
la $10,fdstat
la $11,fddata
fdloop: lbu $9,($10) # get the status
andi $9,$9,fdmask # mask to get required bit
beqz fdloop # wait until next data byte ready
lbu $9,($11) # get data byte
sb $9,($4) # put it in buffer
addiu $4,$4,1 # update pointer
bne $4,$8,fdloop # repeat until $4 reaches $8
```

Here, the CPU polls the status bit for each data byte transfer. The CPU is tied up for 512 * 16_s, i.e., 8ms, during which time interrupts must be disabled. An interrupt-driven alternative might be possible, but the interrupt response time of many processors is barely fast enough: to process an interrupt request, save the interrupted process's context, and to restore context on return from the interrupt might take several microseconds. Ideally, what we want is to relieve the CPU entirely of the task of moving the data between the buffer in memory and the disk controller, and this can be done if we arrange for the disk controller to be able to access the memory directly itself.

Direct Memory Access

The disk controller needs to be able to transfer a block of bytes to or from memory. For this, additional hardware is needed, called a *direct memory access* (DMA) controller. The DMA controller includes an address register, holding the address in memory of the next byte to be read/written, a data register, through which data is transferred from/to memory, and a length register holding the number of bytes still to be transferred. To read a byte from memory, the DMA controller outputs the contents of the address register onto the memory address bus, and asserts the *read signal* in the memory control bus. The memory responds with the data byte, which the DMA controller copies into the data register, to be sent as a serial data stream to the disk. The address register is then incremented, ready for the next byte, and the length register decremented. The DMA operation is complete when the length register reaches zero. Writing to memory is similar, except that the DMA controller *outputs* the data bytes onto the memory data bus.

To read or write a sector, the CPU must first issue the command to seek to the correct track as before, then write initial values into the DMA controller address register (i.e. a pointer to the data buffer in memory) and the length register, and then issue the disk command *read sector m* or *write sector m*. The disk operation now proceeds without further CPU intervention— once the head is above the required sector, that data is transferred between the disk and the memory buffer by the DMA controller, and the CPU is interrupted only when the operation is complete.



Bus Arbitration

We now have two devices the CPU and the disk DMA controller, both controlling the memory bus — how do we ensure that they do not both try to use the bus at the same time? This is achieved with a circuit in the CPU called the *bus arbiter*, which arbitrates between requests by the CPU to use the memory bus and requests by the DMA controller. The DMA controller is connected to the arbiter by two wires, a signal to the arbiter called *bus request* and an acknowledgement signal from the arbiter called *bus grant*.

When the DMA controller has a word to transfer, it asserts *bus request.* When the CPU has finished any momory access already in progress *bus grant* to the DMA controller will be denied; otherwise granted.

1.19 Serial Communication

This is the barest possible means available in every PC to communicate with each other. IBM PC will be having two ports COM1: (address 0x3F8), COM2: (0x2F8). Normally this communication is used between computer and devices outside. It is evident that CPU does not communicate serially with such devices. Rather it communicate through UART interface which takes care of converting serial data to parallel data and *vice versa.* Prominently asynchronous and synchronous serial communication techniques are in wide use. For example, modem is a good example for the first type.

1.19.1 Asynchronous Serial Communication

The connected devices do not have common clock, must synchronize their data transfer. This is carried out by making both sides to agree on some transmission parameters

before data transfer. They are: 1. Speed (mentioned in either in data rate in bits per second or baud rate[2]/signaling rate) 2. No of data bits per transmission, 3. Whether uses parity (odd parity or even parity) or not. 4. No of stop bits and start bits.

Usually when the transmission line is idle its value is logic 1. Usually least significant bit of the data is transmitted first. After sending the last bit, parity bit will be send till that point receiver will be waiting. Usually 1, 1 1/2, 2 stop bits are used. The width of the start bit is same as any data bit.

N81 setting in which one stop/start bits with 8-bit data (including parity) is the one which is commonly used in many modem transmissions. Here percentage of overhead is 20.

1.19.2 Synchronous Serial Communication

Especially this method is proposed to reduce the transmission overhead. This is achieved by sending a block of data rather then byte by byte. It does appends header and trailer information such as source address, destination address, checksum (for error detection), start/stop bit sequences and then the resulting data unit known frame is sent. Though this additional information also overhead but normally this is at lower percentage than asynchronous communication.

High level Data Link Control (HDLC) is a common synchronous communication transmission standard. Here total overhead bits are 48 bits. If we assume data is 256 bits then the overhead is 15.79%. Even if the data size grows the overhead percentage will be reduce, as overhead bits will not change from 48 bits. This is not true in the case of asynchronous serial communication.

When a byte or series of bytes (frame) is received or transmitted at constant time intervals with uniform phase differences, the communication can be called as synchronous. Bits of a data frame are sent in a fixed maximum time interval. Where as in Iso-synchronous mode the maximum time interval can be varied. Two salient characteristics of this style of communication are:

1. Frames can not be sent at random intervals. Thus no need of handshaking.
2. A clock ticking at a certain rate has to be always there for transmitting serially the bits of all the frames.

[2]A 1200 bps line transmitting 7-bit bytes with 1 parity, stop, start bits can send 120 bytes per second

1.19.3 Universal Asynchronous Receiver/Transmitter (UART)

As mentioned above the CPU always communicates in parallel mode. Thus in order to communicate on serial lines UART's are used which takes the responsibility of converting to serial data stream to parallel stream and vice versa.

The UART is the one which is responsible for generating start, stop, and parity bits as well as removing them. The processor can send control signals to UART to indicate speed, word size, parity, and no of parity bits.

UART employs Transmit Holding Register (THR), Transmit Shift Register (TSR). While one byte is transferred from CPU to THR, the one is TSR is sent on the line. This is one form of double buffering employed to reduce delays.

Example devices using UART's are: keypad, mouse, modem, character input/output devices (terminals).

1.19.4 Serial Communication Standards (RS232-C)

At most only one device can be connected to this device unlike USB port. Serial ports of most of the PC's can transmit data up to 115,200bps. This standards supports nine signals given as:

RTS (Request to send)

CTS (Clear to Send)

TD Transmit Data

DTR Data Terminal Ready

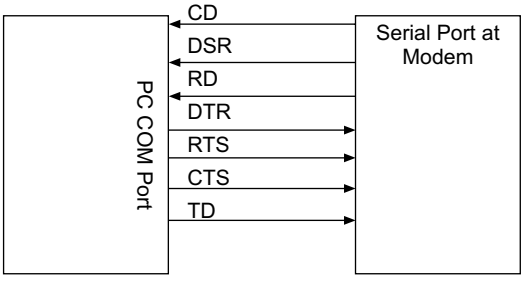
DSR Data Set ready

RD Receive Data

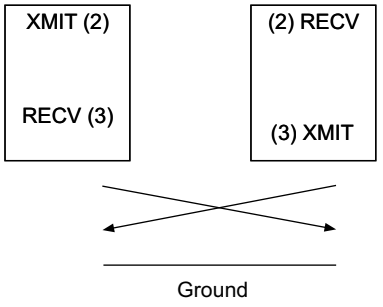
CD Carrier Detect

RI Ring Indicator

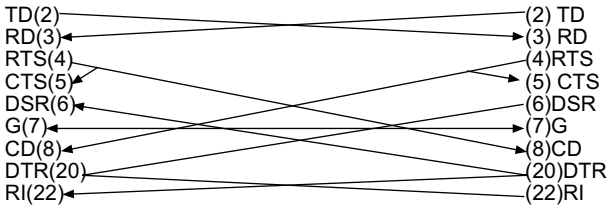
G Ground



Usually RS232-C connector is a 25 pin D-connector (of course 9 pin D-connectors can be also used). At the barest level only 3 circuits are used as shown below and resulting cabling is known as Null Modem.



1.19.5 A complete Null Modem cabling



A maximum separation of 15m at 9600bps is supported in this standard. This uses NRZ-L signalling.

Main drawbacks of RS232C is its limited distance of 15m and if the ground reference pin 7 is different for both sides then undesirable electrical disturbances will be applied to transmitted signal.

RS-449 calls for two sets of connectors: 37-pin for data, control, timing, diagnostics, and a 9-pin for secondary channel. Whereas, RS232C has a single 25 pin connector. RS449 supports both balanced and unbalanced while RS232C supports only unbalanced. A balanced one is in which the signals are carried between the DTEs on a pair of wires. They are sent as a current down on one wire and return on the other; the two wires create complete circuit. In unbalanced one the signal is sent over a single wire with DTEs sharing a common ground. A balanced is less effected by noise.

RS422A and RS423A supports balanced and unbalanced, respectively. RS422A supports 100000bps for 1000m and 10000000bps for 10m whereas RS423A support 3000bps for 1000m and 300000bps for 10m.

1.19.6 The Universal Serial Bus Standards

Unlike RS232-C here we can connect at most 127 devices to a USB port. The data is transmitted in packets. The USB port is much faster than RS232-C port. USB version 1.1 supports 1.5Mbps (3m channel), 12Mbps for 25m channel, where as version 2 supports 480Mbps for 25m channel.

USB bus cable has four wires, one for +5V, two for twisted pairs and one for cable. There will be termination impedance at each end.

Serial signals are NRZI (non return to zero) type.

USB supports data transfer of 4 types given as:

1. controlled data transfer support guaranteed bus access
2. bulk data transfer support low priority large data transfer
3. interrupt driven support periodic bandwidth
4. iso-synchronous transfers which support guaranteed bandwidth

USB employs device polling. USB controllers are further classified as UHCI (universal host controller interface), OHCI (open host controller interface). Communication overhead is more on host CPU in UHCI.

The USB standard specifies four types of packets such as token, data, handshake, and special.

The token packet is used to initiate data transfers. It specifies address (ADDR), direction specifier (packet identifier PID), end of packet (ENDP), and CRC checksum.

Data packets contains no address. Data field value can be upto 8192 bits. Handshake packets are to either send ACK or NACK (negative ACK).

■ **Problem** A computer sends 0.5KB(6144 bits) of data to one of its USB peripherals.

a. Show the packets sent by the computer to perform this transfer. What is the total no of bits transferred?

b. What percentage of the bits transmitted is overhead?

c. How many bits would be required to send the same data using an RS232C serial ort with no parity, 8 data bits and 1 stop bit is used?

■ **Answer:**

According to the above discussion, a data packet in a USB standard can have at most 8192 bits. Our data is only 6144 bits. Thus, no of overhead bits=8+16=24.

Percentage of overhead=24/(6144+24)=0.38%

In the case of RS232C for every 1 byte 1 start bit and 1 stop bit has to be sent. Thus overhead bits = 2*0.5*1024=1024bits

Percentage of overhead = 1024/(6144+1024)=14.3%

1.20 Pipelining

1. For the following code fragment, 2 stage pipeline is proposed; 1'st stage for multiplication (10 ns) and second 2'nd stage for addition (10 ns) is required. Then how much time it takes to complete.

for I = 1 to 100 do A[I] = B[I]*C[I]+D[I]

A. 2000 ns B. 1010 ns

C. 1020 ns D. 2010 ns

Answer: Pipeline Time = (n – k + 1)*cycle time = (100 – 2 + 1)*10ns = 1010ns

2. Repeat the above problem assuming latching in pipeline require 2 ns and fetching time is ignored.

A. 2000 ns B. 1012 ns

C. 1212 ns D. None

Answer: Pipeline time = (100-2+1)*12ns=1212ns

3. A pipeline has k stages with time delays Ti, i=1,..,k and each stage has a latch with latch access time Tl then the clock period of this linear pipeline is given as

A. $\max\{T_1,T_2,\ldots T_k\} + T_l$

B. $\min\{T_1,T_2,\ldots T_k\} + T_l$

C. $\max\{T_1,T_2,\ldots T_k\}$

D. None

4. A pipeline has 4 stages with time delays 60 ns, 50 ns, 90 ns and 80 ns. The interface latch has a delay of 10 ns then cycle time of this pipeline is

A. 90 ns B. 100 ns

C. 60 ns D. None

Answer: Cycle time = max{60, 50, 90, 80}ns + 10ns=100ns

5. For the same problem clock frequency of the pipeline is

A. 10MHz B. 13MHz

C. 20MHz D. None

Answer: Clock Frequency = 1/cycle time = 1/100ns = 10MHz

6. A pipeline has 4 stages with time delays 20 ns, 20 ns, 100 ns and 40 ns then in order to get maximum performance which stage can be divided?

A. Stage with 20 ns B. Stage with 100ns

C. Stage with 40 ns D. None

Answer: The stage which has maximum delay such that variance in stage delays is minimum.

7. A pipeline has 5 stages with time delays 20 ns, 20 ns, 50 ns, 50 ns and 40 ns then in order to get maximum performance which two stages can be combined?

A. First two B. 3'rd & 4'th

C. 4'th & 5'th D. None

8. A RISC processor has a pipeline with 3 stages with delays 40 ns, 80 ns and 50 ns, respectively. What is the clock period and steady state speedup of the pipeline? If a non-pipelined CPU can process an instruction within 160ns what is the actual steady-state speedup of the pipeline?

Answer:

Cycle time = max{40,80,50}=80ns (which is same as the pipeline time)

Non-Pipelined time=160ns

Steady state speed-up = Non-pipelined time/pipelined time=160ns/80ns=2

9. An non-pipelined processor has a cycle time of 25 ns then what will be the cycle time and pipeline latencies if the pipelined version of processor has 5 equally divided pipeline stages with 1ns latency latches? What is going to happen if the stages are made as 50 equally divided stages?

Answer:

5 Stage one

Cycle time = 25/5 + 1 = 6 ns

Pipeline latency = 6*5= 30ns

(This is general rule of thumb is pipeline latency = cycle time * no of stage)

50 Stage one

Cycle time = 25/50+1=1.5ns

Pipeline latency=1.5*50=75ns

10. An non-pipelined processor with a 25ns cycle time is divided into 50 stage with latencies 5,7,6,3, and 4ns. If the pipeline latch latency is 1 ns what is the cycle time of the resulting processor?

Answer: max{5, 7, 6, 3, 4} +1ns = 8ns

Throughput = 1/8ns = 125MHz

Pipeline latency = 5*8 = 40ns

11. Given an non-pipelined processor with 10ns cycle time and pipeline latches with 0.5ns latency, what are the cycle times of pipelined versions of the processor with 2, 4, 8 and 16 stages if the data path is evenly divided among the stages?. What is the latency of each pipelined versions of the processor?.

Answer:

Cycle times are: 5.5, 3, 1.75 and 1.125ns

Pipeline Latencies are: 11, 12, 14 and 18ns

12. For the above pipeline how many stages are required to get cycle time of 2ns? 1ns?

Answer:

No of stages = 10/(2 – 0.5) = 6.67 is 7 stages if the required cycle time is 2ns

No of stages = 10/(1 – 0.5) = 20 stages if the required cycle time is 1ns

13. For the above problem what is the minimum cycle time achievable with a 4-stage pipeline if additional logic is assigned to the final stage to balance the additional latency of the pipeline latches in the other stages?

Answer:

Total latency = 10 +1.5 = 11.5ns (assuming the last one doesn't have any latch)

Thus clock time = 11.5/4 = 2.875

14. A pipelined processor has stages with latencies 2,3,4,7,3,2, and 4 with 1ns latency latches. What is the minimum cycle time that can be achieved?

Answer: 8ns

15. If the first two stages are combined, and 5'th and 6'th is combined giving 5 stages then what is the latency?

Answer: 40ns

16. If we limit to two stages, what is the minimum cycle time?

Answer :

2,3,4,7 and 3,2,4 as identified to be combined to get two stages. Their latencies are 16 and 9ns. Or reverse is also acceptable. i. e 2,3,4 and 7,3,2,4. Thus their latencies are 9 and 16ns. Only after the first one latch is required.

Thus cycle time is 16ns.

Pipeline latency is 32ns.

17. An arithmetic pipeline contains 4 stages with 60, 70, 100 and 80 ns cycle times and the interface registers have a delay of 10ns. Calculate speedup (largest possible)

Answer:

Cycle tme=100+10=110ns

Non-pipelined processor time req = 60 + 70 + 100 + 80 + 10 = 320

(assuming after calculation in last stage results are stored in in registers).

Speedup = 320/110 = 2.9

18. A non-pipelined system takes 50ns to process a task. The same tack can be processed using 6 segment pipeline with a clock cycle of 10ns then calculate the speedup ratio assuming no of tasks are 100? What is the max speedup that can achieved?

Answer: (100*50)ns/((6+99)*10ns)=4.76

Max possible speed-up=6

19. A arithmetic pipeline has 4 stages with time delays 50,30,95 and 45ns. The interface registers delay is 5ns then calculate how long it takes to complete 100 tasks and how can we reduce the total time about one half?

Answer

Cycle time = 95 + 5ns = 100ns

Total time required = 100*100 = 10000ns

Divide 3'rd stage into two stages with latencies 50 and 45 ns and then time will be reduced to almost half.

20. A three stage pipeline is used to execute the code segment

For I=1to n do {X[I] = ((A[I].B[I]) + C[I])*D[I]}

The first stage multiplies A[I]*B[I] in 20ns, then the second stage adds this product to C[I] in 15ns and in the last stage final multiplication with D[I] is done within 20ns. What is the clock period of this pipeline? What is the steady state speedup for this pipeline? For what values of n does the pipeline produces results more quickly than a non-pipelined unit? For what value of n is the speedup exactly 1.5? Assume latch time is 5ns.

Answers:

Clock cycle = 20ns

Steady state Speed-up = (20 – 5 + 15 – 5 + 20)/20 = 2.25

(assuming last stage is not having latch)

Pipeline delay = 3*20=60ns

In a non-pipelined version every operation requires 45ns.

Therefore for n greater than 2 then pipeline is better than non-pipelined one.

1.5 = n*45/((3+n-1)*20)

n = 2

21. A CPU has a clock period of 25ns. Some instructions can be removed from its instruction set to form a second CPU with a clock period of 24ns. These instructions comprise 1% of typical code and must be replaced by four instructions each.

A. Which CPU is better?

B. What percentage of typical code would be removed instruction have to comprise in order to for the two CPU's to have same performance?

C. How many instructions would have to be needed to replace each of the remove instructions for the two CPU's to have same performance?

D. For what value of clock period of the original CPU would the two CPU's have the same performance?

E. For what value of the clock period of the second CPU would the two CPU's have the same performance?

Answer:

Assume every instruction requires same number of cycles to complete (let c)

In the first CPU thus requires 25c ns to complete any instruction.

A. Average time required for second CPU=0.99*24c+0.01*(4*24)c=24.72c ns

Therefore second CPU is better.

B. 25c = (1-x)*24c +x*4*24c

x = 0.01388888

That is 1.38888% of typical code has to be removed

C. 25c = 0.99*24c +0.01*x*24c
x = 5.16
i.e 6 instructions to be replaced.
D. 24.72 ns
E. 25c = 0.99*xc + 0.01*4*xc
x = 24.27ns

22. Repeat the above problem with a clock period 15ns and a reduced CPU with a clock period of 12ns. Instructions removed are 2% and are emulated through 6 instructions each.

Answer:

Assume every instruction requires same number of cycles to complete (let c)

In the first CPU thus requires 25c ns to complete any instruction.

A. Average time required for second
CPU = 0.98*12c + 0.02*(6*12)c = 13.2c ns
Therefore second CPU is better.
B. 15c = (1-x)*12c +x*6*12c
x = 0.05
That is 5% of typical code has to be removed.
C. 15c = 0.98*12c + 0.02*x*12c
x=13.5
i.e 14 instructions to be replaced.
D. 13.2 ns
E. 15c = 0.98xc + 0.02*6*xc
x=13.63 ns

23. A CPU has a clock period of 20ns. It is possible to remove some (2%) instructions to make clock period as 18ns. This 2% instructions can be realised with 3 left over instructions in assembly. Will there be any advantage?

Answer:

Assume every instruction requires same number of cycle to complete (let c) In the first CPU thus requires 20c ns to complete any instruction. Time required for any instruction in second CPU can be calculated as:

0.98*18c+0.02(3*18c) = 18.72c

There fore second CPU requires less time to execute any instruction.

24. Calculate execution time of the following instructions in a 5-stage pipeline (all operands are registers).

ADD r1,r2,3
SUB r4,r5,r6
MUL r8,r2,r1
ASH r5,r2,r1
OR r10,r11,r4

Answer: (A instruction is said to be issued when it enters into execution stage)

Dependencies MUL, ASH depends on ADD

If ADD issues at n'th cycle then

SUB issues at n+1'th cycle

MUL issues at n+3'rd cycle

ASH issues at n+4'th cycle

OR issues at n+5'th cycle

Then, execution time = 5 + 5 = 10 cycles

25. Calculate execution time of the following set of instructions assuming a 5-stage instruction pipeline (all operands are registers).

ADD r3,r4,r5
SUB r7,r3,r9
MUL r8,r9,r10
ASH r4,r8,r12

Answer:

Dependencies SUB depends on ADD

If ADD issues at n'th cycle then

SUB issues at n+3'rd cycle

MUL issues at n+4th cycle

ASH issues at n+7'th cycle

Execution time = 5 + 7 = 12 cycles

26. Re-order the above set of instructions in the following manner. Calculate execution time.

ADD r3,r4,r5
MUL r8,r9,r10
SUB r7,r3,r9
ASH r4,r8,r12

Answer: If ADD issues at n'th cycle then

MUL issues at n+1'th cycle

SUB issues at n+3'rd cycle

ASH issues at n+4'th cycle

Execution time = 5 + 4 = 9 cycles

27. Repeat the above problem assuming bypassing is available (Note: bypassing reduces dependency delay to 2 in the case of non-branch instructions).

Answer:

If ADD issues at n'th cycle then

MUL issues at n + 1'th cycle

SUB issues at n + 2'rd cycle

ASH issues at n + 3'th cycle

Execution time = 5 + 3 = 8 cycles.

Note

- The pipeline is actually slower than the non-pipelined arithmetic unit due to delays of the latches at the end of each stage.
- Maximum speedup occurs when each stage has the same delay.
- If the no of stages increases HW cost, area increases and pipeline delay also increases.
- If a instruction enters into execute stage it is said to be issued.

1.21 Instruction Hazards

As long as instructions are independent then the pipeline can show throughput of 1/cycle time. However, there are many factors which limit the pipeline throughput such as dependencies, branch instructions, etc. The instruction hazards are classified as follows:

Read After Read (RAR): ADD r1,r2,r3
SUB r4,r5,r3

RAR hazard does not cause any problem. Thus both the instructions can be executed sequentially.

Read After Write (RAW): ADD r1,r2,r3
SUB r4,r5,r1

Here, a instruction uses a register which is modified by previous instruction as shown above. These hazards are called as data dependencies or true dependencies. When a RAW hazard occurs then the dependent instruction gets stalled till all the operands are available. This can be done by either by HW or optimizing compiler. This is called as pipeline stall or bubble. HW may include no-operation (NOP) into the code.

Write After Read (WAR): ADD r1,r2,r3
SUB r2,r5,r6

Write After Write (WAW): ADD r1,r2,r3
SUB r1,r5,r6

WAR, WAW are called as name dependencies. They do not cause any delays in the pipeline. For a system with large number of registers the chances of these hazards becomes less always the compiler can select other independent registers in the instructions.

1.21.1 Branch Instructions

Another major problem with pipelining is branch instructions. As processor can not determine which instruction to fetch next until the branch has executed. Particularly branch instructions involving some conditions create data dependencies between the branch instruction and the instruction fetch stage of pipeline. For example, for a 5-stage (fetch, decode, read registers, execute, write back) pipeline only after branch instruction completes execute stage PC value is updated thus the next instruction can be initiated in the next cycle. Thus, if there exists a branch instruction is issued on n'th cycle then the following instruction will be issuing at n+4'th cycle. If the pipeline computed the new value of PC before execute stage then branch delay may become 3 cycles.

Branch delays are also called as control hazards. Branch prediction techniques are employed to reduce this delay.

1.21.2 Structural Hazards

These things occur if pipeline HW does not support simultaneous pipeline operations. For example, if write back and register read operations can not be performed simultaneously then delays may occur.

Scoreboarding is used to identify whether registers are available for reading or they are waiting for previous pipeline instruction to be modified.

Result Forwarding (by-pasing) Results of execute stage will be forwarded to previous stages of the pipeline such that these instructions will be allowed without waiting for the result to be written into register file. However, result forwarding of branch instructions may not improve branch instruction delays as the result of branch instructions are not written into register file commonly. Result forwarding reduce the latency of non-branch instruction to one cycle.

Note on RISC Processors

- In RISC processors all the instructions are of same size.
- Number of addressing modes in RISC processors are limited unlike CISC processors.
- RISC processors limit interaction with memory to loading and storing data. Where as CISC processors supports instructions to directly AND (logical) with memory words.
- RISC processors uses extensively instruction pipelining.
- RISC processors contains large number of registers.
- Another unique characteristic of RISC processors is that they contain hardwired control unit unlike CISC processors which contains programmable control unit.
- In order to take care of waiting times (stalls) delayed loading, delayed branches are employed.
- Speculative execution of instructions is employed to take care of delays (Itanium processor handles jumps without any time loss at all).
- RISC processors are usually supported by optimising compilers such that number of stalls are minimised.

- Separate instruction and Data streams are employed to avoid memory access conflicts.

1.21.3 Register Windows

Although many registers are available in RISC processors it may not possible to access all of them at a time. These registers are divided into Global and register windows (where a window is a set of registers). It is acceptable that a set of registers can be common to more than one window. In fact functions communicate parameters, return values through this overlapped windows.

Most RISC processors use about **8 windows**.

Annuling is the process in which results of a instruction are not stored though calculated because of branching.

CISC processors are designed for assembly language programs where as RISC are designed for compiler, high-level language programs.

The same compiled high-level program may require more instructions for a RISC than CISC.

CISC processors are backward compatible (Ex Intel Family).

PowerPC employs a combination of RISC & CISC technology.

CISC program takes less space in memory.

1.22 Solved Questions

1. Explain about Working Set and Thrashing.

Working set means the set of pages (for virtual memory) or memory blocks that is actively used during a limited time window. If the working set exceeds the size of the physical memory, then pages will keep swapping in and out and slow down the overall computer speed (for virtual memory). It will be similar for the cache. If the number of active memory blocks exceeds the total size of the cache, then memory blocks will keep being copied in and out, and slow down the computer performance. This phenomenon is called thrashing.

2. A DMA module is transferring characters to memory using cycle stealing, from a device transmitting at 9600 bit per second. The processor is fetching instructions at the rate of 3 million instructions per second (MIPS). By how much will the processor be slowed down due to DMA activity?

Answer:

In one second: 1200 characters (bytes) being sent to memory.

If DMA is transferring w number of characters, then it takes w * 1/1200 seconds to transfer them.

"3 million instructions per second" processor is slowed down by the time to send 1200 bytes.

That is, to execute 3 million instructions, it now takes 1 second + w/1200 seconds.

e.g. if w = 1200, then 3 MIPS becomes 1.5 MIPS.

3. Design a 2M by 32 bit memory using SRAM chips of size 128K × 1 bit. Give the array configuration of the chips on the memory board, showing all required input and output signals. Is it possible to come up with a design so that the memory system is byte and word addressable?

Answer:

16 x 32 128K x 1 bit chips

17 bits used to address within each chip

4 bits used to select row (using chip select and decoder)

4. A dynamic RAM that has refresh cycle of 64 times per ms. Each refresh operation requires 150 ns; a memory cycle requires 250 ns. What percentage of the memory's total operating time must be given to refreshes?

Answer: In 1ms : refresh = 64 times, memory cycle

$$= \frac{1\text{ms}}{250\,\text{ns}} = 4000\,\text{times}$$

Therefore, % refresh time

$$= \frac{64\times 150\,ns}{4000\times 250\,ns} = \frac{9600\,ns}{1000000\text{ns}} \times 100\% = 0.96\%$$

5. Given an 8-bit data word stored in memory is 11001010. (a) Using the Hamming algorithm, determine what check bits would be stored in memory with the data word.

Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Num	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	*D8= 1*	*D7= 1*	*D6= 0*	*D5=* 0		*D4=* 1	*D3=* 0	*D2=* 1		*D1=* 0		
Check bit					*C8=* 0				*C4=* 1		*C2=* 0	*C1=* 1

Answer :

C1 = EXOR (D1, D2, D4, D5, D7) = 1

C2 = EXOR (D1, D3, D4, D6, D7) = 0

C4 = EXOR (D2, D3, D4, D8) = 1

C8 = EXOR (D5, D6, D7, D8) = 0

(b) If the check bits received when reading the stored word are 0000, what word was read? Show why.

Answer :

0000 EXOR 0100 = 0100 indicates bit position 4 is wrong. That is check bit C4

0000 EXOR 0101 = 0101 indicates bit position 5 is wrong. That is data bit 2.

So, that word read, 11001010, is actually correct.

So, the word read was 11001000, rather than the correct word 1101010.

6. Given a single platter disk with the following parameters: 7200 rpm, 2000 tracks, 600 sectors per track, 10 μsec per track head travel speed, calculate average seek time, average rotational latency time, average request time.

Average Seek time =10μ sec/track * 2,000 tracks * .5 = 10 ms

Average Rotational Latency time = (7200 rpm / 60 sec/min = 120 rps)=1/ 120 rps * .5 = 4.17 ms

Average request time = (Transfer time = 8.33 ms/rev / 600 sectors/track = 13.88μ sec)

Seek time + Latency time + Transfer time = 10 ms + 4.17 ms + .01388 ms = 14.18 ms

7. Assume that we are planning to enrich a computer by two possible changes: making multiply instructions to run five times faster than before, or making memory access instructions to run three times faster than before. We repeatedly run a program that takes 100 seconds to execute. Of this time, 25% is used for multiplications, 45% for memory access instructions, and 30% for other tasks.

A. What will the speedup be if we improve only multiplication?

B. What will the speedup be if we improve only memory access?

C. What will the speedup be if both improvements are made?

Answer:

T1= time before enhancement = 100 sec

T1= $T_{memory} + T_{mult} + T_{other}$

T_{memory} = .45*100 = 45 sec

T_{mult} = .25*100 = 25 sec

T_{other} = 30 sec

A. If we improve only multiplications:

T = 45+ 25/5 +30 = 80 sec

Speed up = 100/80 = 1.25

B. If we improve only memoery access:

T = 45/3 + 25 + 30 = 70 sec

Speed up = 100/70 = 1.428

C. If we improve multiplication and memory access:

T= 45/3 + 25/5 + 30 = 50

Speed up = 100/50 = 2

8. A given C++ application runs for 4 seconds. An improved compiler is released that requires 0.8 as many instructions as the old one, but it increases CPI by 1.2. How fast can we expect the new program with this new compiler and same system?

Answer : Old execution time = I * CPI* clock cycle time = 4 second

New execution time= 0.8*I * 1.2* CPI * clock cycle time = 0.8*1.2*(I * CPI* clock cycle time) = 0.8 * 1.2* 4 = 3. 84

So, the new program is 4/3.84(=1.04) faster than old program.

9. Machine A implements all floating point operations in hardware while machine B implements them in software using integer instructions.

Program P has the following mix of instructions: floating-point multiply: 10%, floating-point add: 10%, floating-point divide: 10%. The rest are integer instructions.

On machine A floating-point multiplication takes 6 cycles, floating-point add 4 cycles, floating-point divide 20 cycles. All integer instructions take 2 cycles.

On macine B all integer instructions take one cycle. Floating-point multiplication takes 30 instructions, floating-point add 20, floating-point divide 50.

Both machines A and B have a clock rate of 100 MHz. Program P executes 100 million instructions on machine A.

A. What is the CPI (cycle per instruction) on machine A?

Answer: 6x10% + 4x10% + 20x10% + 2x70% = 44

B. How many instructions does program P execute on machine B?

Answer: $0.1*100*10^6$ x 30 + 0.10*100*106 * 20 + $0.10*100*10^6$* 50 + $0.70*100*10^6$ = (300+200+500+70)*10^6=1,070,000,000

C. What is the CPI of program P on machine B?

(Hint: all FP instructions in P are executed using integer instructions)

Answer: 1.0

D. How many millions of floating-point instructions per second are executed of program P on machine A?

Answer:

$$\frac{\#\text{ of FP instructions in P}}{\text{execution time of P}} = \frac{30 \times 10^6}{\text{IC} \times \text{CPI} \times \text{CC}}$$

$$= \frac{30 \times 10^6}{100 \times 10^6 \times 4.4 \times 10\text{ns}} = 6.81 \times 10^6$$

10. Assume that we want to compare the performance of two different computers. Both computers have the same ISA, use the same compiler, and will execute the same program. Computer A is built using the single-cycle implementation. Execution times for each type of instruction for Computer A are given as:

Instruction	Total
beq	5 ns
sw	8 ns
j (jump)	2 ns
lw	9 ns
R-format	7 ns

Computer B is implemented using the multiple-clock-cycle implementation with the cycle time of 2 ns.

Assume the instruction mix for the workload of both computers is 20% beq, 15% sw, 10% jumps, 30% lw, and 25% R-format. Which computer is faster and by how many times?

$CPU\_\_ exec \_\_ time = IC \times CPI \times cycle \_\_ time$

For Computer A:

$CPU\_\_ exec \_\_ time = IC \times 1 \times 1ns = 9\ ICns$

For Computer B:

$$CPI = (0.2)(3) + (0.15)(4) + (0.1)(3) + (0.3)(5) + (0.25)(4)$$
$$= 0.6 + 0.6 + 0.3 + 1.5 + 1.0$$
$$= 4.0$$

$CPU\_\_ exec \_\_ time = IC \times 4.0 \times 2ns = 8\ ICns$

$$\frac{Perf\_B}{Perf\_A} = \frac{CPU\_exec\_time_A}{CPU\_exec\_time_B} = \frac{9ICns}{8ICns} = 1.125$$

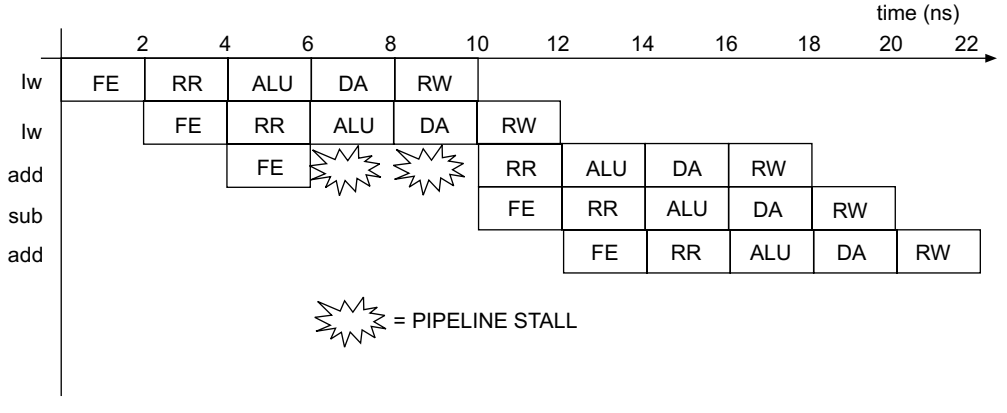
Computer B is 1.125 times faster than Computer A.

11. In a system, a instruction will be broken down into the following five (5) stages: 1) fetch the instruction (FE), 2) read the registers and decode the instruction (RR), 3) use the ALU to perform the specified operation or to compute an address (ALU), 4) access data in memory (DA), and 5) write results into a register (RW). Assume each stage of the pipeline requires the same amount of time to complete, 2 ns, all five steps are used for each instruction, the single-cycle datapath modified for pipelining is used, the register file can be read and written in the same clock cycle, and that all other functional units operate the same as they did for the single-cycle implementation. Draw the multiple-clock-cycle pipeline diagram showing the pipelined execution of the following instruction sequence as it is shown.

Is it possible to improve the execution of the pipeline? **If so,** identify any hazards present in the original sequence, describe any changes made to the code sequence to create the improvements, and draw the resulting multiple-clock-cycle pipeline diagram showing the improved execution. **If not**, indicate that performance cannot be improved . Calculate any speedup realized by your changes compared to the pipelined execution.

```
lw    $s0, 10($t0)
lw    $s1, 100($t0)
add   $s2, $s0, $t1
sub   $s3, $s5, $t1
add   $s4, $s5, $t1
```

Answer: Execution of instructions in the given pipeline is shown below along with probable stalls. It takes 22 ns to complete the execution of all the instructions.

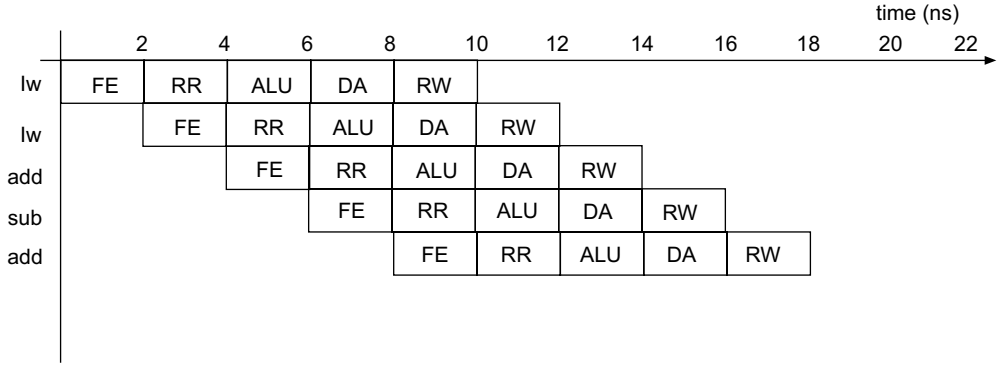


There exists a data hazard with register $s0.

The code sequence can be changed as shown below to improve the execution in the pipeline.

```
lw      $s0, 10($t0)
lw      $s1, 100($t0)
sub     $s3, $s5, $t1
add     $s4, $s5, $t1
add     $s2, $s0, $t1
```

The resulting multiple-clock-cycle pipeline diagram with the modified code sequence is shown below. We may find that this will be taking 18 ns to complete all the instructions.



Thus, the Speedup = 22/18 = 1.22

12. Assume that we have two machines M1 and M2 with the same instruction set architecture. Machine M1 has a clock rate of 500MHz and takes an average of 2.0 clock cycles per instruction (CPI) for a program. Machine M2 has a clock rate of 400MHz and a CPI of 1.2 for a program.

A. Which machine is faster while running this program, and by how much?

B. If a test program executes in 10 million instructions (on both machines), how long will it take to run the program on Machine M1? Machine 2?

C. Suppose that instructions implemented on M1 fall into two performance classes, the first (Class A) takes one clock cycle to execute, while the second (Class B) takes 5 clock cycles to execute. How many Class B instructions are executed when our test program is run?

Answer: Do remember about the following equation to find the time in seconds required to execute a program.

$$\frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

A. Assume I as the number of instructions in the given program. Therefore,

Execution $\text{Time}_{M1} = I \times 2.0 \times 1/(500 \times 10^6) = I/(250 \times 10^6) = 4I \times 10^{-9}$ seconds

Execution $\text{Time}_{M2} = I \times 1.2 \times 1/(400 \times 10^6) = 0.3I/(100 \times 10^6) = 3I \times 10^{-9}$ seconds

Since both M1 and M2 are running the same instructions (same program, same instruction set), we can use I in both equations. Comparing the execution times for M1 and M2, I divides out, and we see that the Execution Time_{M1} / Execution $\text{Time}_{M2} = 4/3$. From this we can conclude that the program runs 33% slower on Machine M1.

B. Same formulas from (a), but now we have number of instructions in the program, $I = 107$. Therefore,

Execution $\text{Time}_{M1} = I \times 2.0 \text{ x } 1/(500 \times 10^6) = 10^7/(250 \times 10^6) = 1/25 = 0.04$ seconds

Execution $\text{Time}_{M2} = I \times 1.2 \times 1/(400 \times 10^6) = 0.3 \times 10^7/(100 \times 10^6) = 3/100 = 0.03$ seconds

C. Let A be the number of Class A instructions, and B the number of Class B instructions. So, $A + B = 10^7$

The execution time for M1 (calculated in clock cycles) is: (1A + 5B)(clock cycle time).

The CPI for the entire program is 2.0. That means that, on average, every instruction executed takes 2 clock cycles to run. In terms of CPI, the total execution time is therefore:

CPI × (clock cycle time) (number of instructions) = 2.0 I (clock cycle time) Thus, $A + 5B = 2.0I = 2 \times 10^7$

$(A + 5B) - (A + B) = 4B = (2 \times 10^7) - 10^7 = 1 \times 10^7$.

Therefore, $4B = 1 \times 10^7$

And $B = 2.5 \times 10^6 = 2{,}5$ million instructions.

13. What does ISA conveys to us?

Answer: The *Instruction Set Architecture* (ISA) provides a view of a processor's features as seen from the perspective of an assembly or machine language programmer. ISA describes the instructions that the processor understands, the way those instructions are presented to the processor, the register set, and the way memory is organised. Of course, a real-world processor's ISA would also include a few additional items, such as its interrupt and/or exception handling facilities, basic data types, and different modes of operation, e.g. supervisor *vs*. normal mode.

14. In stack based machines, where does stack is physically located?. What is stack depth?

Answer: Stack machines are implemented by making the top portion of the stack internal to the processor (i.e., in registers). This is referred to as the stack depth. The rest of the stack is placed in memory. Thus, to access the top values that are within the stack depth, we do not have to access the memory. Obviously, we get better performance by increasing the stack depth. Most scientific calculators also use stack-based operands.

15. Differentiate between Little Endian versus Big Endian machines. Explain how Hex number 12345678 is stored in both the systems?

Little-endian machines:

Machines that store a multiple-byte data with its least significant byte first (at the lower address) and its most significant byte last. This method is used by Intel microprocessors (Most PCs).

Big-endian machines:

Machines that store a multiple-byte data with its most significant byte first (at the lower address) and its least significant byte last. This method is used by Motorola microprocessors and most UNIX machines.

Given Hex Value is Stored in both Big and Little Endian Format as shown here.

Address ⟶	00	01	10	11
Big Endian	12	34	56	78
Little Endian	78	56	34	12

In the following example, table cells represent bytes, and the cell numbers indicate the address of that byte in main memory.

Note: by convention we draw the bytes within a memory word left-to-right for big-endian systems, and right-to-left for little-endian systems.

Word Address	**Big-Endian**				Word Address	**Little-Endian**			
0	0	1	2	3	0	3	2	1	0
4	4	5	6	7	4	7	6	5	4
8	8	9	10	11	8	11	10	9	8
12	12	13	14	15	12	15	14	13	12
	MSB ⟶ LSB					MSB ⟶ LSB			

Note: an N-character ASCII string value is not treated as one large multi-byte value, but rather as N byte values, i.e. the first character of the string always has the lowest address, the last character has the highest address. This is true for both big-endian and little-endian. An N-character Unicode string would be treated as N two-byte value and each two-byte value would require suitable byte-ordering.

The following example show the contents of memory at word address 24 if that word holds the number given by 122E 5F01H in both the big-endian and the little-endian schemes?

	Big Endian					Little Endian			
	MSB ⟶ LSB					MSB ⟶ LSB			
	24	25	26	27		27	26	25	24
Word 24	12	2E	5F	01	Word 24	12	2E	5F	01

The following example show the contents of main memory from word address 24 if those words hold the text RAMANA RAO.

	Big Endian					Little Endian			
	+0	+1	+2	+3		+3	+2	+1	+0
Word 24	R	A	M	A	Word 24	A	M	A	R
Word 28	N	A		R	Word 28	R		A	N
Word 32	A	O	?	?	Word 32	?	?	O	A

The bytes labelled with ? are unknown.

16. Assume We are exploring alternative design of a CPU architecture, and tried to reduce the average clocks per instruction from 3.5 to 2.7 (due to an improvement in the ALU) with decreases clock speed by 25%. Is this worth it?

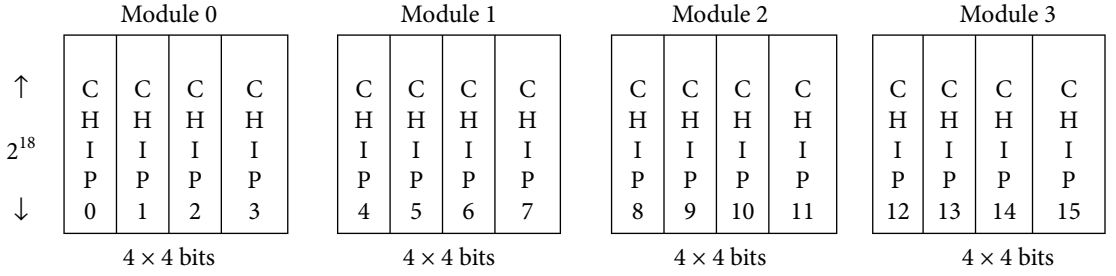
CPUtime = InstructionCount × CPI × ClockCycleTime

Here, InstructionCount is same with both the CPU's. So, we divide CPUtime's of original by modified ones as shown by:

CPUtime(o)/CPUtime(m) = 3.5 × 1.0 / (2.7 × 1.25) = 1.037

So the modified version is almost 4% faster than the original. This, trial is productive trail in the design of the CPU.

17. Design a Main Memory of size 4M x 16 bit (word addressable), using RAM chips of size 256K × 4 bit.



RAM chips per memory module

$$= \frac{\text{Width of Memory Word}}{\text{Width of RAM Chip}} = 16/4 = 4$$

18 bits are required to address a RAM chip (since 256K = 218 = Length of RAM Chip)

A 4Mx16 bit word-addressed memory requires 22 address bits (since $1M = 2^{20}$)

Therefore 4 bits (=22–18) are needed to select a module.

The total number of RAM Chips = (4M × 16) / (256K × 4) = 64.

Total number of Modules = Total number of RAM chips / RamChipsPerModule = 64/4 = 16

18. If the clock rates of machines M1 and M2 are 200 MHz and 300 MHz, find the clock cycles per instruction for program 1 given the following:

Program	**Time on M1**	**Time on M2**
1	10 sec	5 sec
2	3 sec	4 sec

Program	Instructions on M1	Instructions on M2
1	200×10^6	160×10^6

Answer:

CPI= Cycles per second / Instructions per second. So

M1 = 200×10^6 / (200×10^6 / 10 sec) = 10 cycles per instruction

M2 = 160×10^6 / (160×10^6 / 5 sec) = 9.4 cycles per instruction

19. Consider 2 different implementations, M1 and M2 of the same instruction set. There are four classes of instructions (A, B, C and D) in the instruction set. M1 has a clock rate of 500MHz. The average number of cycles for each instruction class on M1 is:

Class	CPI for this class
A	1
B	2
C	3
D	4

M2 has a clock rate of 750 MHz. The average number of cycles for each instruction class on M2 is:

Class	CPI for this class
A	2
B	2
C	4
D	4

Assume that peak performance is defined as the fastest rate that a machine can execute an instruction sequence chosen to maximize that rate. What are the peak performances of M1 and M2 expressed as instructions per second?

Answer: For M1 the peak performance will be achieved with a sequence on instructions of class A, which have a CPI of 1. The peak performance is thus 500 MIPS.

For M2 , a mixture of A and B instructions both of which have a CPI of 2, will achieve peak performance which is 375 MIPS.

20. Consider 2 different implementations, M1 and M2, of the same instruction set. There are three classes of instructions (A, B and C) in the instruction set. M1 has a clock rate of 400 MHz and M2 has a clock rate of 200 MHz. The average number of cycles for each instruction set is given in the following:

Class	CPI M1	CPI M2	C1 Usage	C2 Usage	3rd party
A	4	2	30%	30%	50%
B	6	4	50%	20%	30%
C	8	3	20%	50%	20%

The table also contains information of how the three different compilers use the instruction set. C1 is a compiler produced by the makers of M1, C2 is produced by the makers of M2 and the other is a third party compiler. Assume that each compiler uses the same number of instructions for a given program but that the instruction mix is as described in the table. Using C1 on both M1 and M2 how much faster can the makers of M1 claim that M1 is over M2? Using C2 how much faster is M2 over M1? If we purchase M1 which compiler should we use? For M2?

Answer:

Using C1, the CPI on M1 = 4 * .3 + 6 * .5 + 8 *.2 = 5.8

M2 = 2 * .3 + 4 * .5 + 3* .2 = 3.2

So assuming that M1 is faster we have: (3.2/200E6) / (5.8/400E6) = 1.10 or M1 is 10% faster than M2 using C1

Using C2, the CPI on M1 = 4*.3 + 6*.2 + 8*.5 = 6.4

M2 = 2 * .3 + 4 * .2 + 3* .5 = 2.9

Assuming that M2 is faster we have (6.4/400E6) / (2.9/200E6) = 1.10 or now M2 is 10% faster than M1 using C2.

For the 3rd party compiler:

M1 = 4*.5 + 6*.3 + 8*.2 = 5.4

M2 = 2 * .5 + 4 * .3 + 3* .2 = 2.8

This tells us that the 3rd party compiler is better for both machines since the CPI is lower. If we try M2 as the faster machine we get:

(5.4/400E6) / (2.8/200E6) = .964

So M1 is the faster machine. How much faster? Reverse the equation:

(2.8/200E6) / (5.4/400E6) = 1.037 or M1 is 3.7% faster than M2.

21. Consider a program P, with the following mix of operations:

floating-point multiply	10%
floating-point add	15%
floating-point divide	5%
integer instructions	70%

Machine MFP has floating point hardware and can implement the floating point operations directly. It requires the following number of clock cycles for each instruction class:

floating-point multiply	6
floating-point add	4
floating-point divide	20
integer instructions	2

Machine MNFP has no floating point hardware and so must emulate the floating-point operations using integer instructions. The integer instructions take 2 clock cycles. The number of integer instructions needed to implement each of the floating point operations is:

floating-point multiply	30
floating-point add	20
floating-point divide	50

Both machines have a clock rate of 1000 MHz. Find the native MIPS rating for both machines.

Answer: MIPS = Clock Rate / CPI x10^6

CPI for MFP = .1*6 + .15*4 + 0.05*20 + .7*2 = 3.6

CPI of MNFP = 2.

MIPS for MFP = 1000/CPI = 278

MIPS for MNFP = 1000/CPI = 500

22. If the machine in MFP in the last exercise needs 300 million instructions for a program, how many integer instructions does the machine MNFP requires for the same program?

Answer: This is really just a ratio type problem:

INSTRUCTION	Count on MFP in 10^6	Count on MNFP in 10^6
floating-point multiply	30	900
floating-point add	45	900
floating-point divide	15	750
integer instructions	210	210
TOTAL	300	2760

23. The table below shows the number of floating point operations executed in two different programs and the runtime for those programs on three different machines: (times given in seconds)

Program	FP ops	Computer A	Computer B	Computer C
1	10,000,000	1	10	20
2	100,000,000	1000	100	20

Which machine is faster given total execution time? How much faster is it than the other two?

Answer: Total execution time for Computer A is 1001 seconds, Computer B is 110 seconds and C 40 seconds. So C is the fastest. It is 1001/40 = 25 times faster than A and 110/40 = 2.75 times faster than B.

24. What are Benchmarks? List some popular processor benchmarks.

Programs that are specifically chosen to measure performance of processors are called as benchmarks.

They are used to compare running a program on Computer A *vs*. Computer B. There exists various Types of benchmarks such as: Database access, Math floating point programs, compilers, etc.

Well-known benchmarks:

Whetstone: Synthetic program used for performance testing which emphasizes floating point operations.

Dhrystone: Emphasises integer operations

SPEC: System Performance Evaluation Cooperative:

Non-profit organisation formed to establish standardised benchmarks

SPEC ratio: How fast this computer is relative to a Sun Ultra 5_10 at 300 MHz?

Set of programs to test integer (SPECint92), floating point (SPECfp92), web accesses (SPECweb99), client-server, etc.

Problems with benchmarks:

Often compilers are optimised for the benchmark – not YOUR program.

Characteristics of THEIR programs may be different than YOUR program.

Proper use of benchmarks:

Use programs typical of expected workload.

Use programs typical of expected class of applications: compilers/editors, scientific applications, games.

Using Benchmarks

Weighting each program by its use

Consider the following scenario:

	Time on System A	Time on System B
Program X	1	10
Program Y	1000	100

If the two programs were used 50% of the time we could add the use together to find weighted time:

System A = (.5)(1) + (.5)(1000) = 500.5

System B = (.5)(10) + (.5)(100) = 55

System B appears faster than System A.

If the program is used 90% and Program Y 10% of the time, then the weighted time:

System A = (.9)(1) + (.1)(1000) = 100.9

System B = (.9)(10) + (.1)(100) = 19

System B still appears faster than System A!

When would System A appear faster than SystemB?

1n + 1000(1-n) < 10n + 100(1-n)

1000 + 1n – 1000n < 10n – 100n + 100

900 < 909n

900/909 < n

n > .9900990099

SystemA = .99(1) + .01(1000) = 10.99

SystemB = .99(10) + .01(100) = 10.9

Thus, SystemB is faster always compared to SystemA.

25. A new computer design improves the clock rate from 2 GHz to 2.5 GHz, but has a lower efficiency in the CPI of 1.6 instead of 1.3 due to memory access bottlenecks. The compiler remains the same. Is the computer worth building?

Time before changes= IC*1.3*1/2000 = 0.00065IC

Time after changes= IC*1.6*1/2500= 0.00064IC

As the second one is smaller than first one, the changes can be said as positive benefit on CPU time. So, one can build the computer with the proposed modifications.

26. A given application written in Java runs 15 seconds on a desktop processor. A new Java compiler is released that requires only 60% as many instructions as the old compiler. Unfortunately, it increases CPI by 10%. How fast can we expect the new application to run using this new compiler?.

Time before changes: IC*CPI*cycles/second=15sec

Time after changes:0.6*IC*1.1*CPI*cycles/second

= 0.66*IC*CPI*cycles/second

= 0.66*15=9.9sec

Therefore, speedup=15/9.9=1.515

Example: The following table gives execution times and use distributions for a set of programs. Which system will have the best performance, considering their projected use?

	Execution Time			**Use Distribution**
	System A	**System B**	**System C**	
Compiler	10	15	20	10%
Billing	500	600	400	30%
Editor	30	10	15	30%
Service	800	750	850	30%

System A=0.1*(10) + 0.3(500) + 0.3(30) + 0.3(800) = 400

System B=0.1*(15) + 0.3(600) + 0.3*(10) + 0.3*(750) =409.5

System C = =0.1(20) + 0.3(400) + 0.3(15) + 0.3(850) = 381.5

Thus, System C is better.

27. Explain about datapath and internal bus architectures.

Answer: A typical CPU can be divided into a data section and a control section. The **data section**, which is also called the **datapath**, *contains the registers and*

the ALU. The datapath is capable of performing certain operations on data items. The **control section** is basically the **control unit**, *which issues control signals to the datapath*. Internal to the CPU, data move from one register to another and between ALU and registers. Internal data movements are performed via local buses, which may carry data, instructions, and addresses. Externally, data move from registers to memory and I/O devices, often by means of a system bus. Internal data movement among registers and between the ALU and registers may be carried out using different organisations including one-bus, two-bus, or three-bus organisations. Dedicated datapaths may also be used between components that transfer data between themselves more frequently. For example, the contents of the PC are transferred to the MAR to fetch a next (new) instruction at the beginning of every instruction cycle.

One-Bus Organisation

Using one bus, the CPU registers and the ALU use a single bus to move outgoing and incoming data. Since a bus can handle only a single data movement within one clock cycle, two-operand operations will need two cycles to fetch the operands for the ALU. Additional registers may also be needed to buffer data for the ALU. This bus organisation is though simple and cheap it limits the amount of data transfer that can be done in the same clock cycle, which will kill the overall performance. The following figure shows a one-bus datapath.

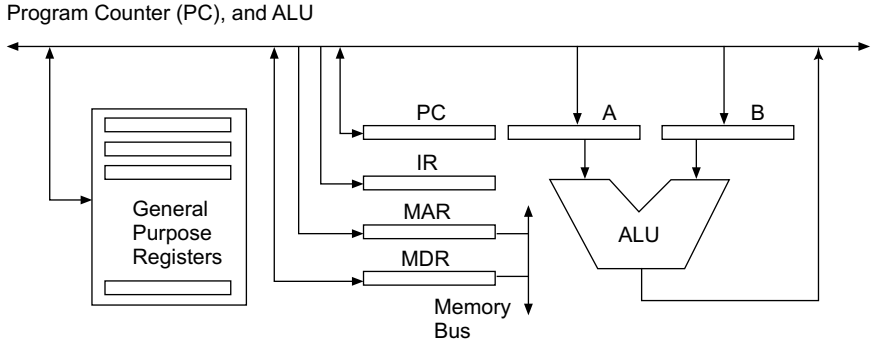


Figure 1.80 One-bus datapath

Two-Bus Organisation

Here, general-purpose registers are connected to two buses. Data can be transferred from two different registers to the input point of the ALU at the same time. Therefore, a two operand operation can fetch both operands in the same clock cycle. An additional buffer register may be needed to hold the output of the ALU when the two buses are busy carrying the two operands. Figure 1.81 a shows a two-bus organisation. In some cases, one of the buses may be dedicated for moving data into registers (in-bus), while the other is dedicated for transferring data out of the registers (out-bus). In this case, the additional buffer register may be used, as one of the ALU inputs, to hold one of the operands. The ALU output can be connected directly to the in-bus, which will transfer the result into one of the registers. Figure b shows a two-bus organisation with in-bus and out-bus.

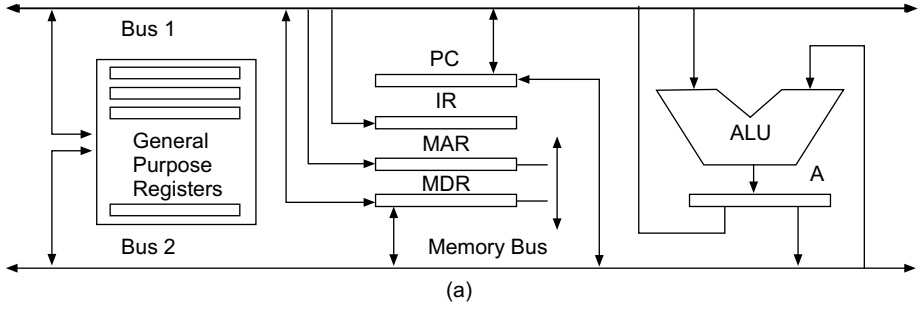


Figure 1.81(a) Two-bus organization

Three-Bus Organisation

Here, two buses are used as source buses while the third is used as destination. The source buses move data out of registers (out-bus), and the destination bus may move data into a register (in-bus). Each of the two out-buses is con-

nected to an ALU input point. The output of the ALU is connected directly to the in-bus. As can be expected, the more buses we have, the more data we can move within a single clock cycle. However, increasing the number of buses will also increase the complexity of the hardware. The following figure shows an example of a three-bus datapath.

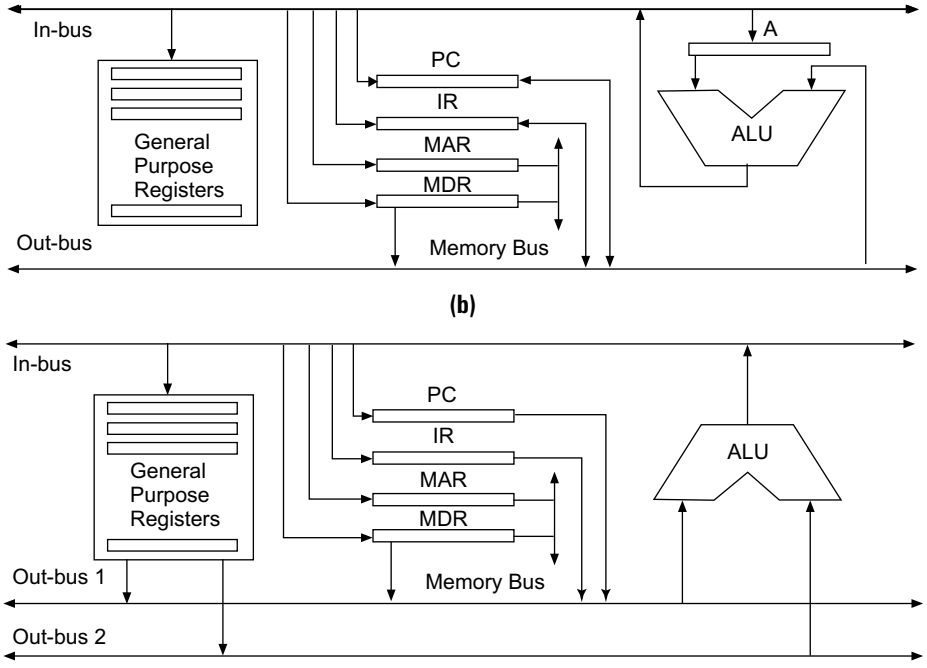


Figure 1.81(b) and (c) Three-bus organizations

28. What are typical elements of a machine instruction?

Answer: Operation code, source operand reference, result operand reference, next instruction reference.

29. What types of location can hold source and destination operands?

Answer: - Main or virtual memory
- Processor register
- I/O device

30. What types of operands are typical in ISA?

Answer: Addresses, numbers, characters, logical data

31. Given the following memory values and a one address instruction with an accumulator, what values do the following instructions load into the accumulator?

Word 20 contains 40
Word 30 contains 50
Word 40 contains 60
Word 50 contains 70

Answer:

20	40
30	50
40	60
50	70

LOAD IMMEDIATE 20 : 20
LOAD DIRECT 20 : 40
LOAD INDIRECT 20 : 60
LOAD IMMEDIATE 30 : 30
LOAD DIRECT 30 : 50
LOAD INDIRECT 30 : 70

32. Lets the address stored in the program counter be designated by the symbol X1. The instruction stored in X1 has an address part (operand reference) X2. The operand needed to execute the instruction is stored in the memory word with address X3. An index register contains the value X4. What is the relationship between these various quantities is the addressing mode of the instruction is

A. direct B. indirect
C. relative D. indexed

Answer:

A. direct X3=X2
B. indirect X3=[X2]
C. relative X3=X2 + X1 +1
D. indexed X3=X2 + X4

33. Consider a 16-bit processor in which the following appears in main memory, starting at location 200:

200	Load to AC	Mode
201	500	
202	Next Instruction	

The first part of the first words indicates that this instruction loads a value into an accumulator. The mode field specifies an addressing mode and if appropriate indicates a source register; assume that when used, the source register is R1, which has the value of 400. There is also a base register that contains the value 100. The value 500 in location 201 may be part of the addressing calculation. Assume that location 399 contains the value 999, location 400 contains the value 1000 and so on. Determine the effective address and the operand to be loaded for the following address modes:

A = contents of an address field in the instruction
R = contents of an address field in the instruction that refers to a register
EA = actual (effective) address of the location containing the referenced operand
(X) = contents of memory location X or register X

Answer:

A = 500; R1 = 400, R_{base}=100

Memory Content:

	OPCODE	OPERAND
ISA	Load to AC	Mode

201	500
202	Next Instruction
203	
399	999
400	1000
401	1001
500	1100
501	1101
502	1102
600	1200
601	1201
602	1202
700	1300
701	1301
702	1302
1000	1600

	OPCODE	OPERAND
ISA	Load to AC	Mode
1001	1601	
1002	1602	
1100	1700	
1101	1701	
1102	1702	

	REGISTER
R1	400
BX	100

Direct
EA = A = 500;
Operand = [500] = 1100
Immediate
EA = 201;
Operand = A = 500;
Indirect
EA = (A) = [500] = 1100;
Operand = [1100] = 1700
Register
EA = R1;
Operand = R1 = 400
Register indirect
EA = (R) = [R1] = 400;
Operand = [400] = 1000
Displacement
EA = A + (R_{base}) = 500 + 100 = 600
Operand = [600] = 1200
Relative
EA = A + (PC) = 500 + 202 = 702
Operand = [702] = 1302

34. Given a simple instruction format as follows:

Opcode (6 bits)	Operand Reference (12 bits)	Operand Reference (12 bits)

A. What is the maximum directly addressable memory capacity (in bytes)?
4096 words

B. How many opcodes are allowed for this ISA?
2^6 = 64 opcodes

C. How many bits are needed for program counter (PC)?
12 bits in PC

D. How many bits are needed for instruction register (IR)?
30 bits in IR

35. What facts go into determining the allocation of bits for instruction formats?

Answer:

Number of addressing modes

Number of operands

Register vs. memory

Number of register sets

Address range

Address granularity

36. Draw the instruction format the following:

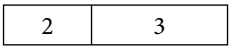
- The machine operates on 16 bit words
- Total opcodes is 32
- 2 types of instructions

Answer:

Format address main (CPU): $2^5=32 \rightarrow 5$ bit

Cache = $2^3 = 8 \rightarrow 3$ bit (line/slot)

Format address main = 5 bit =



MSB (tag) (line/slot) LSB

Memory Map:

Cache Memory

Dec	Address (line/slot)	Tag (2 bit)	Content (word)
7	111	xx	xx
6	110	xx	xx
5	101	xx	xx
4	100	xx	xx
3	011	xx	xx
2	010	xx	xx
1	001	xx	xx
0	000	xx	xx

- Memory Reference Instructions – Opcode 0-29, 1 bit for addressing mode, 1 bit for page, 9 bit for displacement
- Input/Output Instructions – Opcode 30, 128 devices, 16 I/O opcode (I/O command)

Memory Reference Instructions (MRI): Opcodes 5 bit, 1 bit for addressing mode, 1 bit for page, 9 bit for displacement

I/O I: Opcodes 4bit, 7bit for devices, 4bit for I/O opcode (I/O command)

37. Consider a hypothetical system with Main memory having 32-words and Cache having 8-words. The following tables contains content of main memory and cache. The addresses are referred while executing some set of instructions are: 22,26,22,26, 16, 3,16 and 18. Calculate the number of misses.

		Main Memory	
Hex	Dec	Address	Cont (Hex)
1F	31	11111	11
1E	30	11110	10
1D	29	11101	01
1C	28	11100	A1
1B	27	11011	F5
1A	26	11010	F4
19	25	11001	F3
18	24	11000	F1
17	23	10111	FF
16	22	10110	91
15	21	10101	10
14	20	10100	19
13	19	10011	13
12	18	10010	02
11	17	10001	01
10	16	10000	EE
F	15	01111	DD
E	14	01110	CC
D	13	01101	BB

Memory Map:

Hex	Dec	Main Memory Address	Cont (Hex)
C	12	01100	AA
B	11	01011	FF
A	10	01010	C1
9	9	01001	C2
8	8	01000	5F
7	7	00111	13
6	6	00110	23
5	5	00101	B1
4	4	00100	B0
3	3	00011	A3
2	2	00010	A1
1	1	00001	AB
0	0	00000	01

CPU Generate Main Memory:

Generated Address				Format Address		Read Operation Cache			
Address				Tag	Line/Slot	Hit	Miss	Update Cache	Read
(Dec)	(Bin)	Content							
22	10110	91		10	110		√	√	√
26	11010	F4		11	010		√	√	√
22	10110	91		10	110	√			√
26	11010	F4		11	010	√			√
16	10000	EE		10	000		√	√	√
3	00011	A3		00	011		√	√	√
16	10000	EE		10	000	√			√
18	10010	02		10	010		√	√	√

Cache Content (uninitialised)

Dec	Address (line/slot)	Tag (2 bit)	Content (word)
7	111	xx	xx
6	110	xx	xx
5	101	xx	xx
4	100	xx	xx
3	011	xx	xx
2	010	xx	xx
1	001	xx	xx
0	000	xx	xx

BEFORE

Dec	Address (line/slot)	Tag (2 bit)	Content (word)
7	111	xx	xx
6	110	10	91
5	101	xx	xx
4	100	xx	xx
3	011	00	A3
2	010	~~11~~ 10	~~F4~~ 02
1	001	xx	xx
0	000	10	EE

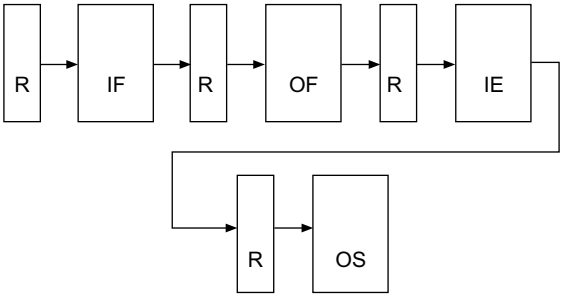
AFTER (UPDATING)

Number of misses = 5.

38. There are 4 segments in a pipeline with the following delays. Assume interface delay as 3ns. Draw space diagram assuming 5 instructions are pumped into pipeline. Calculate speed-up with respect to serial computer.

Segment Name	Segment Execution Time (ns)
Instruction Fetch & Decode (IF)	52
Operand Fetch (OF)	40
Instruction Execute (IE)	30
Operand Store (OS)	40

The simple block diagram of the pipeline is shown here. Do observe that the last stage is not having any interface unit.



The space time diagram for the pipeline (5 instructions):

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T110
IF	I1	I2	I3	I4	I5					
OF		I1	I2	I3	I4	I5				
IE			I1	I2	I3	I4	I5			
OS				I1	I2	I3	I4	I5		

Pipeline cycle time, t_p = Longest segment execution time + interface delay = 52 + 3 = 55ns

Execution time for 5 tasks:

A = Time taken for executing I1 = 4 t_p = 4 * 55

= 220 *ns*

B = The result of the I2, I3, I 4 and I5 will be output at every clock cycle → 4 * 55 = 220 ns

Therefore time to execute 5 instructions = A+B = (220 + 220) ns = 440 ns

Serial execution time of an instruction= (52 + 40 + 30 + 40) ns = 162 ns

Therefore, time needed for 5 instructions = 162 * 5 = 810 ns

Therefore real speedup = Execution in Serial / Execution in Pipeline = 810/440 = 1.84

39. A Computer has clock frequency of 500MHz. Its interrupt handler consumes 400 cycles and a total of 100 cycles are to be spent for data transfer. Assume the data rate as 4 MB/s and 50% of the instructions are observed to be I/O transfers. For driven I/O interrupt, data is transfer in block with size of 16 B. On the other hand, direct memory access (DMA) setup takes 1600 cycles with 16KByte page transfer and one interrupt per page. Calculate the processor overhead during I/O operation for both I/O interrupt and DMA by filling in the table below with the correct value.

Answer:

Parameter	I/O interrupt	DMA
Cycle time	Cycle time = 1/Clock rate =(1/500M) sec	
No of transfer/sec	(4M B/s)/(16 B/transfer) = 2^{22} / 2^4 = 2^{18} transfer/s	(4M B/s)/(16 KB/transfer) = 2^{22} / 2^{14} = 2^8 transfer/s
Total cycle for data transfer	Interrupt cycle + data transfer cycle = 400 + 100 = 500 cycles	Interrupt cycle + data transfer cycle = 400 + 1600 = 2000 cycles.
Total time for data transfer	500 × cycle time = 500 × (1/500M) sec = (1/M) sec =(1×10^{-6})sec	2000 × cycle time = 2000 × (1/500M) sec = (4/M) sec = 4×10^{-6} sec
CPU overhead	[50/100 × 2^{18} transfer/s × (1×10^{-6}) = 0.125 or = 0.125 × 100% = 12.5%	50/100 × 2^8 transfer/s × 4×10^{-6} = 0.00512 or = 0.00512 × 100% = 0.0512%

40. Consider a processor with a main memory and single level cache. Assume 50% of the instructions of a program are of data access type. It takes 3 clock cycles for the data to be sent from the cache to the processor. It requires 24 clock cycles to copy a unit from memory to cache. If the cache hit ratio is 90%,

A. what is the average memory access time?

B. what is the memory stall cycles?

C. with 1000 instructions being executed, what is the number of the wasted cycles?

Answer:

A. AMAT = Hit time + (Miss rate x Miss penalty)

= 3 + (0.1 × 24) = 5.4 cycle

B. Memory stall cycle = Memory access x Miss rate x Miss penalty

= 0.5I × 0.05 × 24 = 0.6I cycle

C. Wasted cycle = 0.6 × 1000 = 600 cycle

1.23 Objective Questions

1. What is the mantissa portion of float number 0.085 when it is stored in 32 bit floating point representation?

A. 3019899 B. 2019899

C. 3019898 D. None

Answer: 0.085 when stored in computer it will be like 00111101101011100001010001111011. The underlined portion represents the mantissa which 3019899.

2. The following are coded representations of the numbers 2 ,4 , 6, 8. Which is not a Gray code

A. 1110,1010,1111,0101

B. 1001,0011,1111,1100

C. 1101,1000,0010,0111

D. None

3. Assume a program consists of 8 pages and a computer has 16 frames of memory. A page consists of 4096 words and memory is word addressable. Currently, page 0 is in frame 2, page 4 is in frame 15, page 6 is in frame 5 and page 7 is in frame 9. No other pages are in memory. Translate the memory addresses below.

A. 111000011110000

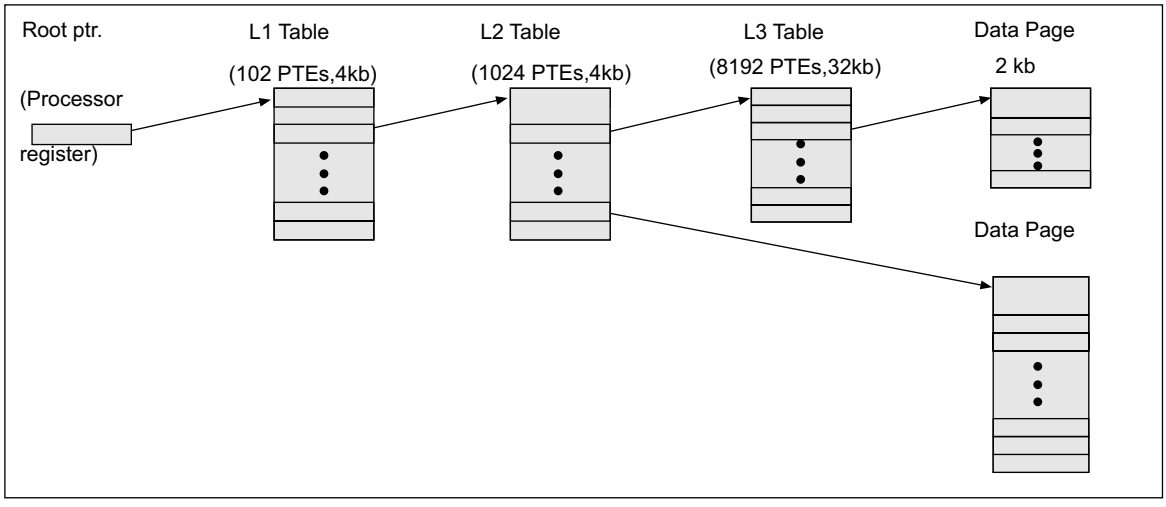
B. 000000000000000

Answers: The first three bits denote the page number (8 pages) and we swap it out for the four bit frame number (16 frames).

A. 111 (page 7) becomes 1001 (frame 9) so 111000011110000 becomes 1001000011110000

B. 000 (page 0) becomes 0010 (frame 2) so 000000000000000 becomes 0010000000000000

4. We have two computer memory management systems with 36-bit physical addresses, 44-bit virtual addresses, and 4-byte PTEs. One system is a single-level page table with 2KB pages and the while the other is a hierarchical page table which supports both 2KB and 16MB pages. The following figure summarizes the hierarchical page table structure and indicates the sizes of the page tables and data pages.



Calculate how much space is needed for each system for the page table for one user process if one 2KB data page is allocated?

2KB page => 11 bit offset

Size of single level page table = No.of pages * PTE= $2^{44}/2^{11}*4= 2^{35}B = 32GB$

Size of hierarchical page table 40KB. Because, we are not allocating whole hierarchy of page tables. We will be allocating just necessary entries in each of the levels as shown in above figure.

Page Table Size = (size of L1) + (size of 1 L2) + (size of 1 L3)
= 4KB + 4KB + 32KB = 40KB

The processor has a fully-associative data TLB with 128 entries, and each entry can map either a 2KB page or a 16MB page. After a TLB miss, a hardware engine walks the page table to reload the TLB. The TLB uses a first-in/first-out (FIFO) replacement policy.

We will evaluate the execution of the following program which adds the elements from three 4MB arrays and stores the results in a fourth 4MB array (note that, 1MB = 1,048,576 Bytes, the starting address of the arrays are given below):

```
double A[524288]; // 4MB array 0x00001800000
double B[524288]; // 4MB array 0x00001c00000
double C[524288]; // 4MB array 0x00002000000
double Y[524288]; // 4MB array 0x00002400000

for(int i=0; i<524288; i++)
  Y[i] = A[i] + B[i] + C[i];
```

Assume that the above program is the only process in the system, and ignore any instruction memory or operating system overheads. The data TLB is initially empty.

How many TLB misses will the program incur with the system that uses the single-level page table?

The program shown above will access 16MB (four arrays each of size 4MB) of consecutive data, and it never reuses the same data.

Number of misses = (total memory)/(page size) = 16MB/2KB = $2^{24-11} = 2^{13}$ = 8K misses

How many TLB misses will the program incur with the hierarchical page table using 2KB pages? What about if it uses 16MB pages?

Misses with 2KB pages: **8k**(Page table doesn't affect TLB miss rate).

Misses with 16MB pages **2**

Although all four arrays could fit in one page, the addresses are not correctly aligned, so the program's data will span two pages.

Example Would a system with only protection be useful (i.e., no translation and no virtual memory)? If so, describe how such a system might be used. If not, explain why.

Yes, protection is useful whenever you want to run more than one application at a time. The other features might not be needed in an embedded system. Protection improves fault tolerance since buggy code "shouldn't" be able to damage other code. Even if only one application is running, it might be made of multiple pieces of code that have bugs, and thus need protection from itself.

Example Would a system with only protection and translation (i.e., no virtual memory) be useful? If so, describe how such a system might be used. If not, explain why.

Yes, it just isn't able to swap out pages to offer the illusion of larger memory capacity. This might be useful with a supercomputer where your job should all fit in memory for peak performance.

Example Would a system with only protection and virtual memory (i.e., no translation) be useful? If so, describe how such a system might be used? If not, explain why.

Yes, but it would be very slow.

Example Mark whether the following modifications will cause each of the categories to increase, decrease, or whether the modification will have no effect. Assume the baseline system uses a two-level page table with a single page size.

	Page Table Size	TLB Hit Rate	Internal Fragmentation
Increase page size	Decrease Larger pages means less pages for the same virtual address space. Less pages means less PTEs which should make a smaller page table.	Increase (probably) The TLB's reach has been extended since each entry maps more memory. For this to improve hit rate there needs to be some spatial locality (there should be).	Increase With larger pages there is a greater chance that each page will not be completely filled.
Making the page table 3 level (page size and virtual address size constant)	Decrease if program has sparsely populated address space. Increase if program has more densely populated address space.	No effect Page size is the same so the same TLB entries will get the same amount of hits and misses.	No effect Page size is the same, so there should not be a change in fragmentation.
Adding support for multiple page sizes	Decreases if the new page size is larger since we will need less PTEs. Could increase if we support a smaller page size and need more PTEs.	Increases with larger pages for same reason as above.	Should decrease as the smallest page to hold data is used to reduce fragmentation. Could increase if larger pages are used to increase TLB hit rate.

OBJECTIVE TYPE QUESTIONS

1. Possible number of digits which are used to represent any number in base-r system

A. r B. r–1

C. $\log_2 r$ D. 2

2. Assuming a flip-flop can take 4 unique stages or level, then the range of positive numbers which we can store in n such a flip-flop's.

A. 0 to 2^n-1 B. $-(2^n-1)$ to (2^n-1)

C. 0 to 4^n-1 D. None

3. If Hamming code is 1010101 then the data is

A. 1010 B. 1101

C. 1110 D. 0111

E. 1110

4. If Hamming code received is 0010111 then which bit position has encountered error?

A. 1 B. 6

C. 3 D. 7

E. 5

5. The number system whose digits weights sum is itself, is called

A. Binary B. Decimal

C. Excess-3 D. None

6. If x is a number represented in n-bit binary system, then –x in 1s complement becomes

A. Just inverse MSB B. 2^n-x-1

C. Complement LSB D. None

7. An 8-bit number x is 00001100, then –x in 1s complement is

A. 12s binary

B. 243 binary code in 8 bits

C. 111011100

D. None

8. If x is a number represented in n-bit binary system, then –x in 2s complement becomes

A. Just inverst MSB B. 2^n-x

C. Complement LSB D. None

9. An 8-bit number x is 00001100, then –x in 1s complement is

A. 12s binary

B. 244 binary code in 8 bits

C. 111011100

D. None

10. If we apply two times 2s complement operation on a 8-bit number N, we get

A. 0 B. 2^8

C. N D. 2^8-N

11. When overflow is occurred, then __ of carry in and carry out of MSB is one.

A. OR B. NXOR

C. XOR D. NOR

12. Overflow in 2s complement addition occurs when

A. Both numbers are largest possible positive numbers

B. Both numbers are smallest possible negative numbers

C. Both A and B are possible

D. One number is positive and another number is negative

13. In a n-bit 2s complement addition, overflow occurs if the result N is

A. Greater than $2^{n-1}-1$ B. Less than -2^{n-1}

C. Both D. None

14. Overflow in 2s complement subtraction occurs when

A. Both numbers are largest possible positive numbers

B. Both the numbers are smallest possible negative numbers

C. Both a and b are possible

D. One number is largest positive and another number is largest negative

15. Assuming 5-bit addition or registers, the sum 12 + 7 leads

A. 19 B. –13

C. Overflow D. None

16. If B and C largest possible positive numbers then over flow may occur

A. B+C B. B–(–C)

C. –B–C D. All

17. Error detection and correction abilities of a code are determined by

A. Code length

B. Number code vectors

C. Minimum distance d_{min}

D. Spacing between parity bits of a code word

18. If the code words are m-bits length and minimum distance of the code is n, then number errors needed to convert a valid code word as another valid code word is

A. m B. n

C. m–n D. n–m

19. To detect single bit errors minimum distance needed for the n-bit code is

A. 3 B. 4

C. n D. 2

20. To correct single bit errors minimum distance needed for the n-bit code is

A. 3 B. 4

C. n D. 2

21. If the minimum distance of a code is 5 then it can

A. Correct single error

B. Detect double bit error

C. Detect three bit errors

D. All

22. Find odd man out of the following

A. 01111011 B. 11111011

C. 10000100 D. 10000101

E. 10010011

23. How do we know that a 2's complement code represents a negative number?

A. The least significant bit is 1.

B. The least significant bit is 0.

C. The most significant bit is 1.

D. The most significant bit is 0.

24. If the following Gray code sequence, what is the missing code? 11010?10000

A. 11011 B. 11110

C. 10000 D. 11000

E. 10001

25. What is the base x for this addition operation to hold: **(223)x + (46)x = (302)x?**

A. 5 B. 6

C. 7 D. 8

E. 9

26. Given the following code: { 10001010, 01011011, 10110001, 01110010, 11101110 } . What is the Hamming distance of this code?

A. 2 B. 3

C. 4 D. 5

E. 6

27. A certain value is represented in the 4-bit sign-and-magnitude, 2s complement, 1s complement and excess-8 schemes as shown below. Which of the following is NOT a representation of this value?

A. $(1011)^{sm}$ B. $(1100)_{2s}$

C. $(1100)_{1s}$ D. $(0101)_{excess\text{-}8}$

E. None of the above

28. Convert the 5-bit binary number $(11101)_2$ into its standard Gray code equivalent.

A. $(11101)_{gray}$ B. $(00010)_{gray}$

C. $(10110)_{gray}$ D. $(10011)_{gray}$

E. $(11110)_{gray}$

29. In which of the following coding schemes that 0111 represents the largest value?

A. The standard 4-bit Gray code

B. The BCD code

C. The Excess-3 code

D. The 84-2-1 code

E. The 2821 code

30. The tribesmen could only count up to 1000 (thousand) as they believed that there was no number beyond that value. Using their number system, how many digits would they need to represent the value 1000?

A. 5 B. 7

C. 10 D. 14

E. 10,000

31. Which of the following addition operations in 5-bit 2's complement number system results in an overflow?

A. $(00100)_{2s} + (01011)_{2s}$

B. $(11011)_{2s} + (01111)_{2s}$

C. $(11101)_{2s} + (10110)_{2s}$

D. $(10010)_{2s} + (11001)_{2s}$

E. $(00000)_{2s} + (11111)_{2s}$

32. Which is true about parity bits?

A. In an odd-parity scheme, a '1' must always be appended to the data.

B. It is possible to correct error with the parity scheme.

C. It is possible to detect a 2-bit switch with the parity scheme.

D. Using parity scheme allows for faster transmission of data.

E. None of the above.

33. The decimal number 2875 is coded as follows in some coding approach. Which of the coding schemes could be self-complementing?

A. 0010 1000 0111 0101

B. 0100 1100 1001 0111

C. 1000 1010 0111 0101

D. 1111 0001 0000 1010

E. None of the above

34. In Hamming codes, parity bits in parity bit stuffed frames are

A. MSB bit B. 0^{th} bit

C. 1^{st} bit D. 3^{rd} bit

35. In the following parity bit stuffed frames which are received by a receiver, the one which contains error

A. 001110010100 B. 101110010100

C. 000110010100 D. 001010000100

36. If the received parity bit stuffed frame contains 101110010100, then error is at

A. 4 B. 5

C. 3 D. 12

37. If the received parity bit stuffed frame is 001100010100, then

A. Parity bits 2,4 are not matching

B. Parity bits 1,3 are not matching

C. Parity bits 1,2 are not matching

D. Parity bits 1,5 are not matching

38. Assuming even parity, seven bit parity code for data 0111 is

A. 0101110 B. 0111110

C. 0001111 D. 1100111

39. The number system in which two representations of zero are available?

A. Sign-magnitude B. Floating point

C. 2s complement D. None

40. In 2s complement, if two numbers with the same sign is added then over flow is said to be occurred if

A. If result is also same sign

B. If result has opposite sign

C. If result is zero

D. None

41. Find odd-one out in terms of 2s complement representation out of the following.

A. 101 B. 1101

C. 11101 D. 11001

42. The following truth table represents

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

A. XNOR B. NAND

C. AND D. None

43. Which of the following is NOT a recognised type of GATE?

A. AND

B. OR

C. NOT

D. EXCLUSIVE OR (XOR)

E. LATCH

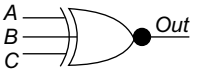
44. The symbol in the figure below is

A. A NOT gate B. An NAND gate

C. A NOR gate D. An XOR gate

E. An XNOR gate

F.



45. Find odd one out

A. AND B. NAND

C. NOT D. XOR

46. In the following, digital one is

A. Telephone B. Cassette Player

C. DVD Player D. None

47. To control real world things through computer, we need

A. Sampling B. Digitization

C. Both D. None

48. Real world things are

A. Digital B. Analog

C. Both D. None

49. Find incorrect one

A. Binary value 1 indicates a voltage of 1V

B. Binary value 0 indicates a voltage of 0V

C. Binary value 0 indicates a voltage of -1V

D. None

50. System which is better immune to noise

A. Digital B. Discrete

C. Analog D. Continuous

51. Find incorrect one regarding Boolean variables

A. AA=A B. A'A'=A'

C. A'A'=A D. AA'=0

52. Device which takes care of sudden power requirements in a circuit

A. SMPS B. capacitor

C. resistor D. UPS

53. Find odd one out

A. An exact voltage of 5V indicates Boolean 1 in TTL

B. An exact voltage of 0V indicates Boolean 0 in TTL

C. A small range around 5V indicates Boolean 1 in TTL

D. None

54. The technology which has better noise susceptibility
A. TTL B. CMOS
C. Both D. None

55. Single T-state Machine
A. RISC B. 386
C. Z80 D. Z180

56. Universal gate
A. NOT B. AND
C. NAND D. XOR

57. The dual of A + 1 = 1?
(Note: * = AND, + = OR and ‘ = NOT)
A. A * 1 = 1 B. A * 0 = 0
C. A + 0 = 0 D. A * A = A
E. A * 1 = 1

58. A literal
A. A Boolean variable
B. The complement of a Boolean variable
C. 1 or 2
D. A Boolean variable interpreted literally
E. The actual understanding of a Boolean variable

59. (A+B+C)(D+E)’ + (A+B+C)(D+E) = .
A. A + B + C B. D + E
C. A’B’C’ D. D’E’
E. None of the above

60. Dual of the Boolean property x + x′y = x + y?
A. x′(x + y′) = x′y′ B. x(x′y) = xy
C. x*x′ + y = xy D. x′(xy′) = x′y′
E. x(x′ + y) = xy

61. Given the function F(X,Y,Z) = XZ + Z(X′+ XY), the equivalent most simplified Boolean representation for F is
A. Z + YZ B. Z+XYZ
C. XZ D. X + YZ
E. None of the above

62. Which of the following Boolean functions is algebraically complete?
A. F = xy B. F = x + y
C. F = x’ D. F = xy + yz
E. F = x + y’

63. Equivalent to Boolean expression (A + B)′(C + D + E)′ + (A + B)′
A. A + B B. A′B′
C. C + D + E D. C′D′E′
E. A′B′C′D′E′

64. Given that F = A′B′+ C′+ D′+ E′, the F′ can also represented as
A. F’= A+B+C+D+E B. F′= ABCDE
C. F’= AB(C+D+E) D. F′= AB+C′+D′+E′
E. F’= (A+B)CDE

65. The Boolean expression A′ + 1 is same as
A. A B. A′
C. 1 D. 0

66. Simplification of the Boolean expression AB + ABC + ABCD + ABCDE + ABCDEF is
A. ABCDEF
B. AB
C. AB + CD + EF
D. A + B + C + D + E + F
E. A + B(C+D(E+F))

67. NOT can be realized using
A. OR B. AND
C. NAND D. XOR

68. No combination of ___ gates can be used to get NOT.
A. AND B. NAND
C. NOR D. None

69. A two input XOR can be realized using
A. Two two input AND
B. Two two input AND, Two NOT and one two input OR gates
C. Two two input NAND, Two XOT and one two input OR gates
D. Two two input NOR, Two NOT and one two input NOR gates

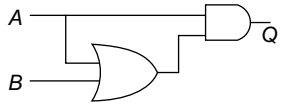
70. Select the Boolean expression that is not equivalent to x·x+x·y
A. x · (x + y) B. (x + y) · x
C. y D. x

71. Select the expression which is equivalent to x · y + x · y · z
A. x · y B. x · z
C. y · z D. x · y · z

72. Select the expression which is equivalent to (x + y) · (x + y)
A. y B. y0
C. x D. None

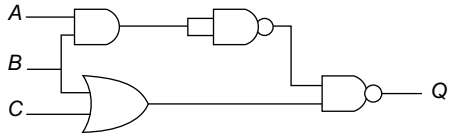
73. Select the expression that is not equivalent to x · (x + y) + y
A. x · x + y · (1 + x) B. x · y + y
C. x · y D. y

74. Equivalent Boolean equation for the following circuit



A. Q=A' B. Q=A
C. Q=B D. Q=0

75. What is the simplified equation of Q?



A. Q=A B. Q=0
C. Q=1 D. None

76. 1+1=10 is valid in
A. Decimal B. Binary
C. Hex D. OR

77. 1+1=1
A. Decimal system B. Hexadecimal system
C. Binary system D. Result of OR gate

78. In TTL logic, 1 indicates
A. Vcc B. Ground
C. 5V D. None

79. If 00 is input, output is zero in
A. AND, OR, NAND B OR, AND, XNOR
C. OR, AND, XOR D. AND, NAND, NXOR

80. If input is 11, then output is 1 in
A. AND, OR, NAND B. OR, AND, XNOR
C. OR, AND, XOR D. AND, OR, NXOR

81. The smallest logical unit of a digital system
A. Transistor B. Gate
C. N, P junctions D. None

82. Logic gates are not constructed from
A. Relays B. Transistors
C. Vacuum tubes D. Transformers

83. Which of the following boolean formulas can be used to implement a parity checker for a 4 bit code stored in binary variables a3, a2, a1, a0? If odd number of 1s are there in the given four bits, we should get parity value as 1 otherwise 0.
A. a3+a2+a1+a0
B. a3 · a2 · a1 · a0
C. a3^a2^a1^a0 (where ^ represents the exclusive OR operation)
D. a0+a1+a2+a3

84. AB ⊕ (A+B) =
A. 1 B. AB
C. A+B D. AB+A'B'
E. A'B+AB'

85. abc′ + bc + a′bc′ =
A. abc B. b
C. b ⊕ c D. None

86. ((X ⊕ Y) ⊕ (X ⊕ Y)) =
A. 0 B. 1
C. XY D. X'Y
E. X'Y+XY'

87. If G, K are Boolean variables, G.K + G'.K' stands for
A. XOR B. XNOR
C. NAND D. NOR
E. NOT

88. If K is a Boolean variable then K+K+ …+K gives
A. 1 B. 0
C. K D. K'

89. If K is a Boolean variable then K.K. ….K gives
A. 1 B. 0
C. K D. K'

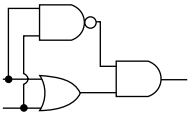
90. If K is a Boolean variable then K+K'+K+K'+K+K' gives
A. 1 B. 0
C. K D. K'

91. If K is a Boolean variable then K.K'.K.K'.K.K' gives
A. 1 B. 0
C. K D. K'

92. Find the correct one
A. ((A·B')' + B')·B =AB
B. A+((A·B')'·C) = BC
C. A + ((B+C)'·A) =A
D. A + ((B+C)'·A) =A+B+C

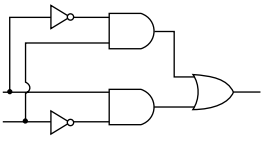
93. To change OR gate to AND gate, we have to
A. Invert each input B. Invert output
C. Do both a and b. D. None

94. The following circuit is



A. XNOR B. XOR
C. NOR D. NAND

95. The following circuit is equivalent to



A. XNOR B. XOR
C. NOR D. NAND

96. It was developed by George Boole, and is often used to refine the determination of system status or to set or clear specific bits.

A. On B. NXOR
C. Boolean Logic D. AND

97. Boolean Logic uses ____ to determine how bits are compared and simulates ____ .
A. Operators, Gates B. Buffers, AND
C. Truth Tables, Off D. NOR, NAND

98. An operation says if and only if all inputs are on, the output will be on. The output will be off if any of the inputs are off. Compliment of this operation is
A. OR B. NAND
C. NOR D. AND

99. The _____ operation says if any input is on then the output will be off.
A. NOT B. OR
C. NOR D. XOR

100. If and only if all of the inputs are on, the output will be off. This is called _____ .
A. NAND B. NOR
C. Truth Tables D. On

101. This operation says that if any input is on, the output will be off. What operation is this?
A. Boolean Logic B. XOR
C. NOR D. AND

102. _____ says that if the inputs are different then the output will be on.
A. Gates B. Low
C. NXOR D. XOR

103. _____ simply changes the input to the opposite (0 to 1 or 1 to 0).
A. Operator
B. A 2-input NOT gate with both of inputs being same.
C. A 2-input NAND gate with both of inputs being same.
D. A 2-input OR gate with both of inputs being same.

104. F = A′ + B′ is same as
A. AND between A and B
B. NAND between A′ and B′
C. NAND between A and B
D. NOR between A and B

105. F = A′ B′ is same as
A. AND between A and B
B. NAND between A′ and B′
C. NAND between A and B
D. NOR between A and B

106. A′ + B′ =
A. AB′ B. A+B′
C. (AB)′ D. None

107. A three level circuit
A. AB+BC B. A(B+C)+BC′
C. (A+B)(A+C) D. None

108. Three level equivalent representation of circuit BA+D+AC
A. AB + AC + D B. A(B + C) + D
C. (A + B)(A + C)D D. None

109. F outputs 1 when a is 0 and b is 0, or when a is 0 and b is 1
A. F(a,b)=a B. F(a,b)=a'b'+a'b
C. F(a,b)=ab'+a'b D. None

110. Find the odd man out
A. F(a,b)=a' B. F(a,b)=a'b'+a'b
C. F(a,b)=(a'+b)(a'+b') D. F(a,b)=a

111. Find incorrect Boolean equation for the statement "F outputs 1 when a is 0, regardless of b's value".
A. F(a,b)=a' B. F(a,b)=a'b'+a'b
C. F(a,b)=(a'+b)(a'+b') D. F(a,b)=a

112. The basic building block for a logical circuit is _____
A. A Flip-Flop B. A Logic Gate
C. An Adder D. None

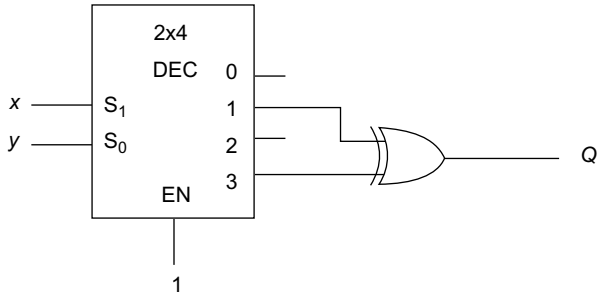
113. Digital quantity
A. Volume of a loud speaker
B. Output of a microphone
C. Timer in a microwave Owen
D. None

114. Next number to $(477)_8$
A. 478 B. $(478)_8$
C. $(500)_8$ D. None

115. Number before $(600)_8$
A. 478 B. $(478)_8$
C. $(577)_8$ D. None

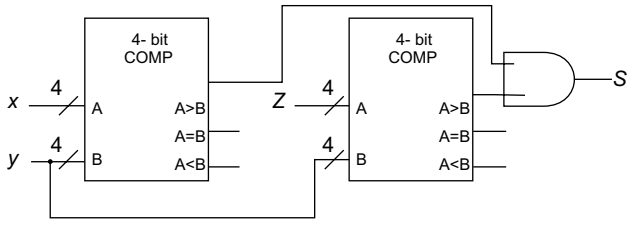
116. Equivalent universal gate for F=A'+B'
A. NOT B. NOR
C. NAND D. AND

117. A Boolean function $Q(x, y)$ is implemented using a 2×4 decoder as shown below. What is Q?



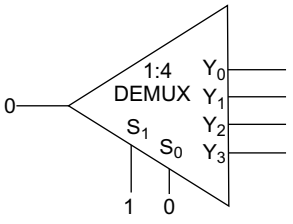
A. $Q(x, y) = x$ B. $Q(x, y) = x'$
C. $Q(x, y) = y$ D. $Q(x, y) = y'$
E. $Q(x, y) = x'y$

118. The circuit below compares three unsigned 4-bit binary numbers X, Y, and Z using the 4-bit magnitude comparators. Which of the following values of X, Y, and Z will result in S being 1?



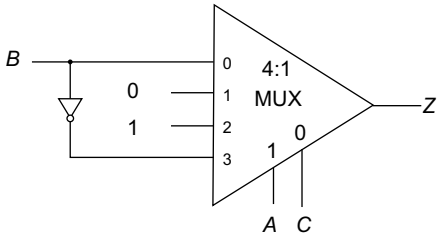
A. X = 0000, Y = 1000, Z = 1111
B. X = 1111, Y = 1000, Z = 0000
C. X = 1111, Y = 0000, Z = 1000
D. X = 1000, Y = 1111, Z = 0000
E. None of the above

119. The circuit below shows a 1:4 demultiplexer with input 0 and selection lines 10. What are the outputs?



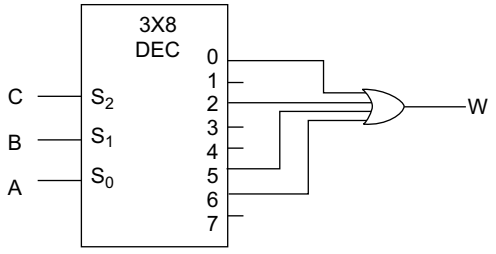
A. Y0 = 0; Y1 = 0; Y2 = 0; Y3 = 0
B. Y0 = 1; Y1 = 0; Y2 = 0; Y3 = 0
C. Y0 = 0; Y1 = 1; Y2 = 0; Y3 = 0
D. Y0 = 0; Y1 = 0; Y2 = 1; Y3 = 0
E. Y0 = 0; Y1 = 0; Y2 = 0; Y3 = 1

120. A Boolean function Z(A,B,C) is implemented using a 4:1 multiplexer as shown below. What is Z?



A. Z(A,B,C) = Σ m(1, 4, 5, 6)
B. Z(A,B,C) = Σ m(2, 3, 4, 6)
C. Z(A,B,C) = Σ m(0, 2, 5, 6)
D. Z(A,B,C) = Σ m(2, 4, 6, 7)
E. Z(A,B,C) = Σ m(2, 4, 5, 6)

121. Tom implemented the Boolean function W(A,B,C) = Σ m(0, 2, 5, 6) using a decoder as shown below. Instead of connecting A, B, C to S2, S1, S0 respectively, he has mistakenly connected C, B, A to S2, S1, S0, respectively. When will his mistake be exposed?



A. When the inputs ABC = 000
B. When the inputs ABC = 010
C. When the inputs ABC = 101
D. When the inputs ABC = 110
E. When the inputs ABC = 111

122. An **8-to-3 encoder** accepts only certain valid inputs. Suppose a Boolean function ***V(A,B,C,D,E,F,G,H)*** generates 1 if the values of *A,B,C,D,E,F,G,H* are valid inputs for the encoder, or 0 otherwise, what is *V(A,B,C,D,E,F,G,H)* in Σ*m* notation?

A. Σ*m*(0, 1, 2, 3, 4, 5, 6, 7)
B. Σ*m*(1, 2, 3, 4, 5, 6, 7, 8)
C. Σ*m*(0, 1, 2, 4, 8, 16, 32, 64)
D. Σ*m*(1, 2, 4, 8, 16, 32, 64, 128)
E. Σ*m*(1, 3, 7, 15, 31, 63, 127, 255)

123. Given an excess 3 code *ABCD*, a Boolean function *F(A,B,C,D)* generates 1 if the actual digit represented by *ABCD* is divisible by 5, or 0 otherwise. Employing don't-care outputs, what is the simplified SOP expression for F?

A. $A' \cdot B' + B' \cdot C' \cdot D'$
B. $B' \cdot (A' + C' \cdot D')$
C. $A' \cdot B' + A \cdot B' \cdot C' \cdot D'$
D. $A' \cdot B' \cdot C \cdot D + A \cdot B' \cdot C' \cdot D'$
E. B'

124. Which of the following combinations can form a 3×8 decoder?

A. Two 2×4 decoders and an inverter.
B. Two 2×4 decoders and one 1×2 decoder.
C. Four 1×2 decoders and one 2×4 decoder.
D. All of the above.
E. None of the above.

125. The circuit which converts 2^n lines to n lines, where n is positive integer

A. Decoder B. Encoder

C. Adder D. Subtractor

126. The circuit which converts n lines to 2^n lines, where n is positive integer

A. Decoder B. Encoder

C. Adder D. Subtractor

127. The circuit which always gives one in only one of its output lines

A. Decoder B. Encoder

C. Adder D. Subtractor

128. What will be the value of all output lines of a decoder if enable is not selected?

A. 1 B. 0

C. X D. None

129. The circuit which always expects one in only one of its input lines is

A. Decoder B. Encoder

C. Adder D. Subtractor

130. What will be the value of all output lines of a encoder if all of its inputs or 0s

A. 1 B. 0

C. X D. None

131. How many full adders are used in bit-serial adder?

A. 2 B. 3

C. 1 D. n

132. The binary numbers A = 1100 and B = 1001 are applied to the inputs of a comparator. What are the output levels?

A. A > B = 1, A < B = 0, A < B = 1

B. A > B = 0, A < B = 1, A = B = 0

C. A > B = 1, A < B = 0, A = B = 0

D. A > B = 0, A < B = 1, A = B = 1

133. A particular Full Adder has

A. 3 inputs and 2 outputs

B. 3 inputs and 3 outputs

C. 2 inputs and 3 outputs

D. 2 inputs and 2 outputs

134. For a 3-to-8 decoder how many 2-to-4 decoders will be required?

A. 2 B. 1

C. 3 D. 4

135. For a 4-to-8 decoder how many 2-to-4 decoders will be required?

A. 2 B. 1

C. 5 D. 4

136. The four outputs of two 4-input multiplexers, connected to form a 16-input multiplexer, are connected together through a 4-input _____ gate.

A. AND B. OR

C. NAND D. XOR

137. BCD to 7-Segment decoder has

A. 3 inputs and 7 outputs

B. 4 inputs and 7 outputs

C. 7 inputs and 3 outputs

D. 7 inputs and 4 outputs

138. Demultiplexer has

A. Single input and single outputs.

B. Multiple inputs and multiple outputs.

C. Single input and multiple outputs.

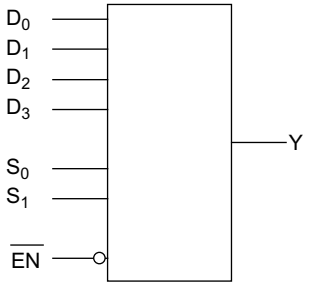
D. Multiple inputs and single output.

139. Single input multiple output

A. MUX B. Decoder

C. DEMUX D. None

140. The following device is most likely a



A. Comparator B. Multiplexer

C. Demultiplexer D. Parity generator

141. Demultiplexer converts _____ data to _____ data

A. Parallel data, serial data

B. Serial data, parallel data

C. Encoded data, decoded data

D. All of the given options.

142. Multiplexer converts _____ data to _____ data

A. Parallel data, serial data

B. Serial data, parallel data

C. Encoded data, decoded data

D. All of the given options.

143. In an n variable system, the term which takes maximum number of literals contains __ number of literals.

A. 1 B. n

C. $\log_2(n)$ D. n/2

144. In an n variable Karnaugh map, the implicant term which contains half of the cells of map as adjacent 1s will be giving a product term with ___ number of literals.

A. 1 B. n

C. log2(n/2) D. n/2

145. In an n variable Karnaugh map, the group which contains ___ adjacent 1s will be giving a product term with $\log2(2^n/2)$ number of literals.

A. 1 B. n

C. $2^n/2$ D. n/2

146. When can we say that no input variable has effect on output variable in an n input variable digital circuit?.

A. If all cells in Karnaugh map are having 0s

B. If all cells in Karnaugh map are having 1s

C. Both a and b.

D. We cannot conclude that simply.

147. Each cell in a Karnaugh map is ___

A. Logic gate B. Logic unit

C. Minterm or maxterm D. A Boolean theorem

148. Each cell in a Karnaugh map holds __

A. An integer B. 0

C. 1 D. b&c

149. Each cell in a Karnaugh map represents ___

A. An AND gate B. Product

C. Sum D. Canonical term

150. Valid grouping of Karnaugh maps is

A. 3 cells

B. Two cells in diagonal

C. Consecutive cells which are of the form 2^1, 2^2, 2^3 etc.,

D. None

151. The variables (literals) are organised in Karnaugh map in ___ such that they are in logically and graphically adjacent.

A. Binary code B. Grey code

C. ASCII code D. UNICODE

152. Karnaugh map can be used

A. To detect racing condition

B. To calculate delay in the circuit

C. To simplify the circuit complexity

D. None

153. Karnaugh map the values are ordered in a ___, such that precisely one variable changes between any pair of cells.

A. Gray code B. Hypercube

C. Mesh D. Algorithm

154. F(a,b,c)=a'+b+c. What should be the nature of K-Map.

A.

1	1	1	
1	1	1	1

B.

1	1		1
1	1	1	1

C.

1		1	1
1	1	1	1

D.

1	1	1	1
1	1	1	

155. Canonical forms

A. Minterms B. Maxterms

C. Both D. Prime implicants

156. Minterm

A. Is product term

B. Contains n literals in an n-variable system

C. Is a canonical form

D. All

157. Canonical forms are usually not minimal. That is, they contains all the literals in primed or non-primed form (Y/N).

158. In a three variable system, if f = a′bc + ab′c′ + ab′c +abc′ +abc then f′ = a′b′c′ + a′b′c + a′bc′ (Y/N).

159. In a three variable system, if f = (a + b + c)(a + b + c') (a + b′ + c) then f′ = (a + b′ + c′)(a′ + b + c)(a′ + b + c′) (a′ + b′ + c)(a + b + c′) (Y/N).

160. Product terms are

A. Minterms B. Maxterms

C. Both D. None

161. In a four variable system with Boolean literal A,B,C,D essential prime implicants out of the prime implicants A′B′D, BC′, AC,A′C′D,AB, B′CD

A. AB B. AC

C. ABD D. B′CD

162. In a four variable system, minimum cover if the prime implicants are A′B′D, BC′, AC,A′C′D,AB, B′CD

A. AC+BC′+AB B. AC+BC+AB

C. AC+BC′+A′B′D D. None

163. In a four variable system, the essential prime implicant if the prime implicants are BD, ABC′, ACD, A′BC, A′C′D

A. ACB B. ABC′

C. BD D. AB′C′

164. In a four variable system, minimum cover if the prime implicants are BD, ABC′, ACD, A′BC, A′C′D

A. BD+AB

B. ABC′+ACD+A′BC

C. ABC′+ACD+A′BC+A′C′D

D. None

165. In Karnaugh map, two adjacent Maxterms differ in ___ variables

A. 2 B. 1

C. n D. n

166. In a Karnaugh map with n variables, two adjacent maxterms differ in _____ variables.

A. 1 B. 2

C. 3 D. 4

167. In a Karnaugh map with m variables, maximum possible grouping contains

A. m cells B. m^{-1} cells

C. m^2 cells D. 2^m cells

168. In a Karnaugh map with m variables, a grouping of n cells (where n is less than or equal to m and n is some integer power of 2) gives ___ number of variables in product term.

A. log2(n) B. m-log2(n)

C. 1 D. None

169. In a Karnaugh map with m variables, a grouping of n cells (where n is less than or equal to m and n is some integer power of 2) gives a prime implicant with __ variables.

A. log2(n) B. m-log2(n)

C. 1 D. None

170. In a Karnaugh map with m variables, a grouping of 2^n cells (where n is an integer) gives ___ number of variables in product term.

A. log2(n) B. m-n

C. 1 D. None

171. If we add two 1s which are in diagonal style in Karnaugh map, we get an ___ term.

A. AND B. Product

C. OR D. XOR

172. In four variable Karnaugh map, two ends of the principal diagonal cells differs in __ literals.

A. 1 B. 2

C. 4 D. 3

173. In four variable Karnaugh map, two ends of the principal diagonal cells differs in __ literals.

A. 1 B. 2

C. 4 D. 3

174. In four variable Karnaugh map, the minterm whose binary code 1111 is located in

A. 2^{nd} row

B. 3^{rd} row

C. Last column

D. Last row and last column

175. Once a Karnaugh map is filled in, simplifying the SOP expression involves

A. circling the largest number of 1's possible.

B. Circling the largest number of 1's such that a group contains 1, 2, 4 ,8 etc., 1s.

C. None

D. Both

176. While taking 1s into groups in Karnaugh map, we have consider them only once (Y/N).

177. The process of building a circuit from a description such as a Boolean equation or truth table is _____.

A. Karnaugh mapping B. Gating

C. Synthesis D. Analysis

178. A product term in a Boolean expression where all variables appear once in true or compliment form is a _____.

A. Maxterm B. Minterm

C. DeMorgan D. Canonterm

179. How many adjacent cells in a Karnaugh map must be circled to eliminate 3 variables?

A. 2 B. 3

C. 6 D. 8

180. How many literals will be eliminated if adjacent cells in a Karnaugh map , where grouped adjacent cells are 8?

A. 2 B. 3

C. 6 D. 8

181. The cell marked 6 in 4-variable K-Map represent minterm 6 or the maxterm 6 having the following binary value of variables A, B, C and D. Select correct option:

A. A=1, B=1, C=0, D=0 B. A=0, B=1, C=1, D=0

C. A=0, B=0, C=1, D=1 D. A=1, B=0, C=0, D=1

182. An example of SOP expression is

A. A + B(C + D)

B. A′B + AC′ + AB′C

C. (A′ + B + C)(A + B′ + C)

D. both (a) and (b)

183. Which of the following terms is not adjacent to others?

A. wxyz′ B. w′xyz′

C. wxy′z D. w′x′y′z′

E. wx′yz

184. A group of eight 1s in a four variable Karnaugh map gives prime implicant with __ literals while the same with three variable Karnaugh map gives prime implicant with ___ literals.

A. 1, 0 B. 1, 1(constant 1)

C. 2, 0 D. None

185. A group of eight 1s in a four variable Karnaugh map gives prime implicant with __ literals while the same with six variable Karnaugh map gives prime implicant with ___ literals.

A. 1, 0 B. 1, 3
C. 2, 0 D. None

186. The Boolean expression A + B′ + C is
A. a sum term B. a literal term
C. a product term D. a complemented term

187. The Boolean expression AB′CD′ is
A. a sumterm B. a product term
C. a literal term D. always 1

188. Number of cells in Karnaugh map will be same as number vertices of an equivalent Boolean cube (Y/N).

189. Find odd man out of the following with regard to grouping of 1s with Karnaugh map.
A. 4 B. 32
C. 31 D. 64

190. Find odd man out of the following with regard to grouping of maxterms with Karnaugh map.
A. 4 B. 8
C. 7 D. 2

191. A logic circuit with an output $X = \bar{A}BC + A\bar{B}$ consists of _____.
A. two AND gates, two OR gates, two inverters
B. three AND gates, two OR gates, one inverter
C. two AND gates, one OR gate, two inverters
D. two AND gates, one OR gate

192. Following is standard POS expression
$(A+\bar{B}+C+\bar{D})(A+\bar{B}+C+\bar{D})(A+B+\bar{C}+\bar{D})$
$(A+B+C+\bar{D})(A+\bar{B}+\bar{C}+D)$
A. True
B. False

193. In a four variable Karnaugh map, total number cells for representing both for minterms and maxterms are:
A. 8 B. 16
C. Half D. All 1s

194. BCD to 7-Segment decoder has
A. 3 inputs and 7 outputs
B. 4 inputs and 7 outputs
C. 7 inputs and 3 outputs
D 7 inputs and 4 outputs

195. When minterms of a four variable Karnaugh map with don't cares are grouped, we have got the SOP as: A′B′+AC. What is minimum possible don't cares that are seen in Karnaugh map?
A. 1 B. 4
C. 8 D. None

196. When minterms of a four variable Karnaugh map are grouped along with don't cares, the resultant Boolean equation is observed to be 1. Then, in worst case how many cells may be don't cares.
A. 1 B. 15
C. 8 D. None

197. When minterms of a four variable Karnaugh map are grouped along with don't cares, the resultant Boolean equation is observed to be 1. Then, in the best case how many cells may be having don't cares.
A. 1 B. 15
C. 8 D. None

198. If we group 2^m minterms in an n variable Karnaugh map, then worst possible number of cells with don't cares can be
A. 2^m B. 2^m-1
C. m–1 D. n

199. If we group 2^m minterms along with don't cares in an n variable Karnaugh map, then best possible number of cells with don't cares in that group can be
A. 2^m B. 2^m-1
C. m–1 D. 1

200. Some grouping of minterms along with don't cares gave prime implicant with m-n terms in an n variable system. Then the worst possible number of don't care cells in the grouping can be
A. m-n–1 B. 2^m-1
C. $2^{m-n}-1$ D. None

201. Some grouping of minterms along with don't cares gave prime implicant with m-n terms in an n variable system. Then the best possible number of don't care cells in the grouping can be ___
A. m–n–1 B. 2^m-1
C. 2^{m-n}-1 D. 1

202. In an n variable Karnaugh map more than half cells are observed to be having don't care type. If we group half of the cells of Karnaugh map, then minimum possible number of don't care cells in it are:
A. 1 B. n/2
C. 2^n D. We cannot say

203. In an n variable Karnaugh map more than half cells are observed to be having don't care type. If we group half of the cells of Karnaugh map, then maximum possible number of don't care cells in it are:
A. 1 B. $2^{n/2}$ C. 2^n D. 2^{n-1}

204. In an n variable Karnaugh map, $2^n/4$ minterms are grouped then the respective prime implicant contains ___ literals.

A. 4 B. n–2
C. n/2 D. None

205. In an n variable Karnaugh map (where n is 2), $2^n/4$ minterms are grouped then the respective prime implicant contains ___ literals.
A. 4 B. 1
C. n/2 D. None

206. In an n variable system, if four 1s are grouped then the respective prime implicant contains _____ less than n literals.
A. 1 B. 4
C. n/4 D. 2

207. In an n variable Karnaugh map, two groups of $2^n/2$ minterms with overlap is seen. Then, the total number of literals in resulting SOP equation are __
A. n/2 B. 2n–1
C. 2n–2 D. None

208. In an n variable Karnaugh map, two groups of $2^n/4$ minterms with overlap is seen. Then, the total number of literals in resulting SOP equation are __
A. n/2 B. 2n–1
C. 2n–4 D. None

209. In an n variable Karnaugh map, two groups of $2^n/2$ minterms without any overlap is seen. Then, the total number of literals in resulting SOP equation are __
A. n/2
B. 2n–1
C. 2n-2
D. 1(The resulting equation itself 1)

210. In an n variable Karnaugh map, two groups of $2^n/2$ minterms with overlap are found. How many cells are common to them?
A. n/4 B. $2^n/4$
C. n/2 D. None

211. In an n variable Karnaugh map, two groups of $2^n/2$ minterms with overlap are found. What will be percentage of overalp?
A. 25% B. 50%
C. 75% D. None

212. In an n variable Karnaugh map, two groups of $2^n/4$ minterms without any overlap is seen. Then, the total number of literals in resulting SOP equation are
A. n/2
B. 2n-4
C. 2n-2
D. 1(The resulting equation itself 1)

213. In an n variable Karnaugh map, a group of minterms may contain ___ number of cells.
A. 2n B. 2^n
C. 3n D. 3^n

214. Find odd man out with respect to an n variable Karnaugh map and grouping of minterms.
A. 4 B. 2^n
C. $2^n/2$ D. 3^n

215. In an n variable Karnaugh map simplification, SOP equation is reported to be having two product terms with n-1 literals. Then, size of the minterm groups is ___
A. n cells B. 2n cells
C. $2^n/2$ cells D. None

216. In a Karnaugh map, a cell represents a minterm or maxterm or don't care. It is observed that the minterm or maxterms are having four literals. Then, how many cells are seen in the Karnaugh map?.
A. 4 B. 8
C. 16 D. 64

217. In a 6 variable Karnaugh map, center most 2x2 cells are found to be 1s in all the four planes (or stack of Karnaugh map parts). How many literals will be seen in the resulting product term?.
A. 1 B. 2
C. 4 D. 16

218. A function F=∑(0,1,2,4,5,6,8,9,12,13,14) is simplified using Karnaugh map. Possible total number of literals in the simplified equation for F is
A. 16 B. 12
C. 5 D. 6
(Answer: F becomes Y'+W'Z'+X'Z)

219. A function F(W,X,Y,Z)=∑(0,1,2,4,5,6,8,9,12,13,14) is simplified using Karnaugh map. Possible simplified equation of F contains
A. Two terms with two literals
B. One term with one literal
C. Total three terms
D. All true

220. In a three variable Karnaugh map, assuming three groups of 1s with two cells are having one cell common to all three groups. Then, possible number of literals in the resulting simplified SOP equation are
A. 4 B. 5
C. 6 D. 3

221. In a three variable Karnaugh map, assuming three groups of 1s with two cells are having one cell common to all three groups. Then, possible number of non-primed literals (literals with complement) in the resulting simplified SOP equation can be:

A. 6 B. 0
C. 2 D. All

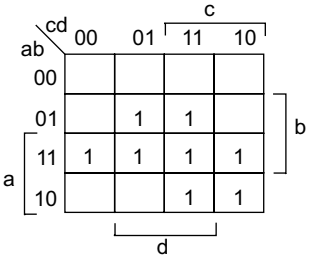
222. In a four variable Karnaugh map, we found three groups of 1s size 2x2 cells along the principal diagonal with 1 common cell to each of the group. Then, the possible number of cells in the simplified SOP equation is

A. 4 B. 5
C. 6 D. 7

223. The minimum cover of F(A, B, C, D) = ∑(0, 1, 4, 5, 712, 14, 15) does not contain

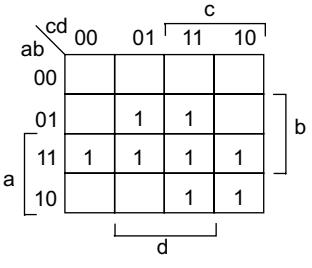
A. AC B. A′C′
C. ABD′ D. BCD

224. When the following Karnaugh map is used to find SOP equation. Number of product terms are:



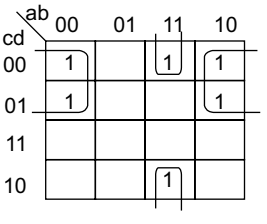
A. 4 B. 3
C. 6 D. 8

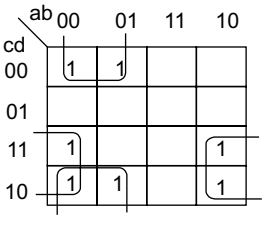
225. When the following Karnaugh map is used to find SOP equation. Assuming that a group is having singe 1, then number of product terms with four 1s are

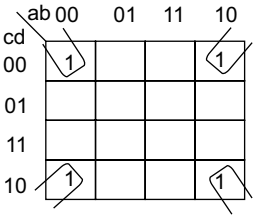


A. 4 B. 2
C. 6 D. 8

226. Assuming simplified Boolean equations (in SOP) are found for each of the following Karnaugh maps. Possible number of literals in each of them are:

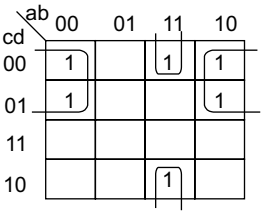


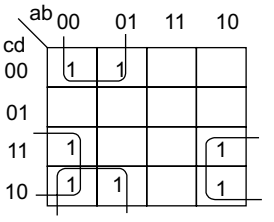


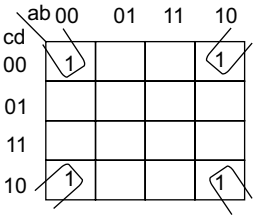


A. 5,9,10 B. 5,6,2
C. 5,4,2 D. 12,2,4

227. Assuming simplified Boolean equations (in SOP) are found for each of the following Karnaugh maps. Which one contains single product term?







A. First one B. Second one
C. Third one D. None

228. After simplifying F(A,B,C,D)=∑ m(0,2,3,6,8,12,13,15), the number of groups with two 1s are

A. 2 B. 3
C. 4 D. 5

229. After simplifying F(A,B,C,D)=∑ m(0,2,3,6,8,12,13,15), the number of product terms with three literals are

A. 2 B. 3
C. 4 D. 5

230. After simplifying F(A,B,C,D)=∑ m(0,2,3,6,8,12,13,15), the number of terms with two literals are

A. 0 B. 3
C. 4 D. 5

231. After simplifying F(A,B,C,D,E)=∑ m(5, 7, 13, 15, 21, 23, 29, 31), the number of groups with two 1s are

A. 0 B. 3
C. 4 D. 5

232. After simplifying F(A,B,C,D,E)=∑ m (5, 7, 13, 15, 21, 23, 29, 31), the number of terms with two literals are

A. 0 B. 1
C. 4 D. 5

233. From Karnaugh map, we found F(A,B,C)=A ⊕ B ⊕ C, then the number of 1s in the Karnaugh map are

A. 2 B. 3

C. 4 D. 5

234. From Karnaugh map, we found F(A,B,C)=A ⊕ B ⊕ C, then the number of groups having four 1s in the Karnaugh map are

A. 0 B. 3

C. 4 D. 5

235. ___ are the terms which cannot be simplified further.

A. Minterms B. Maxterms

C. Canonical terms D. Prime implicants

236. Prime implicants of Q=A'B'C'D' + A'B'C'D + AB'CD' + ABCD'

A. ABC, AB'D B. A'B'C', ACD'

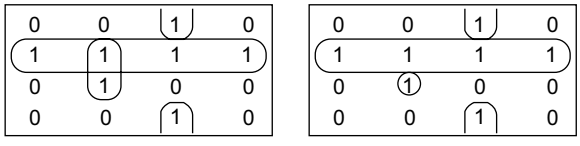
C. AB'C', ACD D. ABC, A'CD'

237. In a four variable Karnaugh map for F(A, B, C, D), if middle two rows fully contains 1s and don't cares then possible prime implicant is

A. AB B. AC

C. B D. A

238. Following two alternative groupings are suggested in a four variable Karnaugh map, which one is preferred?



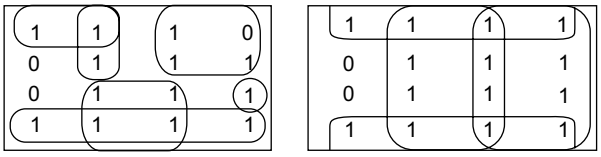
A. First one as it gives less number prime implicants

B. First one as it gives prime implicants with less number of total literals

C. Second as it gives less number of prime implicants.

D. None

239. Following two alternative groupings are suggested in a four variable Karnaugh map, which one is preferred?



A. First one as it gives less number prime implicants

B. Second one as it gives prime implicants with less number of total literals

C. Second as it gives less number of prime implicants.

D. None

240. Simplification of a four variable Karnaugh gave SOP equation as A+B+D′, then number groups eight ones are

A. 5 B. 6

C. 3 D. 4

241. In a Karnaugh map, number of maxterms and minterms are found to be exactly same and no don't care terms are available. All the minterms are adjacent and maxterms also adjacent. Then,

A. SOP equation contains single literal and single term.

B. POS equation contains single literal and single term.

C. Both are true.

D. None

242. There exists m minterms in a n-variable system. If related Boolean equation is represented in canonical form then total number of literals are:

A. mn B. mn/2

C. mn' D. None

243. If an n-variable Karnaugh map contains half of its elements as 1s then total number of literals in the resulting Boolean equation when represented in minterms form and each term is canonical

A. m2 B. $2^n/2n$

C. n(n-1)/2 D. None

244. Most important Boolean postulates which is basis for Karnaugh map grouping is

A. AA=1 B. AA′=A

C. A+A′=1 D. None

245. Most important Boolean postulates which is basis for Karnaugh map grouping is

A. AA=1 B. AA′=0

C. A+A′=A D. None

246. Q(A,B,C)= ABC+ABC′ + A′BC is equivalent to

A. AB+A′BC B. BC+ABC′

C. Both are valid D. None is valid

247. In an n variable Karnaugh map simplification, a product term in SOP is observed to be having m literals, then number adjacent cells which lead to this product term is:

A. m–n B. n–m

C. 2^{n-m} D. 2^{m-n}

248. In a SOP equation of a four variable system, a product term is identified to be having only two literals. If we expand the same into canonical terms, how many minterms we may get?.

A. 2 B. 4/2

C. 4 D. None

249. In a four variable (A,B,C,D) Karnaugh map, we have found four groups of two 1s which are horizontal in nature and one in each row . Each group contains off-diagonal element of that row and element in next column. Then the resulting Boolean equation in SOP form is

A. ABC + ABD + ABC′ +BCD′

B. ABC + AB′C + ABD′ + ACD

C. A′B′D′ + A′BC+ AB′C′+ABD

D. None

250. In a four variable (A,B,C,D) Karnaugh map, we have found four groups of two 1s which are horizontal in nature and one in each row . Each group contains off-diagonal element of that row and element in next column. Then prime implicant is:

A. ABC B. ABD′

C. ABD D. ABC'

251. In a four variable (A,B,C,D) Karnaugh map, we have found four groups of two 1s which are vertical in nature and one in each row . Each group contains off-diagonal element of that row and element below it. Then the resulting Boolean equation in SOP form is

A. ABC + ABD + ABC′ +BCD′

B. B′C′D′ +A′CD′+BCD+AC′D

C. A′B′D′ + A′BC+ AB′C′+ABD

D. None

252. In a four variable (A,B,C,D) Karnaugh map, we have found four groups of two 1s which are vertical in nature and one in each row. Each group contains off-diagonal element of that row and element below it. Then prime implicant is:

A. ABC B. ABD

C. A'CD' D. None

253. A distinguished cell

A. Is the one which is covered by more than one prime implicants

B. Canonical form

C. Is the one which is covered by exactly one prime implicant

D. None

254. If all the elements of both the diagonals of a 4 variable Karnaugh map are 1s, then the number of terms in the resulting SOP are

A. 3 B. 8

C. 4 D. 2

255. If all the elements of both the diagonals of a 4 variable Karnaugh map are 1s, then the number of literals in the resulting SOP are

A. 3 B. 8

C. 4 D. 2

256. If all the elements of both the diagonals of a 6 variable Karnaugh map are 1s in all of the four planes, then the number of terms in the resulting SOP are

A. 3 B. 8

C. 4 D. 2

257. If all the elements of both the diagonals of a 6 variable Karnaugh map are 1s in all of the four planes, then the number of terms in the resulting SOP are

A. 3 B. 8

C. 4 D. 2

258. If all the elements of middle two rows and columns of a 4 variable Karnaugh map are 1s then the resulting SOP equation contains

A. Two literals

B. Two product terms

C. All literals are in their prime form

D. All

259. If in each plane, all the elements of middle two rows and columns of a 6 variable Karnaugh map are 1s then the resulting SOP equation contains

A. Two literals

B. Two product terms

C. All literals are in their prime form

D. All

260. Number essential prime implicants of a three variable Karnaugh map with minterms 1,2,3 and 6 as 1s

A. 1 B. 3

C. 4 D. 2

261. For F(A,B,C)=Σm(1,2,3,6), essential PI

A. AC B. A′C

C. A′C′ D. A′B

262. F(A,B,C) = Σm(0,1,6,7) in simplified form is

A. (A+B)′ B. (A ⊕ B)

C. (A ⊕ B)′ D. None

263. Number of literals in the SOP equation of F(A,B,C) = Σm(0,1,2,3,5,6,7)

A. 6 B. 5

C. 3 D. 4

264. Number of prime literals in the SOP equation of F(A,B,C)=Σm(0,1,2,3,5,6,7)

A. 6 B. 5

C. 3 D. 2

265. How many gates are needed from the SOP equation of F(A,B,C) = Σm(0,1,2,3,5,6,7)

A. 3 two input AND and one 3 input OR

B. 3 input OR
C. 3 input OR and NOT
D. None

266. From a four variable Karnaugh, a simplied SOP equation is given as A′B′C′D + A′BC + AC′D′ + AB′D′, then number minterms with 1s are

A. 10 B. 9
C. 7 D. 6

267. Some expression can be both SOP and POS. Which of the following are such expressions?

i. x′ ii. xy + x′z + xy′z′
iii. xyz′ iv. x + y + z′
A. (i) and (iii) only B. (i) and (iv) only
C. (iii) and (iv) only D. (ii), (iii) and (iv) only
E. (i), (iii) and (iv) only

268. Given a Boolean function F(x,y,z) = z·(x′ + y) + x·y. Which of the following is correct?

A. F(x,y,z) = Σ m(1, 5, 6, 7)
B. F(x,y,z) = Σ m(1, 3, 6, 7)
C. F(x,y,z) = Σ m(3, 5, 6, 7)
D. F(x,y,z) = Σ m(2, 5, 6, 7)
E. F(x,y,z) = Σ m(0, 1, 3, 4)

269. Number PIs and EPIs in the following Karnaugh map

0	0	0	1
1	0	X	X
X	0	X	1
0	0	X	X

A. 3, 1 B. 4, 1
C. 4, 2 D. 5, 2
E. 5, 1

270. SOP of the following Karnaugh map assuming variables are W,X,Y,Z.

0	0	0	1
1	0	X	X
X	0	X	1
0	0	X	X

A. WX′ +W′X′Z B. WX′ + X′Z
C. YZ′ +XY′Z′ D. YZ′+XZ′
E. YZ+XZ

271. POS of the following Karnaugh map assuming variables are W,X,Y,Z.

0	0	0	1
1	0	X	X
X	0	X	1
0	0	X	X

A. Z + X′Y′ B. XZ′+YZ′
C. Z′(X+Y) D. (X+Y)(Y+Z)(Z′+Y)

272. Number of EPIs and PIs in a Karnaugh map with m (0,2,4,5,6,7,8,10,13,15)

A. 3, 3 B. 3, 4
C. 4, 3 D. 4, 2
E. 3, 2

273. Number of product terms in simplified a Karnaugh map with m(0,2,4,5,6,7,8,10,13,15)

A. 1 B. 2
C. 3 D. 4
E. 5

274. The following three Boolean functions are given:

F(A,B,C,D,E) = ΠM(0, 1, 2, 4)
G(A,B,C,D,E) = Σm(0, 2, 4, 6)
H(A,B,C,D,E) = ΠM(3, 4, 5)
What is the Boolean function F · G · H?

A. Σm(4) B. Σm(6)
C. Σm(1, 3, 5, 6) D. Σm(0, 1, 2, 3, 4, 5, 6)
E. None of the above

275. In a 5 variable system, which of the following minterms differs in one literal?

A. m1, m4 B. m7, m9
C. m16, m16 D. m18, m20
E. m19, m23

276. Given the following Karnaugh map, __ is not a prime implicant.

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	1	0
11	0	1	1	1
10	0	1	0	0

A. A′B′D′ B. A′BC
C. ABC D. A′B′C′

277. Given the following Karnaugh map, __ is not a prime implicant.

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	1	0
11	0	1	1	1
10	0	1	0	0

A. A′B′D′ B. BD
C. ABC D. A′B′C′

278. Given the following Karnaugh map of a function F(A,B,C,D), then W' in SOP form.

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	1	0
11	0	1	1	1
10	0	1	0	0

A. A.B′.C + A′.B.D′ + A.C′.D′ + B′.C.D

B. A.B.C′ + A.B′.D′ + A′.C.D′ + B.C′.D

C. A.B′.C + A′.B.D′+ A.C′.D + B′.C.D′

D. A.B.C + A.B.D′+ A.C′.D′ + B.C′.D

E. A.B.C + A′.B.D′+ A.C′.D′ + B.C.D′

279. For a six-variable Boolean function, which of the following sets of minterms can be combined into a single product term?

A. m0 and m63

B. m0, m5, m10 and m15

C. m16, m17, m32 and m33

D. m21, m29, m53 and m63

E. m60, m61, m62 and m63

280. Find correct statement

A. An OR function can be created from only AND gates.

B. An NOR function can be created from only NAND gates.

C. Every product-of-sums expression is a product-of-maxterms expression.

D. A minimal SOP expression can be obtained by including only the essential prime implicants but leaving out all the non-essential prime implicants.

E. None of the above.

281. Given this four-variable Boolean function F:

F = C′.D + A′.B.D + A.B.C′+ B′.C′.D′+ A′.B′.C′

How many distinct minterms does the sum-of-minterms expression for F contain?

A. 8 B. 9

C. 10 D. 11

E. 12

282. Given a K-map of a 10-variable Boolean function and a particular prime implicant on the K-map contains 64 minterms. How many literals are there in the product term corresponding to that prime implicant?

A. 4 B. 6

C. 16 D. 32

E. 24

283. From a Karnaugh map, a function is given as: F(a,b,c,d) = (b ⊕ d). Then select possible structure of the Karnaugh map.

A.

			0
	1	1	0
	1	1	0
			0

B.

	1	1	1
	1	1	1

C.

0			0
	0	0	
	0	0	
0			0

D.

1			1
	1	1	
	1	1	
1			1

284. From a Karnaugh map, a function is given as: F(a,b,c,d) = (b ⊕ d)' . Then select possible structure of the Karnaugh map.

A.

			1
	1	1	1
	1	1	1
			1

B.

	1	1	1
	1	1	1

C.

			1
	1	1	
	1	1	
			1

D.

1			1
	1	1	
	1	1	
1			1

285. From a Karnaugh map, a function is given as: F(a,b,c,d)=(a ⊕ c)'. Then select possible structure of the Karnaugh map.

A.

1	1		
1	1		
		1	1
		1	1

B.

		1	1
		1	1
1	1		
1	1		

C.

			1
	1	1	
	1	1	
			1

D.

1			1
	1	1	
	1	1	
1			1

286. EPIs and PIs for the following Karnaugh map:

1			1
	1	1	
	1	1	
1		1	1

A. bd, bd′, acd B. bd, b′d′, acd

C. bd, b′d′, acd, acd′ D. abd, bc, bd

287. From a Karnaugh map, a function is given as: F(a,b,c,d)=(c ⊕ d). Then select possible structure of the Karnaugh map.

A.

			1
	1	1	1
	1	1	1
			1

B.

	1	1	1
	1	1	1

C.

			1
	1	1	
	1	1	
			1

D.

1	1	1	1
1	1	1	1

288. Find odd one out of the following.

A. Sum term B. Canonical term

C. Product term D. literal

289. In minterms or maxterms literals will be always in non-compliment form (Y/N)

290. In an n variable input digital system with one output, sum of minterms values will be at most

A. log2(n) B. n

C. 2^n D. None

291. The function $f(A,B,C) = AB + AC' + A'C$ is equivalently mentioned as

A. Σm(0,2,3,4) B. Σm()

C. πm(1,3,4,6,7) D. None

292. The function $f(A,B,C) = A(A + C')$. It can equally represented as

A. Σm(0,1,2,3,4) B. ΣM(0,1,2,3)

C. ΣM(1,2,3,9) D. π M(0,1,2,3)

293. The function $f(A,B,C)$ = A. It can be equally represented as

A. Σm(0,1,2,3,4) B. ΣM(0,1,2,3)

C. ΣM(1,2,3,9) D. πM(0,1,2,3)

294. The function $f(A,B,C) = (A + C')$. It can be equally represented as

A. πm(0,1,2,3,4) B. ΣM(0,1,2,3)

C. ΣM(1,2,3,9) D. πM(1,3)

295. The function $f(A,B,C) = A(A + C')$. It can be equally represented as

A. Σm(4,5,6,7) B. ΣM(0,1,2,3)

C. ΣM(1,2,3,9) D. Σm(0,1,2,3)

296. A function f(A,B,C) is found to be same as A after grouping 0s and don't cares in the Karnaugh map. Maximum possible number of don't care cells that are considered while deriving sum term as A is

A. 1 B. 2

C. 3 D. None

297. A function f(A,B,C) is found to be same as A after grouping 0s and don't cares in the Karnaugh map. Maximum possible number of don't care cells that are considered while deriving sum term as A is

A. 1 B. 2

C. 3 D. None

298. A function f(A,B,C,D) is found to be same as A after grouping 0s and don't cares in the Karnaugh map. Maximum possible number of don't care cells that are considered while deriving sum term as A is

A. 1 B. 2

C. 3 D. 7

299. A function f(A,B,C,D) is found to be same as A+C after grouping 0s and don't cares in the Karnaugh map. Maximum possible number of don't care cells that are considered while deriving sum term as A+C is

A. 1 B. 2

C. 3 D. 7

300. A function f(A,B,C,D,E,F) is found to be same as AF' after grouping 1s and don't cares in the Karnaugh map. Maximum possible number of don't care cells that are considered while deriving product term as AF' is

A. 15 B. 2

C. 3 D. 7

301. A function f(A,B,C,D,E,F) is found to be same as AF' after grouping 1s and don't cares in the Karnaugh map. Minimum possible number of don't care cells that are considered while deriving product term as AF' is:

A. 1 B. 2

C. 3 D. 7

302. A function f(A,B,C) is found to be same as A(A+C') after grouping 0s and don't cares in the Karnaugh map. Maximum possible number of don't care cells that are considered while deriving POS equation A(A+C') is:

A. 1 B. 2

C. 5 D. 7

303. Minimizing using Karnaugh map

A. Increases number of levels in the resultant circuit

B. Reduces number of levels in the resultant circuit.

C. Gives circuits with two levels.

D. None

304. A minimal solution will never contain

A. Prime implicants

B. Essential prime implicants

C. Non-prime implicants

D. None

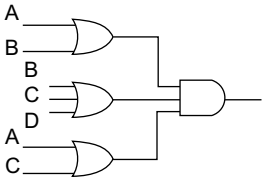
305. Non-prime implicant in the following Karnaugh map

CD \ AB	00	01	11	10
00				1
01			1	1
11		1	1	1
10		1	1	1

A. ABC′D B. AB
C. CD′ D. AB

306. _____ is invalid number of cells in a single group formed by the adjacent cells in K-map.
A. 2 B. 8
C. 12 D. 16

307. The diagram given below represents _____



A. Demorgans law B. Associative law
C. Product of sum form D. Sum of product form

308. A non-standard POS is converted into a standard POS by using the rule _____
A. $A + \overline{A} = 1$ B. $A\,\overline{A} = 0$
C. $1 + A = 1$ D. $A + B = B + A$

309. In the following K-Map which is redundant term that is used to avoid hazard?

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	1	1	1	1
10	1	1	0	0

A. AB B. AC’
C. BC D. None

310. A function F (A,B,C,D) in its minimalistic form is given as: (A+C)(A′ + D′). The redundant term that can be used to avoid hazard
A. BCD B. C′ + D
C. A + D D. None

311. A function F (A,B,C,D) in its minimalistic form is given as: (A + C)(A′ + D′). This has ___.
A. Static 1 hazard B. Static 0 hazard
C. Function hazard D. Dynamic hazard

312. A function F (A,B,C,D) in its minimalistic form is given as: (A+C)(A′ + D′)(B′+C′+D)(C+D′). Note the redundant term
A. A + B′ + D B. C′ + D
C. A′ + B′ + C’ D. A + B + C

313. A function F(x,y,z)=Σ(2,3,5,7) has ___ .
A. Static 1 hazard B. Static 0 hazard
C. Function hazard D. Dynamic hazard

314. The redundant term to be added to avoid hazard in function F(x,y,z) = Σ(2,3,5,7) is
A. x+y B. xz
C. yz D. None

315. The redundant terms to be added to avoid static 1 hazard in the function F(x,y,z,w) = x′y + xw + zw + yz’
A. yw B. xy
C. xw D. None

316. In a function F(x,y,z) = xz′ + yz, the variable whose transition gives static 1 hazard
A. x B. y
C. z D. All

317. The function F = AC′ + A′D+C′D has
A. Static 1 hazard B. Static 0 hazard
C. No hazard at all D. Dynamic hazard

318. Hazards can be avoided by
A. Inducing delays
B. Sample only when stable output is available
C. Avoiding asynchronous inputs
D. Using synchronized clock
E. Using redundant terms
F. All

319. The redundant terms that can be added to make the function F(A,B,C,D) = AC′ + A′D + C′D as hazard free
A. AB B. BD
C. Both D. None

320. Which of these statements are true?
A. A minterm equation is an unreduced sum-of-products Boolean expression.
B. A minterm equation requires the least number of logic gates when implemented.
C. A maxterm equation is an unreduced sum-of-products Boolean expression.
D. A maxterm equation is an unreduced product-of-sums Boolean expression.

321. Gray codes
A. are used to express negative Boolean numbers.
B. are used to order the cells in a Karnaugh map.
C. change only in last bit in sequential numbers.
D. make a halftone image when printed.

322. _____ is one of the examples of synchronous inputs.
A. J-K input B. EN input
C. Preset input (PRE) D. Clear Input (CLR)

323. _____ is one of the examples of asynchronous inputs.
A. J-K input B. S-R input
C. D input D. Clear Input (CLR)

324. The _____ input overrides the _____ input
A. Asynchronous, synchronous
B. Synchronous, asynchronous
C. Preset input (PRE), Clear input (CLR)
D. Clear input (CLR), Preset input (PRE)

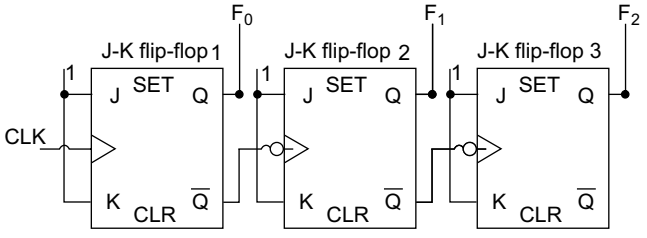
325. _____occurs when the same clock signal arrives at different times at different clock inputs due to propagation delay.
A. Race condition B. Clock Skew
C. Ripple Effect D. None of given options

326. Consider an up/down counter that counts between 0 and 15, if external input(X) is "0" the counter counts upward (0000 to 1111) and if external input (X) is "1" the counter counts downward (1111 to 0000), now suppose that the present state is "1100" and X=1, the next state of the counter will be _____
A. 0000 B. 1101
C. 1011 D. 1111

327. A 8-bit serial in / parallel out shift register contains the value "8", _____ clock signal(s) will be required to shift the value completely out of the register.
A. 1 B. 2
C. 4 D. 8

328. 5-bit Johnson counter sequences through____states.
A. 7 B. 10
C. 32 D. 25

329. Assume that a 4-bit serial in/serial out shift register is initially clear. We wish to store the nibble 1100. What will be the 4-bit pattern after the second clock pulse? (Right-most bit first.)
A. 1100 B. 0011
C. 0000 D. 1111



330. Above is the circuit diagram of _____.
A. Asynchronous up-counter
B. Asynchronous down-counter
C. Synchronous up-counter
D. Synchronous down-counter

331. The sequence of states that are implemented by a n-bit Johnson counter is
A. n+2 (n plus 2)
B. 2n (n multiplied by 2)
C. 2^n (2 raise to power n)
D. n^2 (n raise to power 2)

332. At T_0 the value stored in a 4-bit left shift was "1". What will be the value of register after three clock pulses?
A. 2 B. 4
C. 6 D. 8

333. An 8-bit serial in / parallel out shift register contains the value "8", _____ clock signal(s) will be required to shift the value completely out of the register.
A. 1 B. 2
C. 4 D. 8

334. When both the inputs of edge-triggered J-K flop-flop are set to logic zero _____
A. The flop-flop is triggered
B. Q=0 and Q'=1
C. Q=1 and Q'=0
D. The output of flip-flop remains unchanged

335. How many stats does a module 4 counter have?
A. 1 B. 2
C. 4 D. 16

336. How will a serial in and out shift register accept data serially?
A. One bit at a time B. 8 bit
C. Only after a load plus D. Only after being clear

337. The invalid state of SR latch occur when
A. S=1, R=0 B. S=0, R=1
C. S=1,R=1 D. S=0,R=0

338. To serially shift a byte of data in to a shift register there must be
A. 1 clock plus
B. One load plus
C. 8 clock plus
D. One clock plus for each in the data

339. What is the decimal value of the terminal count of 4 bit binary count?
A. 10 B. 12
C. 15 D. 16

340. Divide- by-10 Johnson counter needs __ flip flops.
A. 5 B. 10
C. 4 D. None

341. Divide- by-10 ring counter needs __ flip flops.
A. 5 B. 10
C. 4 D. None

342. A multiplexer with its inputs from a register can be used to
A. Parallel to serial data B. Serial to serial
C. Serial data to parallel D. None

343. To parallel load a byte into a shift register we need
A. One clock pulse
B. One clock pulse for one bit
C. n/2 clock pulses.
D. None

344. S=1 and R=0, then Q(t+1) = _____ for positive edge triggered flip-flop
A. 0 B. 1
C. Invalid D. Input is invalid

345. If S=1 and R=1, then Q(t+1) = _____ for negative edge triggered flip-flop
A. 0 B. 1
C. Invalid D. Input is invalid

346. The operation of J-K flip-flop is similar to that of the SR flip-flop except that the J-K flip-flop _____
A. Does not have an invalid state
B. Sets to clear when both J = 0 and K = 0
C. It does not show transition on change in pulse
D. It does not accept asynchronous inputs

347. The minimum time for which the input signal has to be maintained at the input of flip-flop is called _____ of the flip-flop.
A. Set-up time
B. Hold time
C. Pulse interval time
D. Pulse stability time (PST)

348. Positive edge-triggered flip-flop changes its state when _____
A. Low-to-high transition of clock
B. High-to-low transition of clock
C. Enable input (EN) is set
D. Preset input (PRE) is set

349. Negative edge-triggered flip-flop changes its state when _____
A. Enable input (EN) is set
B. Preset input (PRE) is set
C. Low-to-high transition of clock
D. High-to-low transition of clock

350. ___the $\bar{Q}$ output of the last flip-flop of the shift register is connected to the data input of the first flip-flop.
A. Moore machine B. Mealy machine
C. Johnson counter D. Ring counter

351. _____ Q output of the last flip-flop of the shift register is connected to the data input of the first flip-flop of the shift register.
A. Moore machine B. Mealy machine
C. Johnson counter D. Ring counter

352. Which is not characteristic of a shift register?
A. Serial in/parallel in B. Serial in/parallel out
C. Parallel in/serial out D. Parallel in/parallel out

353. Assume that a 4-bit serial in/serial out shift register is initially clear. We wish to store the nibble 1100. What will be the 4-bit pattern after the second clock pulse? (Right-most bit first.)
A. 1100 B. 0011
C. 0000 D. 1111

354. The counter whose LSB flip-flop is driven by high logic with its inputs
A. counter with JK FFs B. counter with RS FFs
C. counter with D FFs D. None

355. The counter whose LSB flip-flop only receives clock
A. Synchronous B. Asynchronous
C. A ripple D. Ripple

356. Maximum possible modulus value in a counter with K flip-flops
A. K B. 2K
C. 2^K-1 D. K/2

357. A ripple counter with ___ FF at its LSB will be having its complement output will be connected to its inputs.
A. RS B. SR
C. JK D. D

358. Number of counts in a counter cycle
A. Norm B. Magnitude of counter
C. Modulus D. None

359. Since only the first flip-flop of a ripple counter is controlled by a clock, the counter is _____?
A. Binary B. Asynchronous
C. Synchronous D. Modulus-n

360. ___ is the mod number of a counter with 5 FFs.
A. 10 B. 9
C. 32 D. 16

361. Find odd man out of the following
A. Serial counter B. Ripple counter

C. Asynchronous counter
D. Synchronous counter

362. Without using jumper wires between inputs and outputs, IC7493 can operate as
A. Mod 2 and mod 10 B. Mod 2 and mod 9
C. Mod 2 and mod 8 D. None

363. In order to use as a ___ counter, we have to connect Q0 and CP1.
A. 2 B. 8
C. 6 D. 12

364. To make it work like a mod 12 counter, we have to connect
A. Q0 and CP1 B. Q1 and CP2
C. CP1 and CP0 D. None

365. Using IC 7493, we can realise mod ___ counter.
A. 7 B. 6
C. 12 D. 31

366. Highest count which we can achieve in IC7493
A. 14 B. 15
C. 16 D. 8

367. To make IC7493 to work like a mod 6 counter, we have to MR1, MR2
A. Q0, Q1 B. Q1, Q2
C. Q2, Q3 D. Q2, Q0

368. If we connect Q2, Q0 to MR1, MR2 lines, then we get mod __ counter.
A. 11 B. 12
C. 5 D. 6

369. The flip-flop which has indeterminate state
A. SR B. JK
C. Master slave D. D

370. The flip-flop which does not have inputs which complements the flip-flop content
A. T B. JK
C. Master Slave JK D. D

371. The flip-flop which does not have inputs which does not change the flip-flop state, i.e., flip-flop which does not have no-change situation
A. T B. JK
C. Master Slave JK D. D

372. The flip-flop whose indeterminate state is changed to complement state
A. SR B. JK
C. T D. D

373. Drawbacks of asynchronous counters is
A. Speed B. Rippling effect
C. Both D. None

374. Synchronous counters
A. Faster than their asynchronous counterparts
B. Accurate compared to their asynchronous counterparts
C. All FFs receives clock at the same time
D. Complex compared to their asynchronous counterparts
E. All are valid

375. In Serial shift register that uses D flip-flops, clock is broadcaster to all flip-flops (Y/N)

376. To make JK flip-flop to change from 0 to 0, we have to apply ___ on its J, K lines
A. 0,0 B. 0,1
C. Both D. None

377. To make SR flip-flop to change from 0 to 0, we have to apply ___ on its S,R lines.
A. 0,0 B. 0,1
C. Both D. None

378. In a multiprocessor system, _____ is used to connect processors.
A. Switch B. Cable
C. Bus D. Computer network

379. In a multiprocessor system, inter-process communication is carried out through
A. Data bus B. Address bus
C. Shared memory D. Message passing

380. Stack machines are
A. One addressed machines
B. Zero addressed machines
C. Two address machines
D. Akin to Penitium machines ISA

ANSWER KEY

1. A	**2.** C	**3.** B	**4.** D
5. A	**6.** B	**7.** B	**8.** B
9. B	**10.** C	**11.** C	**12.** C
13. C	**14.** D	**15.** C	**16.** D
17. C	**18.** B	**19.** D	**20.** A
21. D	**22.** E	**23.** C	**24.** D
25. C	**26.** B	**27.** B	**28.** C
29. B	**30.** A	**31.** D	**32.** B
33. E	**34.** C	**35.** B	**36.** C
37. D	**38.** C	**39.** C	**40.** B
41. B	**42.** A	**43.** E	**44.** C
45. D	**46.** A	**47.** C	**48.** B

49. D **50.** A **51.** C **52.** B
53. A **54.** A **55.** A **56.** C
57. C **58.** C **59.** A,E **60.** E
61. E **62.** E **63.** B **64.** E
65. C **66.** B **67.** C **68.** A
69. B **70.** C **71.** A **72.** D
73. C **74.** B **75.** D **76.** B
77. D **78.** C **79.** C **80.** D
81. B **82.** D **83.** C **84.** E
85. B **86.** A **87.** B **88.** C
89. C **90.** A **91.** B **92.** C
93. C **94.** B **95.** B **96.** C
97. A **98.** B **99.** C **100.** A
101. C **102.** D **103.** C **104.** C
105. D **106.** C **107.** B **108.** B
109. B **110.** D **111.** D **112.** B
113. C **114.** C **115.** C **116.** C
117. C **118.** C **119.** A **120.** E
121. D **122.** D **123.** A **124.** D
125. B **126.** A **127.** A **128.** B
129. B **130.** C **131.** C **132.** A
133. A **134.** A **135.** C **136.** B
137. B **138.** C **139.** C **140.** B
141. B **142.** A **143.** B **144.** A
145. C **146.** C **147.** C **148.** D
149. A **150.** C **151.** B **152.** C
153. A **154.** B **155.** C **156.** D
157. Y **158. N** **159. N** **160.** B
161. B **162.** C **163.** B **164.** C
165. B **166.** A **167.** D **168.** B
169. B **170.** B **171.** D **172.** D
173. D **174.** B **175.** B **176. N**
177. D **178.** B **179.** D **180.** B
181. B **182.** D **183.** D **184.** B
185. B **186.** A **187.** B **188. Y**
189. C **190.** C **191.** C **192.** A
193. B **194.** B **195.** A **196.** B
197. A **198.** B **199.** D **200.** C
201. D **202.** A **203.** D **204.** B
205. B **206.** D **207.** C **208.** C
209. D **210.** B **211.** B **212.** D
213. B **214.** D **215.** C **216.** C
217. B **218.** C **219.** D **220.** C
221. D **222.** C **223.** A **224.** B
225. B **226.** C **227.** C **228.** D
229. D **230.** A **231.** A **232.** B
233. C **234.** A **235.** D **236.** B
237. C **238.** B **239.** B **240.** C
241. C **242.** A **243.** B **244.** C
245. B **246.** C **247.** C **248.** C
249. C **250.** C **251.** B **252.** C
253. C **254.** D **255.** C **256.** D
257. C **258.** D **259.** D **260.** D
261. B **262.** C **263.** C **264.** D
265. C **266.** D **267.** E **268.** B
269. C **270.** D **271.** C **272.** D
273. B **274.** B **275.** E **276.** B
277. D **278.** A **279.** E **280.** D
281. E **282.** A **283.** C **284.** D
285. D **286.** C **287.** D **288.** D
289. Y **290.** C **291.** C **292.** D
293. D **294.** D **295.** A **296.** C
297. C **298.** D **299.** C **300.** A
301. A **302.** C **303.** C **304.** C
305. A **306.** C **307.** C **308.** B
309. A **310.** B **311.** B **312.** D
313. A **314.** C **315.** A **316.** C
317. C **318. F** **319.** C **320.** A
321. B **322.** A **323.** C **324.** A
325. C **326.** B **327.** D **328.** D
329. C **330.** C **331.** B **332.** D
333. A **334.** D **335.** C **336.** A
337. C **338.** C **339.** C **340.** B
341. A **342.** A **343.** B **344.** B
345. C **346.** A **347.** A **348.** A
349. D **350.** C **351.** D **352.** A
353. A **354.** A **355.** B,D **356.** C
357. D **358.** C **359.** B **360.** C
361. D **362.** C **363.** D **364.** A
365. D **366.** B **367.** B **368.** C
369. A **370.** D **371.** D **372.** D
373. C **374.** E **375. N** **376.** C
377. C **378.** C **379.** C **380.** B

Previous Years' GATE Questions

1. The smallest integer that can be represented by 8 bit number in 2's complement form is **(GATE 2013)**

A. −256 B. −128

C. −127 D. 0

2. In the following truth table, V=1 if and only if input is valid **(GATE 2013)**

D0	D1	D2	D3	X0	X1	V
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1

What function does the truth table represents?

A. Priority encoder B. Decoder

C. Multiplexer D. De-multiplexer

3. In a k-way set associative cache, the cache is divided into v sets, each of which contains k lines. The lines of a set are placed in sequence one after another. The lines in set s are sequenced after lines in set s+1. The main memory blocks are numbered 0 onwards. The main memory block numbered j must mapped to any one of the cache lines from **(GATE 2013)**

A. (j mod v)*k to (j mod v)*k + (k-1)

B. (j mod v)*k to (j mod v) + (k-1)

C. (j mod k) to (j mod k) + (v-1)

D. (j mod k)*v to (j mod k)*v + (v-1)

4. Which of the following functions does not represent exclusive NOR of x and y? **(GATE 2013)**

A. xy + x'y' B. $x \oplus y'$

C. $x' \oplus y$ D. $x' \oplus y'$

5. Consider the following sequence of micro operations **(GATE 2013)**

MBR ← PC

MAR ← X

PC ← Y

Memory ← MBR

Which one of the following is a possible operation performed by the sequence?

A. Instruction fetch

B. Operand fetch

C. Conditional branch

D. Initiation of interrupt service

6. Consider an instruction pipeline with five stages without any branch prediction. Fetch instruction (FI), decode instruction (DI), Fetch operand (FO), Execute Instruction (EI), and Write operand (WO). The stage delays for FI, DI, FO, EI and WO are 5ns, 7ns, 10ns, 8ns, and 6 ns respectively. There are intermediate storage buffers at each stage and the delay for each buffer is 1ns. A program consisting of 12 instructions I1, I2, I3, I4, …I12 is executed in this pipelined processor. Instruction I4 is the only branch instruction and its branch target is I9. If the branch is taken during the execution of this program, the time (in ns) needed to complete the program is: **(GATE 2013)**

A. 132 B. 165

C. 176 D. 328

7. A RAM chip has a capacity of 1024 words of 8 bits each (1k × 8). The number of 2 × 4 decoders with enable line needed to construct a 16k × 16 RAM from 1K × 8RAM is **(GATE 2013)**

A. 4 B. 5

C. 6 D. 7

The following code segment is executed on a processor which allows only register operands in its instructions. Each instruction can have two source operands one destination operand. Assume that all variables are dead after this code segment

```
c=a+b
d=c*a
e=c+a
x=c*c
if (x>a) {
y=a*a
}
else
{
d=d*d
e=e*e
}
```

8. Suppose the instruction set architecture of the processor has only two registers. The only allowed compiler optimization is code motion, which moves statements from one place to another while preserving correctness. What is the minimum number of spills to memory in the compiled code? **(GATE 2013)**

A. 0 B. 1

C. 2 D. 3

9. What is the minimum number of registers needed in the instruction set architecture of the processor to

compile the code without any spill to memory. Do not apply any optimisation other than optimising register allocation. **(GATE 2013)**

A. 3 B. 4

C. 5 D. 6

A computer has 46 bit virtual address, 32 bit physical address, and a three level paged page table organisation. The page table base address stores the base address of the first level table (T1), which occupies exactly one page. Each entry in T1 stores the base address of a page of the second-level table (T2). Each entry of the table T2 stores the base address of a page of the third level table T3. Each entry of T3 stores a page table entry (PTE). The PTE is 32 bits size. The processor used in the computer has a 1MB 16-way set associative virtually indexed physically indexed tagged cache. The cache block size is 64 bytes.

10. What is the size of a page in KB in this computer? **(GATE 2013)**

A. 2 B. 4

C. 8 D. 16

Answer: It directly visible from the problem statement that is system is using 32 bit physical address space. Thus, 2^{32} B= 2^{2}*2^{30} B =4KB.

11. What is the minimum number of page colors needed to guarantee that no two synonyms map to different sets in the processor cache of this computer? **(GATE 2013)**

A. 2 B. 4

C. 8 D. 16

12. The truth table

X	Y	f(X,Y)
0	0	0
0	1	0
1	1	1
1	1	1

represents the Boolean function **(GATE 2012)**

A. X B. X+Y

C. $X \oplus Y$ D. Y

13. The decimal value 0.5 in IEEE single precision floating point representation has **(GATE 2012)**

A. Fraction bits of 000…000 and exponent value of 0

B. Fraction bits of 000…000 and exponent value of −1

C. Fraction bits of 100…000 and exponent value of 0

D. No exact representation

14. Register renaming is done in pipelined processors **(GATE 2012)**

A. As an alternative to register allocation at compile time

B. For efficient access to function parameters and local variables

C. To handle certain kinds of hazards

D. As part of address translation

15. What is the minimal form of the Karnaugh map shown below? Assume that X denotes a do not care term. **(GATE 2012)**

cd \ ab	00	01	11	10
00	1	X	X	1
01	X			1
11				
10	1			X

A. $\bar{b}\,\bar{d}$ B. $\bar{b}\,\bar{d} + \bar{b}\,c$

C. $\bar{b}\,\bar{d} + a\bar{b}\,\bar{c}\,d$ D. $\bar{b}\,\bar{d} + \bar{b}\,\bar{c} + \bar{c}\,\bar{d}$

16. Consider the virtual page reference string

1, 2, 3, 2, 4, 1, 3, 2, 4, 1

on a demand paged virtual memory system running on a computer system that has main memory size of 3 page frames which are initially empty. Let LRU, FIFO and OPTIMAL denote the number of page faults under the corresponding page replacement policy. Then, **(GATE 2012)**

A. OPTIMAL < LRU < FIFO

B. OPTIMAL < FIFO < LRU

C. OPTIMAL = LRU

D. OPTIMAL = FIFO

17. A computer has a 256 KByte, 4-way set associative, write back data cache with block size of 32 Bytes. The processor sends 32 bit addresses to the cache controller. Each cache tag directory entry contains, in addition to address tag, 2 valid bits, 1 modified bit and 1 replacement bit.

The number of bits in the tag field of an address is **(GATE 2012)**

A. 11 B. 14

C. 16 D. 27

Answer: Cache size is 256KB=216Bytes

Therefore, tag bits are 16, as cache uses 16 bit addresses.

18. The size of the cache tag directory is **(GATE 2012)**

A. 160 Kbits B. 136 Kbits

C. 40 Kbits D. 32 Kbits

Answer: Tag directory entry = 16+2+2=20bits

Number of Tag entries = 256KB/32*20 bits = 160Kbits

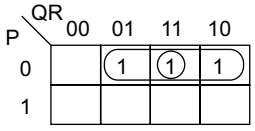
19. The simplified SOP (Sum of product) form of the Boolean expression $(P + \overline{Q} + \overline{R}) \cdot (P + \overline{Q} + R) \cdot (P + Q + \overline{R})$ is

A. $(\overline{P}Q + \overline{R})$ B. $(P + \overline{QP})$

C. $(\overline{PQ} + R)$ D. $(PQ - R)$

$f = (P + \overline{R})(P + \overline{Q})$

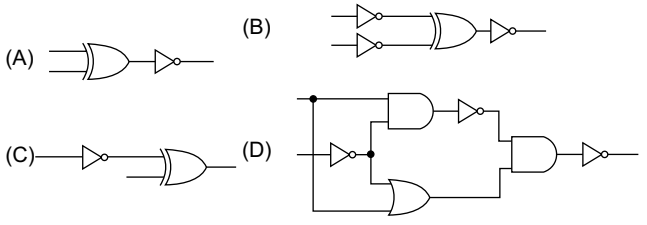
$= P - \overline{Q}\,\overline{R}$



First, we have prepare Karnaugh map as shown above and then simplified expression is arrived at. This is matching with option B.

20. Which one of the following circuits is NOT equivalent to a 2-input XNOR (exclusive NOR) gate?

(GATE 2011)



21. The minimum number of D flip-flops needed to design a mod-258 counter is **(GATE 2011)**

A. 9 B. 8

C. 512 D. 258

We want n value such that $2^n >= 258$. Thus, n = 9.

22. Let the page fault service time be 10ms in a computer with average memory access time being 20ns. If one page fault is generated for every 10^6 memory accesses, what is the effective access time for the memory?

(GATE 2011)

A. 21ns

B. 30ns

C. 23ns

D. 35ns

Answer: Effective Access time $=((10^6 - 1)*20ns + (10ms+20ns))/10^6 = (10^6 * 20ns + 10ms)/10^6$

$= (10^6 * 20*10^{-6}\ ms + 10ms)/10^6$

$= (20 + 10)ms/10^6 = 30ns$

23. Consider a hypothetical processor with an instruction of type LW R1, 20(R2), which during execution reads a 32-bit word from memory and stores it in a 32-bit register R1. The effective address of the memory location is obtained by the addition of constant 20 and the contents of register R2. Which of the following best reflects the addressing mode implemented by this instruction for the operand in memory? **(GATE 2011)**

A. Immediate Addressing

B. Register Addressing

C. Register Indirect Scaled Addressing

D. Base Indexed Addressing

Answer: Here 20 will act as base and content of R2 will be index.

24. A computer handles several interrupt sources of which the following are relevant for this question.

(GATE 2011)

Interrupt from CPU temperature sensor

Interrupt from Mouse

Interrupt from Keyboard

Interrupt from Hard Disk

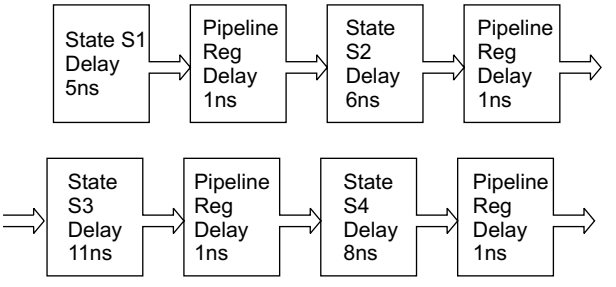
A. Interrupt from Hard Disk

B. Interrupt from Mouse

C. Interrupt from Keyboard

D. Interrupt from CPU temp sensor

25. Consider an instruction pipeline with four stages (S1, S2, S3 and S4) each with combinational circuit only. The pipeline registers are required between each stage and at the end of the last stage. Delays for the stages and for the pipeline registers are as given in the figure.



What is the approximate speed up of the pipeline in steady state under ideal conditions when compared to the corresponding non-pipeline implementation?

(GATE 2011)

A. 4.0 B. 2.5

C. 1.1 D. 3.0

In the given pipeline for every max{5,7,12,9}=12ns we will be getting a completed instruction. Where as in the case of non-pipelined execution, we will be needing 5+6+11+8=30ns to complete an instruction. Thus speed-up=30/12=2.5.

26. An 8KB direct mapped write-back cache is organized as multiple blocks, each of size 32-bytes. The proces-

sor generates 32-bit addresses. The cache controller maintains the tag information for each cache block comprising of the following.

1 Valid bit

1 Modified bit

As many bits as the minimum needed to identify the memory block mapped in the cache.

What is the total size of memory needed at the cache controller to store metadata(tags) for the cache?

(GATE 2011)

A. 4864 bits B. 6144 bits

C. 6656 bits D. 5376 bits

27. On a non-pipelined sequential processor, a program segment, which is a part of the interrupt service routine, is given to transfer 500 bytes from an I/O device to memory.

Initialise the address register

Initialise the count to 500

LOOP: Load a byte from device

Store in memory at address given by address register

Increment the address register

Decrement the count

If count != 0 go to LOOP

Assume that each statement in this program is equivalent to a machine instruction which takes one clock cycle to execute if it is a non-load/store instruction. The load-store instructions take two clock cycles to execute. The designer of the system also has an alternate approach of using the DMA controller to implement the same transfer. The DMA controller requires 20 clock cycles for initialization and other overheads. Each DMA transfer cycle takes two clock cycles to transfer one byte of data from the device to the memory.

What is the approximate speedup when the DMA controller based design is used in place of the interrupt driven program based input-output?

(GATE 2011)

A. 3.4 B. 4.4

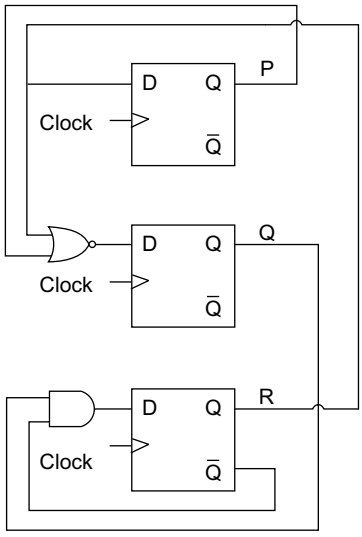
C. 5.1 D. 6.7

No. of clock cycles required by using load-store approach = $2 + 500 \times 7 = 3502$

and that of by using DMA = $20 + 500 \times 2 = 1020$

Required speed up=3502/1020 = 3.4

Consider the following circuit involving three D-type flip-flops used in a certain type of counter configuration.



28. If all the flip-flops were reset to 0 at power on, what is the total number of distinct outputs (states) represented by PQR generated by the counter?

(GATE 2011)

A. 3 B. 4

C. 5 D. 6

29. If at some instance prior to the occurrence of the clock edge, P. Q and R have a value 0, 1 and 0 respectively, what shall be the value of PQR after the clock edge?

(GATE 2011)

A. 000 B. 001

C. 010 D. 011

30. The minterm expansion of f (P, Q, R) = PQ + QR′ + PR' is **(GATE 2010)**

A. $m_2 + m_4 + m_6 + m_7$

B. $m_0 + m_1 + m_3 + m_5$

C. $m_0 + m_1 + m_6 + m_7$

D. $m_2 + m_3 + m_4 + m_5$

Answer: Each term of the given equation is represented in canonical form first and common terms are eliminated.

f(P,Q,R)=PQR + PQR′ + PQR′+ P′QR′ + PQR′+PQ′R′

f(P,Q,R)=PQR+PQR′ + P′QR′ + PQ′R′

As we want minterm expansion, the min terms are: 111, 110, 010, and 100. That is, 2, 4, 6 and 7.

31. A main memory unit with a capacity of 4 megabytes is built using 1M×1-bit DRAM chips. Each DRAM chip has 1K rows of cells with 1K cells in each row. The time taken for a single refresh operation is 100 nanoseconds. The time required to perform one refresh operation on all the cells in the memory unit is

(GATE 2010)

A. 100 nanoseconds

B. $100*2^{10}$ nanoseconds

C. $100*2^{20}$ nanoseconds

D. $3200*2^{20}$ nanoseconds

32. P is a 16-bit signed integer. The 2's complement representation of P is $(F87B)_{16}$. The 2's complement representation of 8*P is **(GATE 2010)**

A. $(C3D8)_{16}$ B. $(187B)_{16}$

C. $(F878)_{16}$ D. $(987B)_{16}$

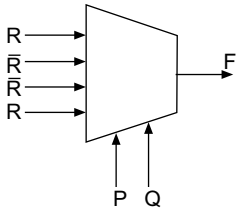
Answer: $(F87B)_{16} = 1111100001111011$

As it is 2s complement and we want to multiply, first we have to find what is that number. As MSB bit is 1, it is negative number. Therefore, to get the magnitude of the number, we apply 1s complement and add 1. The result is 0000011110000101. As we want to multiply with 8, we simply shift by 3 bits left. Thus, resulting number becomes 0011110000101000. As the number is negative, by multiplying with 8 does not change the sign. To get 2s complement of the result, we again apply 1s complement and add 1. The final result is 1100001111011000. The same in hex becomes C3D8.

33. The Boolean expression for the output f of the multiplexer shown below is **(GATE 2010)**

A. $\overline{P \oplus Q \oplus R}$ B. $P \oplus Q \oplus R$

C. $P + Q + R$ D. $\overline{P + Q + R}$



34. A system uses FIFO policy for page replacement. It has 4 page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages but now in the reverse order. How many page faults will occur? **(GATE 2010)**

A. 196 B. 192

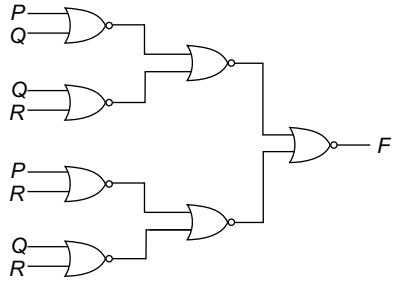
C. 197 D. 195

Answer: As the pages are distinct, during first phase all the 100 page references gives page faults. During second phase pages are referred in the reverse order. That is, for pages 100, 99, 98, and 97 we will not be having page faults as they are already available in RAM. Thus, in total 196 page faults takes place.

35. What is the Boolean expression for the output f of the combinational logic circuit of NOR gates given below?

A. $\overline{Q + R}$ B. $\overline{P - Q}$

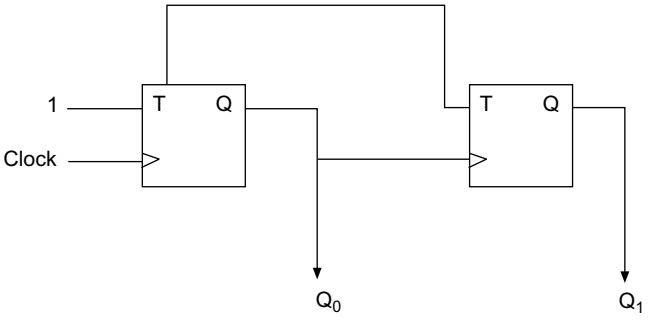
C. $\overline{P - R}$ D. $\overline{P - Q + R}$



36. In the sequential circuit shown below, if the initial value of the output Q1Q0 is 00, what are the next four values of Q1Q0? **(GATE 2010)**

A. 11, 10, 01, 00 B. 10, 11, 01, 00

C. 10, 00, 01, 11 D. 11, 10, 00, 01



37. A 5-stage pipelined processor has Instruction Fetch (IF), Instruction Decode (ID), Operand Fetch (OF), Perform Operation (PO) and Write Operand (WO) stages. The IF, ID, OF and WO stages take 1 clock cycle each for any instruction. The PO stage takes 1 clock cycle for ADD and SUB instructions, 3 clock cycles for MUL instruction, and 6 clock cycles for DIV instruction respectively. Operand forwarding is used in the pipeline. What is the number of clock cycles needed to execute the following sequence of instructions? **(GATE 2010)**

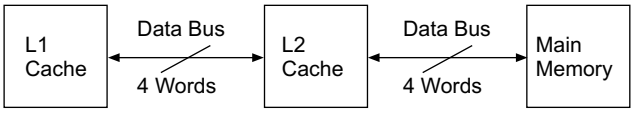
Instruction	Meaning of instruction
I_0 : MUL R_2, R_0, R_1	$R_2 \leftarrow R_0 * R_1$
I_1 : DIV R_5, R_3, R_4	$R_5 \leftarrow R_3 / R_4$
I_2 : ADD R_2, R_5, R_2	$R_2 \leftarrow R_5 + R_2$
I_3 : SUB R_5, R_2, R_6	$R_5 \leftarrow R_2 - R_6$

A. 13 B. 15

C. 17 D. 19

A computer system has an L1 cache, an L2 cache, and a main memory unit connected as shown below. The block size in L1 cache is 4 words. The block size in

L2 cache is 16 words. The memory access times are 2 nanoseconds, 20 nanoseconds and 200 nanoseconds for L1 cache, L2 cache and main memory unit respectively.



38. When there is a miss in L1 cache and a hit in L2 cache, a block is transferred from L2 cache to L1 cache. What is the time taken for this transfer? **(GATE 2010)**

A. nanoseconds B. 20 nanoseconds
C. 22 nanoseconds D. 88 nanoseconds

39. When there is a miss in both L1 cache and L2 cache, first a block is transferred from main memory to L2 cache, and then a block is transferred from L2 cache to L1 cache. What is the total time taken for these transfers? **(GATE 2010)**

A. 222 nanoseconds B. 888 nanoseconds
C. 902 nanoseconds D. 968 nanoseconds

40. What is the minimum number of gates needed to implement the Boolean function f=AB+C if we have to use only 2-input NOR gates? **(GATE 2009)**

A. 2 B. 3
C. 4 D. 5

Answer: f=AB+C= (A+C)(B+C)

(f')'=f=(((A+C)(B+C))')'= ((A+C)' + (B+C)')'

Thus, 3 NOR gates.

41. How many 32Kx1RAM chips are needed to provide a memory capacity of 256K-bytes? **(GATE 2009)**

A. 8 B. 32
C. 64 D. 128

Answer: 256K-bytes/32K × 1= 256 × 2^{10} × 8/(32 × 2^{10}) = $2^{21}/2^{15} = 2^6 = 64$

42. A CPU generally handles an interrupt by executing an interrupt service routine **(GATE 2009)**

A. As soon as interrupt is raised
B. By checking the interrupt register at the end of fetch cycle
C. By checking the interrupt register after completing the current instruction
D. By checking the interrupt register at fixed time intervals.

43. In which of the page replacement algorithms, Belody's anomaly may occurs? **(GATE 2009)**

A. FIFO B. Optimal
C. LRU D. MRU

44. The essential content(s) of each entry of a page table is/are **(GATE 2009)**

A. Virtual page number
B. Page frame number
C. Both virtual page number and page frame number
D. Access right information

45. Consider a 4 stage pipeline processor. The number of cycles needed for instructions I1, I2,I3 and I4 in stages S1, S2, S3, S4 is shown below. **(GATE 2009)**

	S1	S2	S3	S4
I1	2	1	1	1
I2	1	3	2	2
I3	2	1	1	3
I4	1	2	2	2

What is total number cycles needed to execute following loop:

for(1 to 2){I1;I2;I3;I4}

A. 16 B. 23
C. 28 D. 30

46. Consider a 4-way set associative cache (initially empty) with total 16 blocks. The main memory consists of 256 blocks and the request for memory blocks is in the following order:

0,255, 1,4,3,8, 133,159,216,129,63,8,48,32,73,92,155

Which one of the following memory block will NOT in cache if LRU replacement policy is used? **(GATE 2009)**

A. 3 B. 8
C. 129 D. 216

47. A multilevel page table is preferred in comparison to a single level page table for translating virtual address to physical address because **(GATE 2009)**

A. It reduces memory access time to read or write to a memory location
B. It helps to reduce the size of the page table needed to implement virtual address space of a process
C. It is required by the translation look aside buffer
D. It helps to reduce number of page faults in page replacement algorithms.

48. In the IEEE floating point representation the hexadecimal value 0x00000000 corresponds to **(GATE 2008)**

A. The normalized value 2^{-127}
B. The normalized value 2^{-126}

C. The normalized value +0

D. The special value +0

49. In the Karnaugh map shown below, X denotes a don't care term. What is the minimal form of the function represented by the Karnaugh map? **(GATE 2008)**

cd \ ab	00	01	11	10
00	1	1		1
01	X			
11	X			
10	1	1		X

A. $\bar{b}.\bar{d}+\bar{a}.\bar{d}$ B. $\bar{a}.\bar{b}+\bar{b}.\bar{d}+\bar{a}.\bar{b}.\bar{d}$

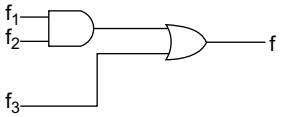
C. $\bar{b}.\bar{d}+\bar{a}.\bar{b}.\bar{d}$ D. $\bar{a}.\bar{b}+\bar{b}.\bar{d}+\bar{a}.\bar{d}$

50. Let r denote number system radix. The only value(s) of r that satisfy the equation $\sqrt{121} = 11$ is / are **(GATE 2008)**

A. decimal 10 B. decimal 11

C. decimal 10 and 11 D. any value >2

51. Given f1, f3 and f in canonical sum of products form (in decimal) for the circuit **(GATE 2008)**



$f_1 = \Sigma m\ (4, 5, 6, 7, 8)$

$f_3 = \Sigma m\ (1, 6, 15)$

$f = \Sigma m\ (1, 6, 8, 15)$

then f_2 is

A. $\Sigma m\ (4, 6)$ B. $\Sigma m\ (4, 8)$

C. $\Sigma m\ (6, 8)$ D. $\Sigma m\ (4, 6, 8)$

52. If P, Q, R are Boolean variables, then $(P+Q')(PQ'+PR)(P'R'+Q')$ simplifies to **(GATE 2008)**

A. PQ' B. PR'

C. $PQ'+R$ D. $PR'+Q$

Answer: $(P+Q')(PQ'+PR)(P'R'+Q')$

$= (PQ'+PR+PQR)(P'R'+Q') = (PQ'+PR(1+QR))(P'R'+Q') = (PQ'+PR)(P'R'+Q') = P(Q'+R)(P'+Q')(R'+Q') = PQ'$

53. Which of the following is/are true of the auto-increment addressing mode? **(GATE 2008)**

I. It is useful in creating self-relocating code

II. If it is included in an Instruction Set Architecture, then an additional ALU is required for effective address calculation

III. The amount of increment depends on the size of the data item accessed

A. I only B. II only

C. III only D. II and III only

54. Which of the following must be true for the RFE (Return from Exception) instruction on a general purpose processor? **(GATE 2008)**

I. It must be a trap instruction

II. It must be a privileged instruction

III. An exception cannot be allowed to occur during execution of an RFE instruction

A. I only B. II only

C. I and II only D. I, II and III only

55. For inclusion to hold between two cache levels L1 and L2 in a multi-level cache hierarchy, which of the following are necessary? **(GATE 2008)**

I. L1 must be a write-through cache

II. L2 must be a write-through cache

III. The associativity of L2 must be greater than that of L1

IV. The L2 cache must be at least as large as the L1 cache

A. IV only B. I and IV only

C. I, II and IV only D. I, II, III and IV

Answer:

Inclusion: If data at L1 is always a subset of data at L2 hence the L2 cache must be at least as large as the L1 cache.

In a write-through cache, every write to the cache causes a synchronous write to the backing store, hence L1 must be write-through cache and L2 need not be write-through.

56. Which of the following are NOT true in a pipelined processor? **(GATE 2008)**

I. Bypassing can handle all RAW hazards

II. Register renaming can eliminate all register carried WAR hazards

III. Control hazard penalties can be eliminated by dynamic branch prediction

A. I and II only B. I and III only

C. II and III only D. I, II and III

57. The use of multiple register windows with overlap causes a reduction in the number of memory accesses for **(GATE 2008)**

I. Function locals and parameters

II. Register saves and restores

III. Instruction fetches

A. I only B. II only

C. III only D. I, II and III

58. In an instruction execution pipeline, the earliest that the data TLB (Translation Lookaside Buffer) can be accessed is **(GATE 2008)**

A. Before effective address calculation has started
B. During effective address calculation
C. After effective address calculation has completed
D. After data cache lookup has completed

59. Which of the following statements about synchronous and asynchronous I/O is NOT true? **(GATE 2008)**

A. An ISR is invoked on completion of I/O in synchronous I/O but not in asynchronous I/O
B. In both synchronous and asynchronous I/O, an ISR (Interrupt Service Routine) is invoked after completion of the I/O
C. A process making a synchronous I/O call waits until I/O is complete, but a process making an asynchronous I/O call does not wait for completion of the I/O
D. In the case of synchronous I/O, the process waiting for the completion of I/O is woken up by the ISR that is invoked after the completion of I/O

60. A processor uses 36 bit physical addresses and 32 bit virtual addresses, with a page frame size of 4 Kbytes. Each page table entry is of size 4 bytes. A three level page table is used for virtual to physical address translation, where the virtual address is used as follows

- Bits 30-31 are used to index into the first level page table
- Bits 21-29 are used to index into the second level page table
- Bits 12-20 are used to index into the third level page table, and
- Bits 0-11 are used as offset within the page

The number of bits required for addressing the next level page table (or page frame) in the page table entry of the first, second and third level page tables are respectively **(GATE 2008)**

A. 20, 20 and 20 B. 24, 24 and 24
C. 24, 24 and 20 D. 25, 25 and 24

Consider a machine with a 2-way set associative data cache of size 64Kbytes and block size 16bytes. The cache is managed using 32 bit virtual addresses and the page size is 4Kbyts. A program to be run on this machine begins as follows:

```
double ARR [1024][ 1024] ;
int i, j ;
/* Initialize array ARR to 0.0 * /
for i 0;i 1024; i
for j 0; j 1024; j
ARR[ i][ j]= 0.0;
```

The size of double is 8Bytes. Array ARR is located in memory starting at the beginning of virtual page 0xFF000 and stored in row major order. The cache is initially empty and no pre-fetching is done. The only data memory references made by the program are those to array ARR

61. The total size of the tags in the cache directory is **(GATE 2008)**

A. 32Kbits B. 34Kbits
C. 64Kbits D. 68Kbits

62. Which of the following array elements has the same cache index as ARR[0][0]? **(GATE 2008)**

A. ARR[0][4]
B. ARR[4][0]
C. ARR[0][5]
D. ARR[5][0]

63. The cache hit ratio for this initialization loop is **(GATE 2008)**

A. 0% B. 25%
C. 50% D. 75%

64. Delayed branching can help in the handling of control hazards

For all delayed conditional branch instructions, irrespective of whether the condition evaluates to true or false **(GATE 2008)**

A. The instruction following the conditional branch instruction in memory is executed
B. The first instruction in the fall through path is executed
C. The first instruction in the taken path is executed
D. The branch takes longer to execute than any other instruction

65. The following code is to run on a pipelined processor with one branch delay slot:

I1 = ADD R2 ← R7 + R8
I2 = SUB R4 ← R5 – R6
I3 = ADD R1 ← R2 + R3
I4 = STORE Memory⌊R4⌋ ← R1
BRANCH to Label if R1 = = 0

Which of the instructions I1, I2, I3 or I4 can legitimately occupy the delay slot without any other program modification? **(GATE 2008)**

A. I1 B. I2
C. I3 D. I4

ANSWER KEY

1. B	**2.** A	**3.** B	**4.** D
5. D	**6.** C	**7.** B	**8.** C
9. B	**10.** B	**11.** C	**12.** A
13. B	**14.** C	**15.** B	**16.** B
17. C	**18.** A	**19.** B	**20.** D
21. A	**22.** B	**23.** D	**24.** D
25. B	**26.** D	**27.** A	**28.** B
29. D	**30.** A	**31.** A	**32.** A
33. B	**34.** A	**35.** A	**36.** A
37. B	**38.** C	**39.** A	**40.** B
41. C	**42.** C	**43.** B	**44.** B
45. B	**46.** D	**47.** B	**48.** D
49. A	**50.** D	**51.** C	**52.** A
53. C	**54.** B	**55.** B	**56.** C
57. A	**58.** C	**59.** B	**60.** D
61. B	**62.** B	**63.** C	**64.** A
65. B			

CHAPTER TWO 2

Programming, Data Structures and Algorithms

2.1 Programming

In this chapter we will study Programming in C, Functions, Recursion, Parameter Passing, Scope, Binding, Abstract data types and Arrays.

2.1.1 Origin of C

The C Programming Language was initially developed by Denis Ritchie using a Unix system in 1972. This was varied and modified until a standard was defined by Brian Kernighan and Dennis Ritchie in 1978 in "The C Programming Language".

Advantages/Features of C

C language has become the language of choice of more than four decades among system programmers and application programmers. Possible Reasons for its popularity can be summarised as:

1. **Powerful and flexibility:** The power and popular UNIX are written in C. The complier and interpreter for FORTAN, PASCAL, LISP, and BASIC are written in C.
2. **Portability:** C program written in one system can be run on other system with little modifications.
3. **Efficiency:** The program written in C language is highly efficient like assembly language in speed and memory management.
4. **Programmer oriented:** It has the flexible control structure and gives access to hardware and enables to manipulate individual bits of memory.
5. **Modularity:** C program can be modularising for step wise refinement. The complex program can be modularised into simple programs.

2.1.1.1 C Program Structure

C program is traditionally arranged in the following order but not strictly as a rule.

Function prototypes and global data declarations
The main() function
Function definitions

Header files contain function *prototypes of* the standard library functions and declarations for the various variables or constants needed. The line

#include <stdio.h>

instructs the pre-processor to include the file stdio.h into the program before compilation so that function declarations of standard input/output functions are included in our C program. The angle braces denote that the compiler should look in the default "INCLUDE" directory for this file. A pair of double quotes indicate that the compiler should search in the specified path, e.g.

#include "d:\\myfile.h"

The main() function is also called as driver or entry point. Program execution starts from main.

2.1.1.2 A Note on Programming Languages

Programming languages are classified as shown in Table 2.1.

Table 2.1 Classification of Programming Languages

Programming Languages	Low Level Languages	Ex: - Machine Language, Assembly Language.
	Medium Level Languages	Ex: - C
	High Level Languages	Ex: - C, C++, Java, Fortran, COBAL, BASIC.

High Level Languages (HLL) uniquely identified through their rich grammar and structure with which we can specify what we wanted in a straightforward manner, where as low-level languages will not be having high level grammar.

High Level Languages statements are self-complete. For example, consider a statement in C language **A = A – B**. This is understandable for any one even though they do not have skill in C language. Everyone can understand that B value is subtracted from A and result is stored in A. Now, consider an equivalent assembly statement **SUB A,B.** No doubt, here also the action is same as above. However, it raises many doubts; Whether A has to be subtracted from B or B has to be subtracted from A. Where to store the results? In A or B? That is, this type of instructions are not straightforward or self complete like HLL instructions.

Low level languages will be having freedom to access the parts of the machine such as registers, etc. directly. Also, a low level program takes less resources (CPU time, memory) compared to equivalent HLL.

No two processors will be having the same machine language instruction sets or "Assembly Language" instruction sets. Thus, programs written using a processors "Machine Language (OR) Assembly Language" will work only on that processor. That is, Machine Level Language Programs (OR) Assembly Language Programs are not portable.

HLL Programs can work on variety of machines without many modifications. Thus, they are portable and thus their value will be more. Medium level language is the one which contains both the flavours.

According to Software Engineering, a programming language should have the following characteristics:

1. program should be easily readable
2. program should be easily understandable
3. program should be easily repairable
4. program should be easily upgradable
5. program should be concise(short)
6. program should be easily portable

2.1.1.3 What are Translators?

In general, translators convert programs written in one language to another language. For example, we have the following types of translators:

- **Assemblers**
- **Compilers**
- **Converters**
- **Interpreters**

An assembler takes "Assembly Language Programs" as input and converts them into machine language. Usually assemblers are specific to processors. That is, universal assemblers are not available as of now. For example, for Intel family of processors the following assemblers are available:

MASM	Microsoft Assembler
TASM	Turbo Assembler
NASM	GNU Assembler

Compilers take High Level Language Program as input and convert it into machine language. Of course, we do not have universal compilers which take any "High Level Language Program" and give machine language program which run's on any machine. Instead, for one HLL language alone we may find hundreds of compilers. For example:

Names of C – Language Compilers	
DOS / Windows	**Unix**
Microsoft 'C'	Sun 'C'
Turbo 'C'	Solaris 'C'
Borland 'C'	MIPS 'C'
Zortex 'C'	GNU 'C'
Speed 'C'	
PCC 'C'	
DJGPP 'C'	

Of course, every language will have its standards. For C language, we have "American National Standard Institute (ANSI) C" standards. No doubt, it is possible that more than one person/company will be selling their C - Language Compiler. All of them supposed to follow ANSI C standards. However, these vendors supply their compilers along with some readymade programs or libraries to attract the customers. If we develop our programs using these readymade programs, then our programs lacks portability. Thus, we have to worry about portability of the code also.

Converters, convert programs in one high level language to another high-level language. For example, Pascal to 'C' and FORTRAN to 'C', etc.,

High-level language programs are converted into machine language only if they are free from language wise mistakes (which are also called as syntactical errors or compile time errors). Of Course, unless machine language programs are available, we cannot run the program.

Interpreter's takes High-level language programs instructions one after another and executes immediately. Usually Interpreter's are needed to identify run time errors, which are also called as bugs.

Neither compilers nor assemblers cheek the logical mistakes in our programs. Logic is the sole responsibility of the programmer.

2.1.1.4 Errors Possible during Programming

Saliently, errors during programming are classified as:

- **Compile Time Errors (syntactical errors or language wise errors)**
- **Run Time Errors (Bugs)**
- **Computational Errors**

Compile time errors occur when we violate language rules while writing our programs. For example, an executable statement in C language has to be ended with a semicolon. If we violate this in our program, we get an error which we can call as compile time error or syntactical error. Unless, we correct all the language wise errors in our program, it will not be converted to machine language. Unless, machine language file is available, we cannot run our program. These errors are reported with respect to our source program itself. Thus, it is relatively easy to identify and correct them.

Those errors which occur during the execution of a program are called as run-time errors or bugs. Run time errors may occur because of many reasons such as:

(1) If our programs violate the rules and regulations of the operating system. For example, if our program tries to access memory which is not allocated to us, we may get an error such as "Memory Segment Violation".

(2) The errors depend on data and program.

(3) Because of HW faults also, we may get run-time errors.

Consider the calculation of square root of the discriminant function of a quadratic equation $ax^2+bx+c=0$.

Probable steps in the program can be:

1. Read coefficients a, b and c.
2. Calculate dis = sqrt(b*b–4*a*c)
3. Print dis.

Now, assume that we have entered values "1 4 1". The program may give us 3.464. Let us assume that we have entered values "4 1 1". This time, we may get an error such as "Domain Error" as we are trying to calculate square root of a negative number. This type of error is run-time error and it is depending on the data and problem.

Because, we may not get same error with any program with same input "4 1 1".

Usually, run-time errors are reported with respect to machine language file. Thus, it is very difficult to understand and correct. Thus, when a program is giving run time errors, we will run interpreters to find the reason. The process of identifying or finding the run-time errors and fixing them is called as de-bugging or bug fixing.

Computational errors are the ones, which will distort the final value of the program. These errors are may occur because of many reasons including the finite behaviour of the computers such as finite word size, etc.,. Also because of mathematical equations, and operations on them may lead to computational error.

For example, if x = 10/3 = 3.3333…. y = x*3

According to mathematics, y = 10, but in computers, y = 9.999999.

Note

Most of the computers give real valued results accurately until 6th decimal point.

In some applications such as astronomy, biology requires high precision. Mistakes in the 31st decimal point may also lead to catastrophic results. Thus, we have to give proper attention to computational errors also.

Syntax and Semantics

Syntax is the grammar of a language: the rule governing the structure and contents of semantics. In other words, it is a rule that tells us whether program written using programming languages are correct or not. **Semantics** means the relationship between words or symbols and their intended meaning in programming. Programming languages are subject to certain semantic rules thus, program statements can be syntactically correct but semantically incorrect.

Difference between Syntax and Semantics

	Syntax		**Semantics**
(1)	It is grammatical rule, governed by the programming language	(1)	It is the meaning of a grammatical rule of a programming language
(2)	Difficult to learn and understand	(2)	Easy to learn and understand
(3)	It has got only one approach to perform a job	(3)	It has many way to complete the syntax
(4)	Errors are created if rules violated	(4)	Errors are created when sentences or statements doesn't form properly

2.1.1.5 Program Development Stages

A typical program development involves:

1. Preparing Algorithm
2. Preparing Flow Chart
3. Coding
4. Compiling and Commissioning

Programming Techniques (paradigms)
There are various programming techniques such as:

- Structural programming
- Modular design
- Top-down designing
- Bottom up designing
- Object oriented programming

Structural programming
Structural programming can be seen as subset of procedural programming, one of the major techniques for programming. Procedural programming is based on the idea of modularity, where the entire program is divided into modules. Again each module is composed of one or more procedures also called as functions or subroutines. E.g.: C
Features

- Emphasis is given on procedure
- Programs are divided into different functions
- Top down method is used in programs

Advantages

- Complexity of the program is reduced
- Easy maintenance
- Location of error is possible
- Debugging time decreases

Disadvantages

- More memory space is required. When the numbers of modules are out of certain range, performance of program is not satisfactory.

Modular design
In modular approach, large program is divided into many small discrete components called –modules. For example: Pascal procedure or function C, C++ function. Modules are debugged and tested separately and are combined to build system.
Top down
In this approach, designers convert the large structure of a program in terms of smaller operations, implement and test the smaller operations, and then tie them together into a whole program.

Bottom-Up designing
In this approach software is constructed from small piece of code to whole system. The first step is to identify individual subtask which can be used as building clock to compose the overall task tools and libraries.

Object oriented programming
Object-oriented programming gives greater flexibility and easy maintenance in programming, and is thus widely popular in large-scale software systems development.
Class : The unit of definition of data and behavior (functionality) for some kind-of-thing. A class is the basis of modularity and structure in an object-oriented computer program. For example, the 'class of Dogs' might be a set which includes the various breeds of dogs. Consider a stencil which is a thin sheet of material, such as paper, plastic, or metal, with letters or a design cut from it, used to produce the letters or design on an underlying surface by applying pigment (color) through the cut-out holes in the stencil. If we apply red color, we get matter with red color; if we apply blue color, we get the same matter with blue color. We can consider this stencil as a class while the text in red or text in blue as objects.
Object: An instance of a class, an object is the run-time manifestation (*instantiation*) of a *particular exemplar* of a class. Each object has its own data, though the code within a class.
Method (also known as *message*) : How code can use an object of some class. A *method* is a form of subroutine operating on a single object.
Inheritance: A mechanism for creating subclasses, inheritance provides a way to define a (sub) class. A subclass *inherits* all the members of its superclass(es), but it can extend their behavior and add new members.
Abstraction: Combining multiple smaller operations into single unit that can be referred to by name.
Encapsulation: Separating implementation from interfaces. User only needs to "What objects does" rather than "how the objects does".

Polymorphism: Using the same name to do different operations on objects of different data types.

Reusability: Once the objects are created, they can be reused again and again.

2.1.1.6 Introduction to C Compiling: A Unix/Linux example

The easiest case of compilation is when we have all our source code set in a single file. Let us assume there is a file named 'x.c' that we want to compile. We will do so using a command similar to:

cc x.c (In most of the flavors of UNIX's)

gcc x.c (In Gnu C compiler)

acc x.c (In Solaris)

Every compiler might show its messages (errors, warnings, etc.) differently, but in all cases, we will get a file 'a.out' as a result, if the compilation completed successfully. In the case of Windows systems, if our file is xyz.c then the resultant executable file name will be xyz.exe. It is true with Turbo C and Dev-C++ compilers also.

C program file compilation in general is split into roughly 5 stages (as shown in Fig. 2.1): Preprocessing, Parsing, Translation, Assembling, and Linking. With most of the compilers, similar stages are seen.

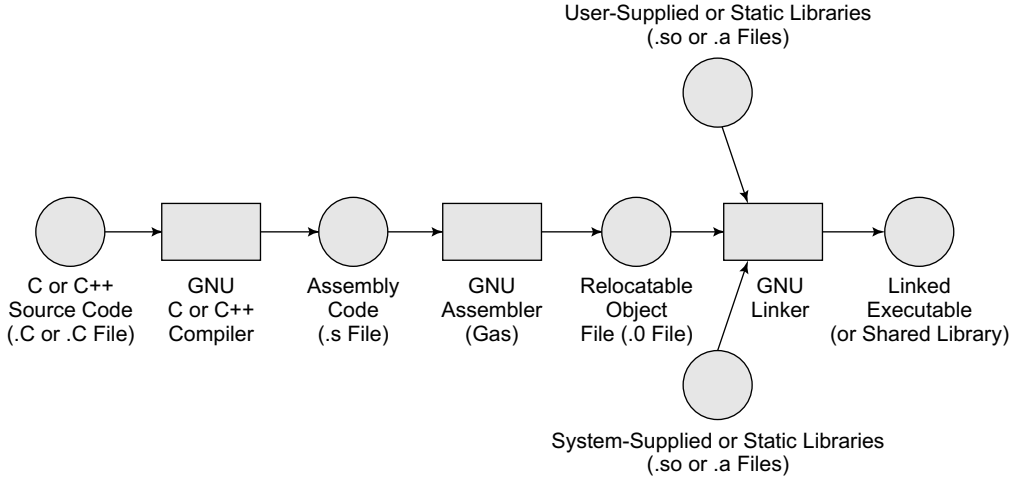


Figure 2.1 Stages in C Program Compilation using GNU C compiler

Understanding the Compilation Steps

Now that we have learned that compilation is not just a simple process, let us try to see what is the complete list of steps taken by the compiler in order to compile a C program.

Driver: We can invoke it by typing "cc" or "gcc". This is actually the "engine", that drives the whole set of tools the compiler is made of. We invoke it, and it begins to invoke the other tools one by one, passing the output of each tool as an input to the next tool.

C Pre-Processor: Normally called "cpp". It takes a C source file, and handles all the pre-processor definitions (#include files, #define macros, conditional source code inclusion with #ifdef, etc.)

The C Compiler: Normally called "cc1". This is the actual compiler, which translates the input file into assembly language.

Optimiser: Sometimes comes as a separate module and sometimes found inside the compiler module. This one handles the optimisation on a representation of the code that is language-neutral. This way, we can use the same optimiser for compilers of different programming languages.

Assembler: Sometimes called "as". This takes the assembly code generated by the compiler, and translates it into machine language code kept in object files.

Linker-Loader: This is the tool that takes all the object files (and C libraries), and links them together, to form one executable file, in a format the operating system supports. A common format these days is known as "ELF". On SunOS systems, and other older systems, a format named "a.out" was used. This format defines the internal structure of the executable file - location of data segment, location of source code segment, location of debug information and so on.

Suppose that we want the resulting program to be called with another name other than "a.out", then we can use the following line to compile it:

cc –o executable_filename x.c

gcc –o executable_filename x.c

Running the Resulting Program

Once we created the program, we wish to run it. This is usually done by simply typing its name at the command prompt.

executable_filename

However, this requires that the current directory be in our PATH (which is a variable telling our UNIX shell where to look for programs we are trying to run). In many cases, this directory is not placed in our PATH. Thus in order to run our program we can try:

./executable_filename

This time we explicitly told our UNIX shell that we want to run the program which is in the current directory.

2.1.1.7 Compilation Steps under Dos/Windows

No doubt the compilation process under Windows/DOS plat forms also works in almost same fashion. However, resultant file names may change. For instance, final executable file name will be having an extension of EXE.

2.1.2 Variables, Data Types, I/O and Operators

In order to store data, variables are used in programming languages. However, based on what we wanted to store in the variables, we associate different types to variables. For example, an integer can represent the age of a person reasonably. So, the variable in which we would like to store age will be declared as integer type.

2.1.2.1 Basic Data Types

In C language, there are five basic data types **char, int, float, double, and void**. Note that the size of an int depends on the operating system. The following table contains various data types and their memory sizes.

char	1 byte (8 bits) with range -128 to 127
int	16-bit OS : 2 bytes with range -32768 to 32767 32-bit OS : 4 bytes with range -2,147,483,648 to 2,147,483,647
float	4 bytes with range 10-38 to 1038 with 7 digits of precision
double	8 bytes with range 10-308 to 10308 with 15 digits of precision
void	generic pointer, used to indicate no function parameters etc.

Modifying Basic Types

Except for type ***void*** type, the meaning of the above basic types can be altered by combining with the following keywords.

signed
unsigned
long
short

The *signed* and *unsigned* modifiers may be applied to types char and int and will simply change the range of possible values. For example, an *unsigned char* has a range of 0 to 255, all positive, as opposed to a signed char which has a range of -128 to 127. An *unsigned integer* on a 16-bit system has a range of 0 to 65535 as opposed to a *signed int* which has a range of -32768 to 32767. Note however that the default for type *int* or *char* is signed so that the type *signed char* is always equivalent to type *char* and the type *signed int* is always equivalent to *int*.

The *long* modifier may be applied to type int and double only. A *long int* will require 4 bytes of storage no matter what operating system is in use and has a range of -2,147,483,648 to 2,147,483,647. A *long double* will require 10 bytes of storage and will be able to maintain up to 19 digits of precision.

The *short* modifier may be applied only to type int and will give a 2 byte integer independent of the operating system in use.

Note that the keyword *int* may be omitted without error so that the type *unsigned* is the same as type *unsigned int*, the type *long* is equivalent to the type *long int*, and the type *short* is equivalent to the type *short int*.

2.1.2.2 Variables

A variable is a named piece of memory which is used to hold a value which may be modified by the program. A variable thus has three attributes that are of interest to us: its **type,** its **value** and its **address.** The variable's type informs us what type and range of values it can represent (or store) and how much memory is used to store that value. The variable's address informs us where in memory the variable is located. C variables are declared as:

type variable-list;

For example:

```
        int i;
 char a, b, ch;
```

Variables are declared in three general areas in a C program. When declared inside functions as follows they are termed **local** variables and are visible (or accessible) within the function (or code block) only.

```
void main(){
int i, j ;
}
```

A local variable is created i.e., allocated memory for storage upon entry into the code block in which it is declared and is destroyed i.e., its memory is released on exit. This means that values cannot be stored in these variables for use in any subsequent calls to the function.

When declared outside functions they are termed **global** variables and are visible throughout the file or have file scope. These variables are created at program start-up and can be used for the lifetime of the program.

```
int i ;
void main(){
}
```

When declared within the braces of a function they are termed the formal parameters of the function which is explained later. However, the variables declared inside function (as shown), are again automatic type.

```
int func1( int a, char b ) {
 int x, y;
}
```

Variable Names

Names of variables and functions in C are called identifiers and are case sensitive. The first character of an identifier must be either a letter or an underscore while the remaining characters may be letters, numbers, or underscores. Identifiers in C can be up to 31 characters in length.

Identifiers and Keywords

An identifier can be defined as the name of the variable, function, arrays, structures, constants etc. are created by the programmer. They are the fundamental requirements of any programming language.

Keywords

Keywords are reserved identifiers and they cannot be used as names for the program variables. The keywords are also called as reserved words. The meaning of the keywords are already given to the compiler. There are 32 keywords available in C:

auto	**cons**	**double**	**float**	**int**
short	**struct**	**unsigned**	**break**	**continue**
else	**for**	**long**	**signed**	**switch**
void	**case**	**default**	**enum**	**goto**
register	**sizeof**	**typedef**	**volatile**	**char**
do	**extern**	**if**	**return**	**static**
union	**while**			

Delimiters

Delimiters are used for syntactic meaning in C. These are as follows:

:	colon	used for label
;	semicolon	end of statements
( )	parenthesis	used in expression
{}	curly braces	used for block of statements
[]	square bracket	used for array
#	hash	preprocessor directives
,	comma	variables delimiter

Initialising Variables

When variables are declared in a program it just means that an appropriate amount of memory is allocated to them for their exclusive use. This memory however is **not initialised** to zero or to any other value automatically and so will contain random values unless specifically initialised before use. That is, soon after declaration of a variable, it is said to be un-initialised or it is said to be having garbage value or trash value. Never, one is supposed to use un-initialised variables to the right hand side of an equality or expression.

type var-name = constant ;

For example:

```
char ch = 'N' ;
double d = 12.2323 ;
int i, j = 20 ;/*note in this case only j is initialised */
```

2.1.2.3 Storage Classes

There are four storage class modifiers used in C which determine an identifier's storage duration and scope.

auto
static
register
extern

An identifier's storage duration is the period during which that identifier exists in memory. Some identifiers exist for a short time only, some are repeatedly created and destroyed and some exist for the entire duration of the program. An identifier's scope specifies what sections of code it is accessible from.

The auto storage class is implicitly the default storage class used and simply specifies a normal local variable which is visible within its own code block only and which is created and destroyed automatically upon entry and exit respectively from the code block.

The register storage class also specifies a normal local variable but it also requests that the compiler store a variable so that it may be accessed as quickly as possible, possibly from a CPU register.

The static storage class causes a local variable to become permanent within its own code block i.e. it retains its memory space and hence its value between function calls.

When applied to global variables the static modifier causes them to be visible only within the physical source file that contains them i.e., to have file scope. Whereas the extern modifier which is the implicit default for global variables enables them to be accessed in more than one source file.

For example in the case where there are two C source code files to be compiled together to give one executable and where one specific global variable needs to be used by both then extern class allows the programmer to inform the compiler of the existence of this global variable in both files.

2.1.2.4 Constants

Constants are fixed values that cannot be altered by the program and can be numbers, characters or strings. For example, the following lines contains various types of constants:

char : 'M', '$', '9'

int : 19, 190, -1900

unsigned : 0, 255

float : 12.123456, -1.573e10, 1.347654E-13

double : 433.34534545454, 1.3556456456456E-200

long : 65536, 2222222

string : "Hello Welcome to C Jungle\n"

Floating point constants default to type double. For example the following code segment will cause the some compilers to issue a warning pertaining to floating point conversion in the case of fval but not in the case of dval.

```
float fval ;
double dval ;
fval = 123.345 ;
dval = 123.345 ;
```

However, the value may be coerced to type float by the use of a modifier as follows:

```
f = 123.345F ;
```

Integer constants may also be forced to be a certain type as follows:

1080U --- unsigned

1800L --- long

Integer constants may be represented as either decimal which is the default, as hexadecimal when preceded by "0x", e.g. 0x2A, or as octal when preceded by "O", e.g. O27.

Character constants are normally represented between single quotes, e.g. 'a', 'b', etc. However, they may also be represented using their ASCII (or decimal) values e.g. 97 is the ASCII value for the letter 'a', and so the following two statements are equivalent.

```
char ch = 97 ;
char ch = 'a' ;
```

There are also a number of special character constants sometimes called *Escape Sequences*, which are preceded by the backslash character '\', and have special meanings in C. Table 2.2 illustrates the same.

Table 2.2

\n	newline
\t	tab
\b	backspace
\'	single quote
\"	double quote
\0	null character
\a	beep sound
\xdd	represent as hexadecimal constant

2.1.2.5 Console Input / Output or Interactive Input/output

This section introduces some of the more common input and output functions provided in the C standard library.

printf()

The printf() function is used for formatted output and uses a control string which is made up of a series of format specifiers to govern how it prints out the values of the variables or constants required. The more common format specifiers are as follows :

%c character	%f floating point
%d signed integer	%lf double floating point
%i signed integer	%e exponential notation
%u unsigned integer	%s string
%ld signed long	%x unsigned hexadecimal
%lu unsigned long	%o unsigned octal
	%% prints a % sign

Field Width Specifiers

Field width specifiers are used in the control string to format the numbers or characters output appropriately.

%[total width printed][.decimal places printed]format specifier

Where square braces indicate optional arguments.

There are also a number of flags that can be used in conjunction with field width specifiers to modify the output format. These are placed directly after the % sign. A – (minus sign) causes the output to be left-justified within the specified field, a + (plus sign) displays a plus sign preceding positive values and a minus preceding negative values, and a 0 (zero) causes a field to be padded using zeros rather than space characters.

The printf() function returns how many characters it has printed. For example, the following code fragment gives output in two lines, in first line 19289 and in the second line 6. Though, n variable takes 2 or 4 bytes in memory, while printing it will be converted to ASCII digits (6 digits). In the innermost printf(), we are printing 5 digits plus newline. Thus, the innermost printf() returns 6, which outer printf() prints.

```
int n=19289;
printf("%d\n", printf("%d\n",n) );
```

Similarly, the following code will display 26 as the last printf() output.

```
printf("%d\n", print("Hello Welcome To C Jungle\n") );
```

Similarly, the following code will display 12 as the last printf() output.

```
float n=-1.23338;
printf("%d\n", printf("%20f8\n",n) );
```

scanf()

This function is similar to the printf function except that it is used for formatted input. The format specifiers have the same meaning as for printf() and the space character or TAB character or the newline character are normally used as delimiters between different numerical inputs.

The and character is the *address of* operator in C, it returns the address in memory of the variable it acts on. (This is because C functions are nominally call--by--value. Thus in order to change the value of a calling parameter we must tell the function exactly where the variable resides in memory and so allow the function to alter it directly rather than to use-lessly alter a copy of it).

The scanf function has a return value which represents the number of fields it was able to convert successfully. For example:

```
num = scanf( "%c %d", &ch, &i );
```

This scanf call requires two fields, a character and an integer, to be read in so the value placed in *num* after the call should be 2 if this was successful. However if the input was "a Rambo" then the first character field will be read correctly as 'a' but the integer field will not be converted correctly as the function cannot reconcile "Rambo" as an integer. Thus the function will return 1 indicating that one field was successfully converted. Thus to be safe the return value of the scanf function should be checked always and some appropriate action taken if the value is incorrect.

The following program gives a run-time error as we are trying to store the value that is read from the keyboard in the memory location which is not allocated. The scanf function needs addresses as its arguments after first argument. Here, we have declared variable x. As we are simply using x in scanf() function, and x is not initialised, it assumes the value entered by the user to be stored in the memory pointed by the value of x which is garbage. Thus, we get run time error.

```
int x;
printf("%d\n",scanf("%d", x));
```

getchar() and putchar()

These functions are used to input and output single characters. The *getchar()* function reads the ASCII value of a character input at the keyboard and displays the character while *putchar()* displays a character on the standard output device i.e., the screen.

The input functions described above, scanf() and getchar() are termed buffered input functions. This means that whatever the user types at the keyboard is first stored in a data buffer and is not actually read into the program until either the buffer fills up and has to be flushed or until the user flushes the buffer by hitting **ret** whereupon the required data is read into the program. The important thing to remember with buffered input is that no matter how much data is taken

into the buffer when it is flushed the program just reads as much data as it needs from the start of the buffer allowing whatever else that may be in the buffer to be discarded.

_flushall()

The _flushall function writes the contents of all output buffers to the screen and clears the contents of all input buffers. The next input operation (if there is one) then reads new data from the input device into the buffers. This function should be used always in conjunction with the buffered input functions to clear out unwanted characters from the buffer after each input call.

getch() and getche()

These functions perform the same operation as getchar() except that they are unbuffered input functions i.e., it is not necessary to type **ret** to cause the values to be read into the program they are read in immediately the key is pressed. getche() echoes the character hit to the screen while getch() does not.

2.1.2.6 Operators

Assignment Operator

```
int x ;
x = 20 ;
```

Some common notation:

lvalue -- left hand side of an assignment operation

rvalue -- right hand side of an assignment operation

Type Conversions: The value of the right hand side of an assignment is converted to the type of the lvalue. This may sometimes yield compiler warnings if information is lost in the conversion. For example:

```
int x ;
char ch ;
float f ;
ch = x ;  /* ch is assigned lower 8 bits of x, the remaining bits are discarded so we have a possible
information loss  */
x = f ;   /* x is assigned non fractional part of f only within int range, information loss possible  */
f = x ;   /* value of x is converted to floating point */
```

Multiple assignments are possible to any degree in C, the assignment operator has right to left associatavity which means that the rightmost expression is evaluated first. For example:

x = y = z = 1000 ;

In this case the expression z = 1000 is carried out first. This causes the value 1000 to be placed in z with the value of the whole expression being 1000 also. This expression value is then taken and assigned by the next assignment operator on the left i.e. x = y = (z = 1000).

Arithmetic Operators

+ – * / --- same rules as mathematics with * and / being evaluated before + and -.

% -- modulus / remainder operator

For Example:

```
int a = 5, b = 2, x ;
float c = 5.0, d = 2.0, f ;
x = a / b ; //  integer division, x = 2.
f = c / d ; //  floating point division, f = 2.5.
x = 5 % 2 ; //  remainder operator, x = 1.
x = 7 + 3 * 6 / 2 - 1 ;
// x=15,* and / evaluated ahead of + and -.
```

Note that parentheses may be used to clarify or modify the evaluation of expressions of any type in C in the same way as in normal arithmetic.

Modulus operator (%) or Remainder operator: It is a binary operator i.e., it requires two operands. The operands should be integer type; they can be integer variables, constants or expressions (not float type arguments).

If A, B are two operands then A%B is remainder of |A| / |B| with the sign as that of first operand.

Example:

10%3 = 1 −10%3 = -1 10%-3 = 1 −10%-3 = −1

If **a, b** are two integers and **a%b** value is zero then **b** can be said as factor to **a**.
If **a, b** are two integers then **a%b** values lies between **0 to (b−1)**.
If **a** is an integer and **a%2** value is **zero** then a can be said as even number; otherwise a can be said as odd number.
If **a, b** are two integers and **a** is less than **b** then **a%b** value is **a**.
If **a** is an integer and then **a%1** value is **0**.

The Comma Operator

The comma operator can be used to link related expressions together. A comma separated list of expression is evaluated left to right and value of right most expression is the value of the combined expression.

For example the statement

value = (x = 10, y = 9, x + y);

First assigns 10 to x and 9 to y and finally assigns 19 to value. Since comma has the lowest precedence in operators the parenthesis is necessary.

2.1.2.7 Unary Operators

C language supports variety of unary operators (which takes one operand) such as:

(1) Unary minus
(2) Negation operator(!)
(3) Unary increment/decrement

We need not required to give emphasis about unary minus as its semantics is same as mathematics.

Negation operator is a unary operator, whose operand can be a variable or constant of any type. If the operand is true negation of it becomes false; if the operand is false then the negation of it becomes true.

Increment and Decrement Operators

There are two special unary operators in C, Increment ++, and Decrement -- , which cause the variable they act on to be incremented or decremented by 1, respectively.

For example: x++ ;
/* equivalent to x = x + 1 ; */

++ and -- can be used in prefix or postfix notation. In prefix notation the value of the variable is either incremented or decremented and is then read while in postfix notation the value of the variable is read first and is then incremented or decremented.

Operators	Postfix	Prefix
Increment	i++	++i
Decrement	i--	--i

Unary Increment / Decrement operators are applicable to integer type of variables only (which includes int, char, long, etc.,); but not to constants or expressions. Postfix Increment/Decrement operator if applied to a variable in an expression then the current value of the variable is used in evaluating the expression and then the variable value will be Incremented /Decremented by one. Similarly Prefix Increment/Decrement operator when applied to a variable in an expression, first the value of the variable is Incremented or Decremented by one and then the modified value of the variable is used in evaluating the expression.

In an expression, a variable contains m prefix and n postfix operators then before evaluating the expression the variable value will be modified by m times, with the modified value expression will be evaluated after that the variable will be further modified by n times.

Implicit Assignment Operators

Many C operators can be combined with the assignment operator as shorthand notation. For example:

x = x + 10 ; can be replaced by
x += 10 ;

Similarly, we can use -=, *=, /=, %=, etc.

These shorthand operators improve the speed of execution as they require the expression, the variable x in the above example, to be evaluated once rather than twice.

Relational Operators

The full set of relational operators are provided in shorthand notation

> >= < <= == !=

Result of a relational operator will be true or false (numerically 1 or 0 respectively). They can be used with all type of variables and constants including int, float, long, char.

Logical Operators

A	B	A&&B	A||B
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

C language supports two logical operators given as:

1. Logical AND (&&)
2. Logical OR (||)

Logical And or logical Or operators are binary operators, i.e., two operands are required for them. The operands can be variables, constants or expressions of any type. The result of "Logical AND", "Logical OR" operators is always true or false; numerically one or zero, respectively.

The result of the logical AND will be true only if both operands are true otherwise false.

Where as the result of "Logical OR" is true only if at least one of the operand is true. See the above truth table.

While evaluating logical AND operator if the first operand itself is identified to be false, then the second operand will not be evaluated; even if it is true, final result will be false. Similarly, while evaluating the logical OR operator, if the first operand itself is identified to be true then the second operand will not be evaluated at all.

With out loosing the logical significance, one can use *, + in place of logical AND, logical OR. However, it does not mean that in place of *, +, we can use Logical AND, Logical OR in arithmetical statements without distorting the value. Logical AND, Logical OR will be faster than arithmetic operators.

Bitwise Operators

These are special operators that act on *char* or *int* arguments only. They allow the programmer to get closer to the machine level by operating at bit-level in their arguments.

&	Bitwise AND	|	Bitwise OR
^	Bitwise XOR	~	Ones Complement
>>	Shift Right	<<	Shift left

Recall that type char is one byte in size. This means it is made up of 8 distinct bits or binary digits normally designated as illustrated below with Bit 0 being the Least Significant Bit (LSB) and Bit 7 being the Most Significant Bit (MSB). The value represented below is 13 in decimal.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	1	0	1

An integer on a 16 bit OS is two bytes in size and so Bit 15 will be the MSB while on a 32 bit system the integer is four bytes in size with Bit 31 as the MSB.

Bitwise AND (&)

If any two bits in the same bit position are set then the resultant bit in that position is set otherwise it is zero. For example:

```
    1011 0010   (178)
&   0011 1111   (63)
=   0011 0010   (50)
```

Bitwise OR (|)

If either bit in corresponding positions are set the resultant bit in that position is set. For example:

```
    1011 0010   (178)
|   0000 1000   (63)
=   1011 1010   (186)
```

Bitwise XOR (^)

If the bits in corresponding positions are different then the resultant bit is set. For example:

```
        1011 0010    (178)
^       0011 1100    (63)
=       1000 1110    (142)
```

Shift Operators (<< and >>)

These move all bits in the operand left or right by a specified number of places.

variable << number of places

variable >> number of places

For example:

2 << 2 = 8

i.e.

0000 0010 becomes 0000 1000

Note : shift left by one place multiplies by 2

shift right by one place divides by 2

Ones Complement (~)

Complement operator reverses the state of each bit of the operand. For example:

1101 0011 becomes 0010 1100

The bitwise operators are most commonly used in system level programming where individual bits of an integer will represent certain real life entities which are either on or off, one or zero. The programmer will need to be able to manipulate individual bits directly in these situations.

A *mask* variable which allows us to ignore certain bit positions and concentrate the operation only on those of specific interest to us is almost always used in these situations. The value given to the mask variable depends on the operator being used and the result required.

Implicit and Explicit Type Conversions

In mixed type expressions all operands are converted temporarily up to the type of the largest operand in the expression. Normally, this automatic or implicit casting of operands follows the following guidelines in ascending order.

long double
double
float
unsigned long
long
unsigned int
signed int

For Example:

```
int i ;
float f1, f2 ;
f1 = f2 + i ;
```

Since f2 is a floating point variable the value contained in the integer variable is temporarily converted or ***cast to*** a floating point variable also to standardise the addition operation in this case. However it is important to realise that no permanent modification is made to the integer variable.

Explicit casting coerces the expression to be of specific type and is carried out by means of the **cast operator** which has the following syntax.

(type) expression

For example if we have an integer x, and we wish to use floating point division in the expression x/2 we might do the following

(float) x / 2

which causes x to be temporarily cast to a floating point value and then implicit casting causes the whole operation to be floating point division. The same results could be achieved by stating the operation as

x / 2.0

which essentially does the same thing but the former is more obvious and descriptive of what is happening.

It should be noted that all of these casting operations, both implicit and explicit, require processor time. Therefore for optimum efficiency the number of conversions should be kept to a minimum.

Size of Operator

The sizeof operator gives the amount of storage, in bytes, associated with a variable or a type (including aggregate types as we will see later on). The expression is either an identifier or a type-cast expression (a type specifier enclosed in parentheses).

Precedence of Operators

When several operations are combined into one C expression the compiler has to rely on a strict set of precedence rules to decide which operation will take preference. The precedence of C operators is given below.

Precedence	Operator	Associativity
Highest	( ) [ ] –> .	left to right
	! ~ ++ -- +(unary) –(unary) (type) * & sizeof	right to left
	* / %	left to right
	+ –	left to right
	<< >>	left to right
	< <= > >=	left to right
	== !=	left to right
	&	left to right
	^	left to right
	|	left to right
	&&	left to right
	||	left to right
	? :	right to left
	= += –= *= /= %= &= ^= |= <<= >>=	right to left
Lowest	,	left to right

Operators at the top of the table have highest precedence and when combined with other operators at the same expression level will be evaluated first. For example take the expression

2 + 10 * 5 ;

Here * and + are being applied at the same level in the expression but which comes first ? The answer lies in the precedence table where the * is at a higher level than the + and so will be applied first.

When two operators with the same precedence level are applied at the same expression level the associativity of the operators comes into play.

For example in the expression

2 + 3 – 4 ;

the + and - operators are at the same precedence level but associate from left to right and so the addition will be performed first. However in the expression

x = y = 2 ;

as we have noted already the assignment operator associates from right to left and so the rightmost assignment is first performed.

Note

As we have seen already parentheses can be used to supersede the precedence rules and force evaluation along the lines we require. For example to force the addition in **2 + 10 * 5 ;** to be carried out first we would write it as **(2 + 10) * 5;**

■ **Example** What is the value of z after execution of the following statements.

int z, x = 5,y = -10, a = 4, b = 2;

z = x++ - --y * b /a;

■ **Answer:** Note that as x contains postfix operator, its value will be changed only after evaluating the expression. That is, expression will be evaluated with the current value of x that is 5. First y is incremented by 1. Thus it becomes -11. Then y will be multiplied by b. Thus, we get -22. This 22 will be divided by 4. Thus we get -5. This is subtracted from x value that is 5. Thus, expression value becomes 10 and after that x values becomes 6. Thus z value becomes 10.

```
z= (5 - (((-11) * 2)/4))
= (5-(-22/4))
= (5-(-5)) // 22/4 = 5.4 since integers so, 5 will be taken .4 dropped!
= 5+5=10
```

2.1.2.8 Type Overflow & Underflow

When the value to be stored in a variable of a particular type is larger than the range of values that type can hold, we call it as overflow. Likewise when the value is smaller than the range of values the type can hold, we say it as underflow. Overflow and underflow are only a problem when dealing with integer arithmetic. This is because C simply ignores the situation and continues on as if nothing had happened. With signed integer arithmetic adding two large positive numbers, the result of which will be larger than the largest positive signed int, it will lead to a negative value being returned as the sign bit will be overwritten with the result of the sum. Similarly, with unsigned integer arithmetic adding 1 to the largest unsigned integer will give 0.

Floating point overflow is not a problem as the system itself is informed when it occurs which causes your program to terminate with a run-time error. If this happens we need to promote the variables involved in the offending operation to the largest possible and try again.

2.1.3 Control Statements

There are two types of control statements: iteration statements that allow us to repeat one or more simple statements a certain number of times and decision statements that allow us to choose to execute one sequence of instructions over one or more others depending on certain circumstances. Control statements are often regarded as compound statements in that they are normally combined with simpler statements which carry out the operations required. However, it should be noted that each control statement is still just a single statement from the compiler's point of view.

2.1.3.1 Decision Statements

2.1.3.1.1 If Statement

The ***if*** statement is the most general method for allowing conditional execution in C.

```
if ( condition ){
        statement body ;}
else
        {statement body ;}
```

or just :

```
if ( condition )
        {statement body ;}
```

In the first more general form of the statement one of two code blocks are to be executed. If the condition evaluates to TRUE the first statement body is executed otherwise for all other situations the second statement body is executed.

In the second form of the statement the statement body is executed if the condition evaluates to TRUE. No action is taken otherwise.

As with all other control statements the statement body can also involve multiple statements, again contained within curly braces.

Nested if statements

if – else statements like all other decision or iteration statements in C can be nested to whatever extent is required. Care should be taken however to ensure that the if and else parts of the statement are matched correctly -- the rule to follow is that the else statement matches the most recent unmatched if statement.

if - else - if ladder

When a programming situation requires the choice of one case from many different cases successive if statements can be tied together forming what is sometimes called an *if-else-if ladder.*

```
if ( condition_1 )
   statement_1 ;
else if ( condition_2 )
   statement_2 ;
else if ( condition_3 )
   statement_3 ;
. . .
else if ( condition_n )
   statement_n ;
else
   statement_default ;
```

Essentially, what we have here is a complete *if-else* statement hanging onto each else statement working from the bottom up.

Caution is advisable when coding the *if-else-if ladder* as it tends to be prone to error due to mismatched *if-else* clauses.

2.1.3.1.2 Conditional Operator ?

This is a special shorthand operator in C and replaces the following segment

```
if ( condition )
        expr_1 ;
else
        expr_2 ;
```

with the more elegant

```
condition ? expr_1 : expr_2 ;
```

or

```
variable=condition ? expr_1 : expr_2 ;
```

The ?: operator is a ternary operator in that it requires three arguments. One of the advantages of the ?: operator is that it reduces simple conditions to one simple line of code which can be thrown unobtrusively into a larger section of code.

2.1.3.1.3 The Switch Statement

This is a multi-branch statement similar to the ***if - else ladder*** (with limitations) but clearer and easier to code.

```
switch ( expression )
   {
   case constant1 :   statement1 ;
                      break ;
   case constant2 :   statement2 ;
                      break ;
```

```
...
default :   statement ;
}
```

The value of expression is tested for equality against the values of each of the constants specified in the **case** statements in the order written until a match is found. The statements associated with that case statement are then executed until a break statement or the end of the switch statement is encountered.

When a break statement is encountered execution jumps to the statement immediately following the switch statement.

The default section is optional, however if it is not included the default is that nothing happens and execution simply falls through the end of the switch statement.

The switch statement however is limited by the following

- Can only test for equality with **integer constants** in case statements.
- No two case statement constants may be the same.
- Character constants are automatically converted to integer.

Note

The break statement need not be included at the end of the case statement body.

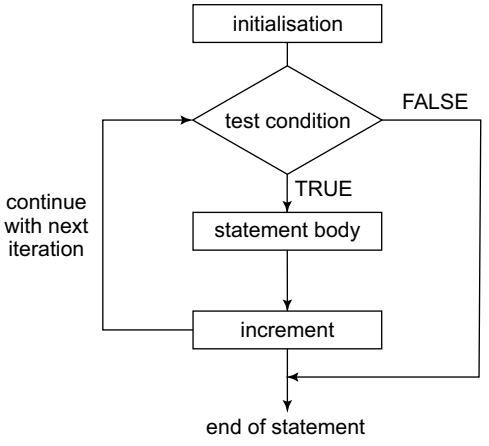
2.1.3.2 Iteration Statements

2.1.3.2.1 For Statement

The ***for*** statement is most often used in situations where the programmer knows in advance how many times a particular set of statements are to be repeated. The for statement is sometimes termed a counted loop.

```
for ([initialisation] ; [condition] ; [increment])
              [statement body] ;
```

initialisation: this is usually an assignment to set a loop counter variable for example.
condition: determines when loop will terminate.
increment: defines how the loop control variable will change each time the loop is executed.
statement body: can be a single statement, no statement or a block of statements.
The *for* statement executes as follows:



Note

The square braces above are to denote optional sections in the syntax but are not part of the syntax. The semi-colons must be present in the syntax.

Multiple Initialisations

C has a special operator called the **comma operator** which allows separate expressions to be tied together into one statement. For example it may be tidier to initialise two variables in a loop as follows:

```
for ( x = 0, sum = 0; x <= 100; x++ )
        {
          printf( "%d\n", x) ;
          sum += x ;
        }
```

Any of the four sections associated with a for loop may be omitted but the semi-colons must be present always. Thus, an infinite loop may be created as follows:

```
for ( ;  ;  )
        statement body ;
```

or indeed by having a faulty terminating condition.

Sometimes a statement may not even have a body to execute as in the following example where we just want to create a time delay.

```
for ( t = 0; t < big_num ; t++ )  ;
```

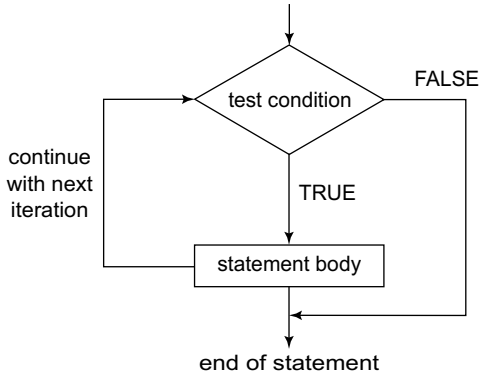
It is possible to build a nested structure of for loops, for example the following creates a large time delay using just integer variables.

```
unsigned int x, y ;
for ( x = 0; x < 65535; x++ )
        for ( y = 0; y < 65535; y++ ) ;
```

While statement

The ***while*** statement is typically used in situations where it is not known in advance how many iterations are required.

```
while ( condition )
        statement body ;
```

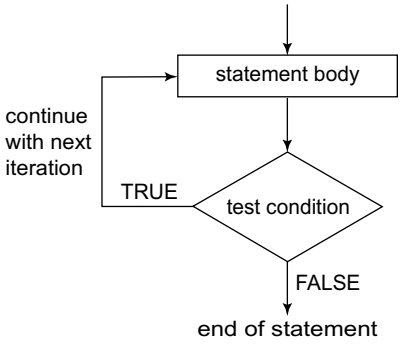


A *for* loop is of course the more natural choice where the number of loop iterations is known beforehand whereas a *while* loop caters for unexpected situations more easily.

do-while

The terminating condition in the *for* and *while* loops is always tested before the body of the loop is executed, so of course the body of the loop may not be executed at all. In the *do while* statement on the other hand the statement body is always executed at least once as the condition is tested at the end of the body of the loop.

```
do
{
statement body ;
}  while ( condition ) ;
```



For example: To read in a number from the keyboard until a value in the range 1 to 10 is entered.

```
int i ;
do {
  scanf( "%d\n", &i ) ;
  _flushall() ;
}while ( i < 1 || i > 10 ) ;
```

In this case we know at least one number is required to be read so the *do-while* might be the natural choice over a normal *while* loop.

break statement

When a ***break*** statement is encountered inside a while (or for, do/while or switch) statement the statement is immediately terminated and execution resumes at the next statement following the while (or for, do/while or switch) statement. For example:

```
for ( x = 1 ; x <= 10  ; x++  ){
   if ( x > 4  )  break ;
   printf( "%d  " , x ) ;
}
```

Output : "1 2 3 4"

...

continue statement

The ***continue*** statement terminates the current iteration of a *while*, *for* or *do*/*while* statement and resumes execution back at the beginning of the loop body with the next iteration. For example:

```
int n=0, s=0,m;
while(1){
scanf("%d", &m);
if(m==0) break;
if(m%2) continue;
s+=m;
n++;
}
if(n) printf("%d\n", s/n);
```

Consider the above code fragment:

It is an infinite loop which reads integers in each iteration. It runs till we enter 0. If we enter 0, the control comes out of the loop and prints the average because of execution of *break* statement. If the number entered is not even then it executes continue statement making program execution to go to *while*(1) statement. The statements after continue will not be executed in that iteration. Control simply goes to *while*(1) when we execute *continue* statement.

Also, consider the following code fragment. If x value is 3, then control goes to x++. Thus, we get 1, 2 4, 5 as output.

```
for ( x = 1; x <= 5; x++ ){
   if ( x == 3 ) continue ;
   printf( "%d  ", x ) ;
}
```

2.1.3.3 Unnecessary Type Conversions

C is a weakly typed language. Most unnecessary conversions occur in assignments, arithmetic expressions and parameter passing. Consider the following code segment which simply computes the sum of a user input list of integers and their average value.

```
double average, sum = 0.0 ;
short value, i ;
...
for ( i=0; i < 1000; i ++ ) {
        scanf( "%d", &value ) ;
        sum = sum + value ;
        }
average = sum / 1000 ;
```

1. The conversion from value, of type short *int*, to the same type as sum, type double, occurs 1000 times in the *for* loop so the inherent inefficiency in that one line is repeated 1000 times which makes it substantial.

   If we redefine the variable sum to be of type short we will eliminate these conversions completely. However as the range of values possible for a short are quite small we may encounter overflow problems so we might define sum to be of type long instead. The conversion from short to long will now be implicit in the statement but it is more efficient to convert from short to long than it is from short to double.

2. Because of our modifications above the statement

   ```
   average = sum / 1000 ;
   ```

   now involves integer division which is not what we require here. (Note however that an implicit conversion of 1000 from *int* to long occurs here which may be simply avoided as follows:

   ```
   average = sum / 1000L ;
   ```

   with no time penalty whatsoever as it is carried out at compile time.)

   To remedy the situation we simply do the following:

   ```
   average = sum / 1000.0 ;
   ```

3. The statement

   ```
   sum = sum + value ;
   ```

   also involves another source of inefficiency. The variable sum is loaded twice in the statement unnecessarily. If the shorthand operator += were used instead we will eliminate this.

   ```
   sum += value ;
   ```

2.1.4 Arrays and Strings

An array is a collection of variables of the same type that are referenced by a common name. Specific elements or variables in the array are accessed by means of an index into the array. In C all arrays consist of contiguous memory locations. The lowest address corresponds to the first element in the array while the largest address corresponds to the last element in the array. C supports both single and multi-dimensional arrays.

Arrays are needed if we need to use same related data more than once in a program. For instance, consider the problem of calculating average marks of those students whose marks are more than class average. Without array if we want the program, first we have to read all the students marks to calculate average and then second time also we have to read all

the students marks to compare with the class average. If we plan to use array, once we can read the students marks into an array and use any number of times. Thus, input operations of a program can be reduced; thus program becomes fast.

2.1.4.1 Single Dimensional Arrays

We can declare 1-D array as:
type A[size] ;
where type is the type of each element in the array, A is any valid C identifier, and size is the number of elements in the array which has to be an integer constant value.

In C language, all arrays use zero as the index of their first element.

For example a five element integer array can be declared as:

```
int array[ 5 ] ;
```

The above array storage in the memory can be considerd as follows for a 32-bit system where each int requires 4 bytes.

array[0]	12	loc^n 1000
array[1]	–345	loc^n 1004
array[2]	342	loc^n 1008
array[3]	–30000	loc^n 1012
array[4]	23455	loc^n 1016

The valid indices for array above are 0 .. 4, i.e. 0 to number of elements – 1. To determine to size of an array at run time the sizeof operator can be used. This returns the size in bytes of its argument. The name of the array is given as the operand

size_of_array = sizeof (array_name) ;

We can also declare the array as:

```
#define N 5
int array[N];
```

where N is a symbolic constant,
C carries out no boundary checking on array access; thus the program has to ensure that array element accesses should be within the bounds of the array. If the program tries to access an array element outside of the bounds of the array, C will try and accommodate the operation. For example, if a program tries to access element array[5] above which does not exist the system will give access to the location where element array[5] should be i.e. 5 x 4 bytes from the beginning of the array.

array[0]	12	loc^n 1000
array[1]	–345	loc^n 1004
array[2]	342	loc^n 1008
array[3]	–30000	loc^n 1012
array[4]	23455	loc^n 1016
array[5]	123	loc^n 1020

This piece of memory does not belong to the array and is likely to be in use by some other variable in the program. If we are just reading a value from this location the situation isn't so drastic our logic just goes haywire. However if we are writing to this memory location we will be changing values belonging to another section of the program which can be catastrophic.

Initialising Arrays

Arrays can be initialised at time of declaration in the following manner.

```
type array[ size ] = { value list };
```

For example:

```
int i[5] = {1, 2, 3, 4, 5 } ;
```

i[0] = 1, i[1] = 2, etc. We can also refer the elements as: 0[i], 1[i], etc.

The size specification in the declaration may be omitted which causes the compiler to count the number of elements in the value list and allocate appropriate storage. For example: the following declaration allocates 5 elements for array A.

```
int A[ ] = { 1, 2, 3, 4, 5 } ;
```

2.1.4.2 Strings

In C a string is defined as a character array which is terminated by a special character, the null character '\0', as there is no string type as such in C. Thus, the string or character array must always be defined to be one character longer than is needed in order to cater for the '\0'. For example, we can declare a string to hold 5 characters as:

```
char s[6] ;
```

A string constant is simply a list of characters within double quotes. For example "Hello" with the '\0' character being automatically appended at the end by the compiler. A string may be initialised as simply as follows :

```
char s[6] = "Hello" ;
```

'H'	'e'	'l'	'l'	'o'	'\0'

as opposed to

```
char s[6] = { 'H', 'e', 'l', 'l', 'o', '\0' } ;
```

Again the size specification may be omitted allowing the compiler to determine the size required.

2.1.4.3 Multidimensional Arrays

Multidimensional arrays of any dimension are possible in C but in practice only two or three dimensional arrays are workable. The most common multidimensional array is a two dimensional array **for example the computer display, board games, a mathematical matrix etc.**

```
type   name [ rows ] [ columns ]  ;
```

For example: A 2D integer array of dimension 2×3 can be declared as:

```
int d[ 2 ] [ 3 ] ;
```

d[0][0]	d[0][1]	d[0][2]
d[1][0]	d[1][1]	d[1][2]

A two dimensional array is actually an array of arrays, in the above case an array of two integer 1-D arrays (the rows) each with three elements, and is stored row-wise in memory. That is, first all the elements of 0^{th} row are stored in memory, then all the elements of 1^{st} row, and vice versa. To initialise a multidimensional array all but the **leftmost** index must be specified so that the compiler can index the array properly. For example:

```
int d[ 5 ] [ 3 ] = { 1, 2, 3, 4, 5, 6 };
int d[ ] [ 3 ] = { 1, 2, 3, 4, 5, 6 };
```

However, it is more useful to enclose the individual row values in curly braces for clarity as follows.

```
int d[ ] [ 3 ] = { {1, 2, 3}, {4, 5, 6} } ;
int d[5 ] [ 3 ] = { {1, 2, 3}, {4, 5, 6} } ;
```

In all the above initialisations, first and second rows are filled with the specified integer constants.

2.1.4.4 Arrays of Strings

An array of strings is in fact a two dimensional array of characters but it is more useful to view this as an array of individual single dimension character arrays or strings. For example:

```
char A[ 10 ] [ 30 ] ;
```

where the row index is used to access the individual row strings and where the column index is the size of each string, thus A is an array of 10 strings each with a maximum size of 29 characters leaving one extra for the terminating null character.

We can also declare and initialise a 2-D character array as:

```
char A[10][20]={ "Rama", "Abhi", "Anuj"}  or
char A[][20]={ "Rama", "Abhi", "Anuj"}
```

Here, A[0] is string "Rama", A[1] is "Abhi" and vice versa. However, in the later one, number of strings or rows becomes 3.

2.1.5 Functions

Functions are essentially just groups of statements that are to be executed as a unit in a given order and that can be referenced by a unique name. The only way to execute these statements is by invoking them or calling them using the function's name.

Traditional program design methodology typically involves a ***top-down*** or structured approach to developing software solutions. The main task is first divided into a number of simpler sub-tasks. If these sub-tasks are still too complex they are subdivided further into simpler sub-tasks, and so on until the sub-tasks become simple enough to be programmed easily.

Functions are the highest level of the building blocks given to us in C and correspond to the sub-tasks or logical units referred to above. The identification of functions in program design is an important step and will in general be a continuous process subject to modification as more becomes known about the programming problem in progress.

Syntax of a function definition is given as:

```
return_type function_name (  parameter_list )
  {
  body of function ;
  }
```

Many functions will produce a result or return some information to the point at which it is called. These functions specify the type of this quantity via the ***return_type*** section of the function definition. If the return type is *void* it indicates the function returns nothing.

The ***function_name*** may be any valid C identifier and must be unique in a particular program.

If a function requires information from the point in the program from which it is called this may be passed to it by means of the ***parameter_list***. The parameter list must identify the names and types of all of the parameters to the function individually. If the function takes no parameters the braces can be left empty or use the keyword void to indicate that situation more clearly.

2.1.5.1 Function Prototype (declaration or signature)

When writing programs in C it is normal practice to write the main() function first and to position all user functions after it or indeed in another file. Thus if a user function is called directly in main() the compiler will not know anything about it at this point, i.e. if it takes parameters etc. This means we need to give the compiler this information by providing a function prototype or declaration before the function is called.

```
type_spec function_name( type_par1, type_par2, etc. );
```

This declaration simply informs the compiler what type the function returns and what type and how many parameters it takes. Names may or may not be given to the parameters at this time.

2.1.5.2 Function Definition and Local Variables

A function definition actually defines what the function does and is essentially a discrete block of code which cannot be accessed by any statement in any other function except by formally calling the function. Thus any variables declared and used in a function are private or local to that function and cannot be accessed by any other function.

Local variables are classed as automatic variables because each time a function is called the variable is automatically created and is destroyed when the function returns control to the calling function. By created we mean that memory is set aside to store the variable's value and by destroyed we mean that the memory required is released. Thus a local variable cannot hold a value between consecutive calls to the function.

2.1.5.3 Static Local Variables

The keyword static can be used to force a local variable to retain its value between function calls. The static variable is created when the function is first called. The variable retains its last value during subsequent calls to the function and is only destroyed when the program terminates.

2.1.5.4 Scope Rules

The scope of an identifier is the area of the program in which the identifier can be accessed.

Identifiers declared inside a code block are said to have block scope. The block scope ends at the terminating curly brace '}' of the code block. Local variables for example are visible, i.e. can be accessed, from within the function, i.e. code block, in which they are declared. Any block can contain variable declarations be it the body of a loop statement, if statement, etc. or simply a block of code marked off by a curly brace pair. When these blocks are nested and an outer and inner block contain variables with the same name then the variable in the outer block is inaccessible until the inner block terminates.

Global variables are variables which are declared outside all functions and which are visible to all functions from that point on. These are said to have file scope.

All functions are at the same level in C, i.e. cannot define a function within a function in C. Thus within the same source file all functions have file scope i.e. all functions are visible or can be called by each other (assuming they have been prototyped properly before they are called).

2.1.5.5 Return Value

The **return** statement is used to return a value to the calling function if necessary.

return expression ;

If a function has a return type of type void the expression section can be omitted completely or indeed the whole return statement can be omitted and the closing curly brace of the function will cause execution to return appropriately to the calling function.

The return value of the function need not always be used when calling it. In the above example if we are not interested in know how often hello() has been called we simply ignore that information and invoke the function with

```
hello() ;
```

When the main() function returns a value, it returns it to the operating system. Zero is commonly returned to indicate successful normal termination of a program to the operating system and other values could be used to indicate abnormal termination of the program. This value may be used in batch processing or in debugging the program.

2.1.5.6 Function Arguments

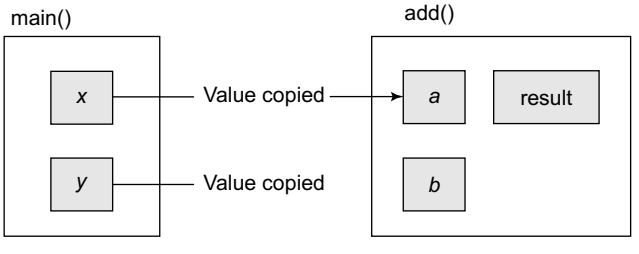
The types of all function arguments should be declared in the function prototype as well as in the function definition.

In C, arguments are passed to functions using the *call-by-value* scheme. This means that the compiler copies the value of the argument passed by the calling function into the formal parameter list of the called function. Thus if we change the values of the formal parameters within the called function we will have no effect on the calling arguments. The formal parameters of a function are thus local variables of the function and are created upon entry and destroyed on exit.

```
int add ( int a, int b )     {
int result ;
result = a + b ;
return result ; // parentheses used for clarity here
}
```

In the formal parameter list of a function the parameters must be individually typed.

The add() function here has three local variables, the two formal parameters and the variable result. There is no connection between the calling arguments, x and y, and the formal parameters, a and b, other than that the formal parameters are initialised with the values in the calling arguments when the function is invoked. The situation is depicted below to emphasise the independence of the various variables.



2.1.5.7 Recursion

A recursive function is a function that calls itself either directly or indirectly through another function. For example to evaluate the factorial of a number, n

$n! = n * n–1 * n–2 * ... * 3 * 2 * 1.$

We can represent factorial calculation recursively as somehow calculate factorial n-1 and multiply it with n to get factorial n. That is,

$n! = n * (n-1)!$

where the original problem has been reduced in complexity slightly. We continue this process until we get the problem down to a task that may be solved directly, in this case as far as evaluating the factorial of 1 which is simply 1.

So a recursive function to evaluate the factorial of a number will simply keep calling itself until the argument is 1. All of the previous (n-1) recursive calls will still be active waiting until the simplest problem is solved before the more complex intermediate steps can be built back up giving the final solution.

```
short factorial( short num ) {
if ( num <= 1 ) return 1 ;
else
   return ( num * factorial( num - 1 ) ) ;
}
```

This function will not work very well as it is, because the values of factorials grow very large very quickly. For example, the value of 8! is 40320 which is too large to be held in a short so integer overflow will occur when the value entered is greater than 7. What can be the solution to this ?

While admittedly simple to encode in certain situations programs with recursive functions can be slower than those without because there is a time delay in actually calling the function and passing parameters to it. There is also a memory penalty involved. If a large number of recursive calls are needed this means that there are that many functions active at that time which may exhaust the machine's memory resources. Each one of these function calls has to maintain its own set of parameters on the program stack.

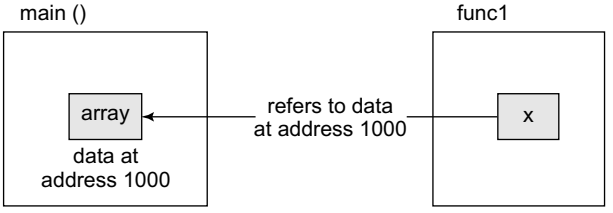
2.1.5.8 Arrays as Arguments to Functions (1-D)

In C it is impossible to pass an entire array as an argument to a function—instead the address of the array is passed as a parameter to the function.

The name of an array without any index is the address of the first element of the array and hence of the whole array as it is stored contiguously. However we need to know the size of the array in the function—either by passing an extra parameter or by using the size of operator. For example:

```
void main(){
int array[20] ;
func1( array ) ;/* passes pointer to array to func1 */
}
```

Since, we are passing the address of the array the function will be able to manipulate the actual data of the array in main(). This is call by reference as we are not making a copy of the data but are instead passing its address to the function. Thus the called function is manipulating the same data space as the calling function.



In the function receiving the array the formal parameters can be declared in one of three almost equivalent ways as follows:

- As a sized array :

```
func1 ( int x[10] ) {
...
}
```

- As an unsized array :

```
func1 ( int x[ ] ) {
...
}
```

- As an actual pointer

```
func1 ( int *x ) {
...
}
```

All three methods are identical because each tells us that in this case the address of an array of integers is to be expected.

Note that in cases 2 and 3 above where we specify the formal parameter as an unsized array or simply as a pointer we cannot determine the size of the array passed in using the sizeof operator as the compiler does not know what dimensions the array has at this point. Instead size of returns the size of the pointer itself, two in the case of near pointers in a 16-bit system but four in 32-bit systems.

2.1.5.9 Passing Multidimensional Arrays

Function calls with multi-dimensional arrays will be the same as with single dimension arrays as we will still only pass the address of the first element of the array.

However, to declare the formal parameters to the function we need to specify all but one of the dimensions of the array so that it may be indexed properly in the function. For example :

2D array of doubles: double x[10][20] ;
Call func1 with x a parameter: func1(x) ;
Declaration in func1:

```
func1( double y[ ][20] ) {
...
}
```

The compiler must at least be informed how many columns the matrix has to index it correctly. For example to access element y[5][3] of the array in memory the compiler might do the following

element No = 5 * 20 + 3 = 103.

Multi-dimensional arrays are stored row-wise so y[5][3] is the 4^{th} element in the 6^{th} row.

Since, we are dealing with an array of doubles this means it must access the memory location 103 X 8 bytes from the beginning of the array.

Thus the compiler needs to know how many elements are in each row of the 2D array above. In general the compiler needs to know all dimensions except the leftmost at the very least.

2.1.5.10 #define directive

This is a pre-processor command which is used to replace any text within a C program with a more informative pseudonym. For example:

```
#define  PI  3.14159
```

When the pre-processor is called it replaces each instance of the phrase PI with the correct replacement string which is then compiled. The advantage of using this is that if we wish to change the value of PI at any stage we need only change it in this one place rather than at each point the value is used.

2.1.5.11 Macros

Macros make use of the #define directive to replace a chunk of C code to perform the same task as a function but will execute much faster since the overhead of a function call will not be involved. However, the actual code involved is replaced at each call to the macro so the program will be larger than with a function.

```
#define LOOP      for(i=0;i<n;i++)
```

Macros can also be defined so that they may take arguments. For example:

```
#define  PR( fl )  printf( "%8.2f ", fl )
void main(){
float num = 10.234 ;
PR( num ) ;
}
```

What the compiler actually sees is : printf("%8.2f ", num) ;

While admittedly advantageous and neat in certain situations care must be taken when coding macros as they are notorious for introducing unwanted side effects into programs. For example:

```
#define MAX(A, B)   ( (A) > (B) ? (A) : (B) )
   void main( ){
   int i = 20, j = 40 , k ;
   k = MAX( i++, j++ ) ;
   printf( " i = %d, j = %d\n", i, j );
   ...
   }
```

The above program might be expected to output the following

i = 21, j = 41

whereas it produces the following as its output

i = 21, j = 42

where the larger value is incremented twice. This is because the macro MAX is actually translated into the following by the compiler

```
( ( i++ ) > ( j++ ) ? ( i++ ) : ( j++ ) )
```

so that the larger parameter is incremented twice in error since parameter passing to macros is essentially text replacement.

Take another example:

```
#define  norm(a,b)  sqrt(a*a + b*b)
float p=3, q=4, r, s;
r=norm(p,q);
s=norm(p+1, q+1);
```

We might be expecting r and s values as 5 and square root of 41. However, we get 5 and 4. This is because, when expanded the second one becomes: s = sqrt(p+1*p+1+q+1*q+1). That is, sqrt(2p + 1 + 2q + 1).

As we have mentioned previously the use of functions involves an overhead in passing parameters to them and obtaining a return value from them. For this reason they can slow down your program if used excessively.

The alternative to this is to use macros in place of functions. This eliminates the penalty inherent in the function call but does make the program larger. Therefore, in general macros should only be used in place of small 'would be' functions.

The penalty involved in the function call itself is also the reason why iterative methods are preferred over recursive methods in numerical programming.

2.1.6 Pointers

Pointers are one of the most important mechanisms in C. A ***pointer*** is a ***variable*** that is used to store a ***memory address***. Most commonly the address is the location of another variable in memory. If one variable holds the address of another then it is said to point to the second variable.

Address	Value	Variable
1000		
1004	1012	ptr
1008		
1012	23	A
1016		

In the above illustration, A is a variable of type *int* with a value 23 and stored at memory location 1012. *ptr* is a variable of type *pointer to int* which has a value of 1012 and is stored at memory location 1004. Thus *ptr* is said to point to the variable A and allows us to refer indirectly to it in memory.

Note

It should be remembered that *ptr* is a variable itself with a specific piece of memory associated with it, in this 32-bit case four bytes at address 1004 which is used to store an address.

2.1.6.1 Pointer Variables

Pointers like all other variables in C must be declared as such prior to use.

```
type   *ptr ;
```

which indicates that *ptr* is a pointer to a variable of type *type*. For example

```
int  *ptr ;
```

declares a pointer ptr to variables of type int.

Note

The type of the pointer variable ptr is ***int*** *. The declaration of a pointer variable normally sets aside just two or four bytes of storage for the pointer. In 16-bit systems two byte pointers are termed near pointers and are used in small memory model programs where all addresses are just segment offset addresses and 16 bits in length. In larger memory model programs, addresses include segment and offset addresses and are 32 bits long and thus pointers are 4 bytes in size and are termed far pointers. In 32-bit systems we have a flat address system where every part of memory is accessible using 32-bit pointers.

2.1.6.2 Pointer Operators * and &

We already know that **&** is a unary operator that returns the **address** of its operand which must be a variable. For Example:

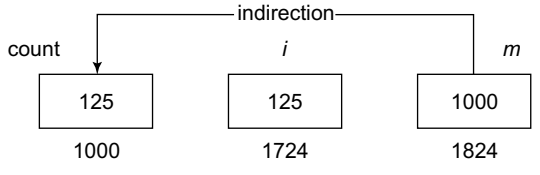
```
int *m ;
int count=125, i;
m = &count ; /* m is a pointer to int, count, i are integers */
```

The address of the variable count is placed in the pointer variable m.

The * operator is the complement of the address operator **&** and is normally termed as the indirection operator. Like the **&** operator it is a unary operator and it returns the **value** of the variable located at the address its operand points. For example:

```
i = *m ;
```

assigns the value (count variable value) which is located at the memory location whose address is stored in m, to the integer i. So essentially in this case we have assigned the value of the variable count to the variable i. The final situation is given as follows :



One of the most frequent causes of error when dealing with pointers is using an un-initialised pointer. Pointers should be initialised when they are declared or in an assignment statement. Like any variable if we do not specifically assign a value to a pointer variable it may contain any value. This is extremely dangerous when dealing with pointers because the pointer may point to any arbitrary location in memory, possibly to an unused location but also possibly to a memory location that is used by the operating system. If our program tries to change the value at this address it may cause the whole system to crash. Therefore, it is important to initialise all pointers before use either explicitly in our program or when defining the pointer. A pointer may also be initialised to 0 (zero) or NULL which means it is pointing at nothing. This will cause a run-time error if the pointer is inadvertently used in this state. It is useful to be able to test if a pointer has a null value or not as a means of determining if it is pointing at something useful in a program.

Note that NULL is #defined in <stdio.h>.

For Example:

```
int var1, var2 ;
int *ptr1, *ptr2 = &var2 ;
int *ptr3 = NULL ;
...
ptr1 = &var1 ;
```

ptr1 and *ptr2* are now pointing to data locations within the program so we are free to manipulate them at will, i.e. we are free to manipulate the piece of memory they point to.

2.1.6.3 Call by Reference

We can write function which takes addresses as their arguments. This style if called as passing by address. When we send address of a scalar variable to a function then what ever operations the function does in this address really takes place on the actual argument. The following example uses this concept to exchange values of two integer variables.

```
#include <stdio.h>
void swap( int *, int  * ) ;
void main( ){
int a, b ;
printf( "Enter two numbers" ) ;
scanf( " %d %d ", &a, &b ) ;
printf( "a = %d ;  b = %d \n", a, b ) ;
swap( &a, &b ) ;
printf( "a = %d ;  b = %d \n", a, b ) ;
}
void swap ( int  *ptr1, int  *ptr2 ) {
int temp ;
temp = *ptr2 ;
*ptr2 = *ptr1 ;
*ptr1 = temp ;
}
```

The swap() function is now written to take integer pointers as parameters and so is called in main() as

```
swap( &a, &b ) ;
```

where the addresses of the variables are passed and copied into the pointer variables in the parameter list of swap(). These pointers must be de-referenced to manipulate the values, and it is values in the same memory locations as in main() we are swapping unlike the previous version of swap where we were only swapping local data values.

2.1.6.4 Pointers and Arrays

There is a very close relationship between pointer and array notation in C. As we have seen already the name of an array (or string) is actually the address in memory of the array and so it is essentially a **constant pointer.** For example:

```
char str[80], *ptr ;
ptr =  str ;
/* causes ptr to point to start of string str  */
ptr = &str[0] ;
/* this performs the same as above */
```

It is illegal however to do the following.

```
str = ptr ;       /* illegal */
```

as str is a constant pointer and so its value i.e. the address it holds cannot be changed.

Instead of using the normal method of accessing array elements using an index we can use pointers in much the same way to access them as follows.

```
char str[80], *ptr , ch;
ptr = str ;// position the pointer appropriately
*ptr = 'a' ;// access first element i.e. str[0]
ch = *( ptr + 1 )  ;// access second element i.e. str[1]
```

Thus ***(array + index)** is equivalent to **array[index]**.

Note that the parentheses are necessary above as the precedence of * is higher than that of +. The expression

```
ch = *ptr + 1 ;
```

for example says to access the character pointed to by ptr (str[0] in above example with value 'a') and to add the value 1 to it. This causes the ASCII value of 'a' to be incremented by 1 so that the value assigned to the variable ch is 'b'.

In fact so close is the relationship between the two forms that we can do the following.

```
int x[10], *ptr ;
ptr = x ;
ptr[4] = 10 ; /*accesses element 5 of array by indexing a pointer*/
```

2.1.6.5 Pointer Arithmetic

Pointer variables can be manipulated in certain limited ways. Many of the manipulations are most useful when dealing with arrays which are stored in contiguous memory locations. Knowing the layout of memory enables us to traverse it using a pointer and not get completely lost.

Assignment

```
int count, *p1, *p2 ;
p1 = &count ;// assign the address of a variable directly
p2 = p1 ;//assign the value of another pointer variable
```

Pointer Addition / Subtraction

The value a pointer holds is just the address of a variable in memory, which is normally a four byte entity. It is possible to modify this address by integer addition and subtraction if necessary. Consider the following we assume a 32-bit system and hence 32-bit integers.

```
int *ptr ;
int array[3] = { 100, 101, 102 } ;
ptr = array ;
```

		Address	Value
	ptr	1000	2008
	array[0]	2008	100
	array[1]	2012	101
	array[2]	2016	102

We now have the pointer variable ptr pointing at the start of *array* which is stored at memory location 2008 in our illustration. Since we know that element array[1] is stored at address 2012 directly after element array[0] we could perform the following to access its value using the pointer.

```
ptr += 1 ;
```

This surprisingly will cause ptr to hold the value 1012 which is the address of array[1], so we can access the value of element array[1]. The reason for this is that ptr is defined to be a pointer to type int, which are four bytes in size on a 32-bit system. When we add 1 to ptr what we want to happen is to point to the **next integer** in memory. Since an integer requires four bytes of storage the compiler increments ptr by 4. Likewise a pointer to type char would be incremented by 1, a pointer to float by 4, etc.

Similarly we can carry out integer subtraction to move the pointer backwards in memory.

```
ptr = ptr - 1 ;
ptr -= 10 ;
```

The shorthand operators ++ and -- can also be used with pointers. In our continuing example with integers the statement ***ptr++ ;*** will cause the address in ptr to be incremented by 4 and so point to the next integer in memory and similarly ***ptr--* ;** will cause the address in ptr to be decremented by 4 and point to the previous integer in memory.

Note

Two pointer variables may not be added together (it does not make any logical sense).

```
char *p1, *p2 ;
p1 = p1 + p2 ; /* illegal operation */
```

Two pointers may however be subtracted as follows.

```
int *p1, *p2, array[3], count ;
p1 = array ;
p2 = &array[2] ;
count = p2 - p1 ; /* legal */
```

The result of such an operation is not however a pointer, it is the number of elements of the base type of the pointer that lie between the two pointers in memory.

Pointer Comparisons

We can compare pointers using the relational operators ==, <, and > to establish whether two pointers point to the same location, to a lower location in memory, or to a higher location in memory. These operations are again used in conjunction with arrays when dealing with sorting algorithms etc.

Strings and pointers

Strings can be initialised using pointer or array notation as follows

```
char *str = "Hello\n" ;
char string[] = "Hello\n" ;
```

in both cases the compiler allocates just sufficient storage for both strings. However, str is dynamic pointer variable while string is constant pointer variable.

2.1.6.6 Arrays of Pointers

It is possible to declare arrays of pointers in C the same as any other 'type'. For example

```
int *x[10] ;
```

declares an array of ten integer pointers.

To make one of the pointers point to a variable one might do the following.

```
x[ 2 ] = &var ;
```

To access the value pointed to by x[2] we would do the following

```
*x[ 2 ]
```

which simply de-references the pointer x[2] using the * operator.

Passing this array to a function can be done by treating it the same as a normal array which happens to be an array of elements of type int *. For example:

```
void display( int *q[ ], int size ){
  int t ;
  for ( t=0; t < size; t++ )
        printf( "%d ", *q[t] ) ;
  }
```

Note that q is actually a pointer to an array of pointers as we will see later on with multiple indirection.

A common use of pointer arrays is to hold arrays of strings.

2.1.6.7 Command Line Arguments

Command line arguments allow us to pass information into the program as it is run. For example the simple operating system command *type* uses command line arguments as follows

```
c:\\Windows>type text.dat
```

where the name of the file to be printed is taken into the type program and the contents of the file then printed out.

In C there are two in-built arguments to the main() function commonly called **argc** and **argv** which are used to process command line arguments.

```
void main( int argc,  char *argv[ ] ) {
...
}
```

argc is used to hold the total number of arguments used on the command line which is always at least one because the program name is considered the first command line argument.

argv is a pointer to an array of pointers to strings where each element in argv points to a command line argument. For example argv[0] points to the first string, the program name. The above thing can be also written as:

```
void main( int argc,  char **argv) {
...
}
```

The following version of main is used to access environment variables.

```
void main( int argc,  char *argv[ ], char **environ ) {
...
}
```

2.1.6.8 Dynamic Memory Allocation

This is the means by which a program can obtain and release memory at run-time. This is very important in the case of programs which use large data items, e.g. databases which may need to allocate variable amounts of memory or which might have finished with a particular data block and want to release the memory used to store it for other uses.

We already know that while creating language supported arrays, we need to supply size of array as a constant. This style has some drawbacks such as: 1. Program lacks flexibility, 2. There will be a great wastage of memory. In fact, all applications will not know in advance how many elements are needed in for arrays. By declaring or creating dynamic arrays, we can alleviate the above draw backs in addition to increase the utility of memory.

The functions **malloc()** and **free()** form the core of C's dynamic memory allocation and are prototyped in <malloc. h>. malloc() allocates memory from the heap, i.e. unused memory while available and free() releases memory back to the heap.

The following is the prototype for the malloc() function

```
void * malloc( size_t num_bytes ) ;
```

malloc() allocates num_bytes bytes of storage and returns a pointer to type void to the block of memory if successful, which can be cast to whatever type is required. If malloc() is unable to allocate the requested amount of memory it returns a NULL pointer.

For example to allocate memory for 100 characters we might do the following

```
#include <malloc.h>
void main(){
char *p ;
if ( !( p = malloc( sizeof( char ) * 100 ) )  {
     puts( "Out of memory" ) ;
     exit(1) ;
     }
}
```

The return type void * is automatically cast to the type of the lvalue type but to make it more explicit we would do the following

```
if (  !( (char * )p = malloc( sizeof( char  ) * 100  ) ){
          puts( "Out of memory" ) ;
          exit(1) ;
          }
```

To free the block of memory allocated we do the following

```
free ( p ) ;
```

There are a number of memory allocation functions included in the standard library including calloc( ), _fmalloc(), etc. Care must be taken to ensure that memory allocated with a particular allocation function is released with its appropriate de-allocation function, e.g. memory allocated with malloc() is freed only with free().

Consider the following example:

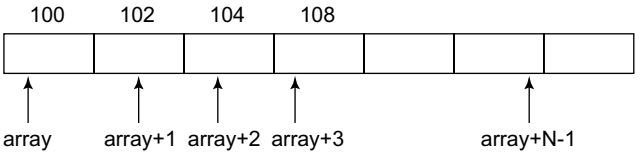
```
int *array, N, i,j;
printf("Enter the size of the array to be created \n");
scanf("%d", &N);
/* allocating memory for the array */
array=(int *) malloc(N* sizeof(int));
```

In the above statement, **malloc()** function is called to allocate a chunk of memory. As we are interested to create 1-D integer array; for the portability reasons we are sending value of the expression **N*sizeof(int)** to specify required memory. The malloc function allocates memory and returns the generic address of the same. In reality, we can see this chunk of memory whatever way we want. The same can be seen as array of 1-D characters, or integer array or float array. However, type casting the returned address to integer address, we are assigning address of first element to the pointer variable **array,** and the allocated memory can be seen as 1-D integer array. If **array** value is the address of first element, then **array+1, array+2, …** are the addresses of subsequent elements. If **array** value is 100 and the machine uses 2 byte integers, then the addresses of subsequent elements will be 102, 104, 106, etc.

Similarly, if we want to look at the chunk of memory as 1-D float array then we have to declare a float type of pointer variable and the starting address of this chunk of memory has to be assigned to it after type casting. If **array** is pointer to such a array and its value is 100 then subsequent elements addresses becomes 104, 108, etc., as usually float takes 4 bytes. **Thus, pointer arithmetic's depends on pointer type.**

Note

Dynamic memory life is entire life of the program or till we free it. Scope is also global. Any function can access it if it knows the array dynamic address.



Summary

- **array** is pointer to first element of the array
- **array+i** is the address of the i'th element of the array which is created
- ***(array+i)** is the value of the i'th element of the dynamic array created
- with dynamic array also, we can use **array[i]** to access value of the i'th element.

Reading the data into array

In general, while reading, we have to inform the **scanf()** function that what type of data to be read and where (memory cell number) it has to be stored. Thus, in the following line we are trying to read data into the dynamic array which pointed by pointer, **array**. As usual, we have to read data into an array, element by element by supplying element addresses to **scanf()** function. Thus, here, we supply **array+i** to **scanf()** function as second argument to inform that the **scanf()** about where to store the data which it reads. Here, **array+i** is the address of the ith element of the dynamic array. In a nutshell, the following statement reads data into dynamically created 1-D array.

```
for(i=0;i<N;i++)  scanf("%d", array+i );
```

In order to print the array element either we can use indirection operator to address, i.e., *(array+i) or array[i].

```
for(i=0;i<N;i++) printf("%d  %d\n", *(array+i), array[i] );
```

Increment and Decrement Operators on Pointers

Like normal variables, we can also apply increment and decrement operators on (dynamic) pointers. Consider the following code fragment. We have pointer **p** which points a dynamically created array having three elements 17, 2 and 3. Assuming **p** is pointing to first element of the array, i.e., **p** value is address of the first element which is **0x0010**. Assuming that integer takes 4 bytes on our machine, **p++** make **p** to point to next element in the array, i.e., 2. That is, after the execution of the first **p++** statement **p** value becomes **0X0014**, which is the address of second element of the array. Similarly, after the execution of the second **p++** statement, **p** will be pointing to third element of the array, which is 3. Thus, **p** value becomes **0x0018**.

Similarly, the following code contains dynamically created character array for which **q** is pointer. That is, **q** value is the address of the first element of the array. That is, we assume **q** value **0X0007**. Now, after the execution of first **q++** statement, **q** will be pointing to the second element of the array. That is, **q** value becomes **0X0008** as character takes 1 byte. Similarly, after execution of second **q++** statement, **q** value becomes **0X0009** which is the address of the third element of the string.

Thus, as **p** is integer pointer, its value will be incremented by 4 after increment operator. Similarly, as **q** is character pointer, after applying increment operators its value is incrementing by one. This is applicable for other type of pointer also.

■ **Example** What is the difference between string variables a and p in the following program. Will this program gets compiled? Will it be executed?

```
#include<stdio.h>
int main(){
   char a[]="string", *p="string";
   printf("\n%s  %s\n", a, p);
a++;
p++;
printf("%s  %s\n", a, p);
return 0;
 }
```

■ **Answer:** **a** is a constant pointer whereas **p** is not.

Memory allocated for array **a** is in initialised data segment, where as **p** is in un-initialised data segment.

Program will not be compiled; error will appear on **a++** line as **a** is constant pointer.

■ **Example** What will be the output of the following program.

```
char a[]="Rama",  *p="Rams";
printf("%c %c", 1 [a], 1 [p]);
```

■ **Answer:** Displays a a. Index value followed by array name in square brackets is also. Acceptable way of accessing array elements.

■ **Example** What is the difference between string variables a and p in the following program.

```
char *a="Rama", *p;
p=(char *) malloc(10);
strcpy(p,"Rama");
printf("\n%s  %s\n", a, p);
printf("%p  %p\n", a, p);
```

■ **Answer:** Memory allocated for array **a** is un-initialised data segment, where as **p** is in stack segment.

■ **Example** Does the following program fragment compiles and runs?

```
int *a={1,2,2,2,2}, *p;
p=(int *) malloc(10);
printf("%p  %p\n", a, p);
```

■ **Answer :** No. Memory cannot be created for the first array.

Dangling Memory

The memory which is allocated at our request but we are unable to access the same. This is a logical mistake which is commonly occurs in functions. For example, while we are in a function, we may allocate arrays dynamically and use them and while we leave the function the allocated memory will not be de-allocated leading to dangling memory problem. In our account, memory is allocated, but the same can not be accessed in callee function or program as we do not return any information about the allocated memory to callee function.

Dangling Pointer

It is also a pointer, which points to a memory which is not currently in our account. Any attempt to access the information in the memory pointed by this type of pointer will lead to memory segment violation. Usually, this occurs because of user's mistake. After freeing the memory using free() function, yet, the pointer variable will be having its value pointing to previous memory. Unknowingly, if we access that memory through this pointer, we will be facing this difficulty. See the following code.

```
char *p;
p=(char *) malloc(80);
scanf("%s", p):
printf("Address of the string %p and string is %s", p, p);
free(p);  /* Memory id freed*/
printf("Address of the string %p and string is %s", p, p);
/* Trying to access after de-allocation*/
```

Note

On some compilers, we will not have any difficulty with the above code.

Precedence of Operators on pointers and Arithmetic of pointers

Conducting arithmetical operations on pointers is little different than conducting them on other integer data types. To begin, only addition and subtraction operations are allowed to be conducted, the others make no sense in the world of pointers. But, both addition and subtraction have a different behaviour with pointers according to the size of the data type

to which they point to. When we saw the different data types that exist, we saw that some occupy more or less space than others in the memory. For example, in the case of integer numbers, char occupies 1 byte, short occupies 2 bytes and long occupies 4. Let's suppose that we have 3 pointers:

```
char *mychar;
short *myshort;
long *mylong;
```

and that we know that they point to memory locations 1000, 2000 and 3000, respectively.
So if we write:

```
mychar++;
myshort++;
mylong++;
```

mychar, as you may expect, would contain the value 1001. Nevertheless, **myshort** would contain the value 2002, and **mylong** would contain 3004. The reason is that when adding 1 to a pointer we are making it to point to the following element of the same type with which it has been defined, and therefore the size in bytes of the type pointed is added to the pointer.

This is applicable both when adding and subtracting any number to a pointer. It would happen exactly the same with the following also

```
mychar = mychar + 1;
myshort = myshort + 1;
mylong = mylong + 1;
```

It may result important to warn you that both increase (++) and decrease (--) operators have a greater priority than the reference operator asterisk (*), therefore the following expressions may lead to confusion:

```
*p++;
*p++ = *q++;
```

The first one is equivalent to ***(p++)** and what it does is to increase p (the address where it points to - not the value that contains). The second, because both increase operators (++) are after the expressions to be evaluated and not before, first the value of ***q** is assigned to ***p** and then they are both **q** and **p** increased by one. It is equivalent to:

```
*p = *q;
p++;
q++;
```

Like always, we recommend to use of parenthesis () in order to avoid unexpected results.

For the operators ++, --, and the pointer reference for indirection *, precedence is from right to left. When one of those operators is detected by the compiler, the compiler will determine whether indirection is required. The following cases using a pointer to an integer help to illustrate this point:
Given the following definitions:

```
int  *p, i;
i=99; p=&i;
```

The compiler determines the operation to be performed:

***p**	Access the contents of i via the pointer p
++p	Increment p to point to the next integer
p	Access the address stored in p
++*p	Increment the contents of i via the pointer p before referencing
***++p**	Increment p to point to the next integer, then reference the integer.
***p++**	Increment p to point to the next integer, after the current integer pointed to by p is referenced.
p++*	ILLEGAL unless * is used to multiply the pointer value
p*++	ILLEGAL

■ **Example** The code fragment is to demonstrate the above concepts.

```
int i, j, k, *p;
static int array[]= { 1,2,3 };
j=999; k=1234; i=9; p=&i;
printf("%d\n",i++);  /* prints: 9 */
printf("%d\n",*p);   /* prints: 10 */
printf("%d\n",++*p); /* prints: 11 */
printf("%d\n",*++p);
/* prints: 999, First, p gets incremented to point
     to the next integer. This just happens to be j since j is defined after i. Then the contents of
     integer j would be passed to printf(). This is the makings of a major bug. This is shown only to
     illustrate how a pointer could be misused without the compiler catching the problem. */
printf("%d\n",*p++);
```

/* prints: 999, p gets incremented after it passes 999 */
/* it just so happens that the next integer would be k, since k is defined after j. This would normally create a bug for the programmer that could be difficult to find. */

```
printf("%d\n",*p);
```

/* prints: 1234, Since p was incremented on the
previous line, the pointer was incremented to point
to the next integer. This just happens to be the
integer k - watch out for bugs like these! */

```
/* ERROR printf("%d\n",p++*);  */
/* ERROR printf("%d\n",p*++);  */
```

/ * A better utilisation of pointer arithmetic could be implemented when the contents of an array are accessed.*/

```
p=array;            /*make a pointer from the array ..*/
printf("%d\n",*p);   /* prints: 1  */
printf("%d\n",++*p); /* prints: 2  */
printf("%d\n",*++p); /* prints: 2  */
printf("%d\n",*p++); /* prints: 2  */
printf(«%d\n»,*p);   /* prints: 3  */
```

void pointers

The type of pointer, void is a special type of pointer. void pointers can point to any data type, from an integer value or a float to a string of characters. Its sole limitation is that the pointed data cannot be referenced directly (we cannot use reference asterisk * operator on them), since its length is always undetermined, and for that reason we will always have to resort to type casting or assignations to turn our void pointer to a pointer of a concrete data type that we can refer.
One of its utilities may be for passing generic parameters to a function:

```
#include <stdio.h>
  void increase (void* data, int type){
  switch (type)  {
  case sizeof(char) : (*((char*)data))++; break;
  case sizeof(short): (*((short*)data))++; break;
  case sizeof(long) : (*((long*)data))++; break;
}
```

```
}
int main (){
    char a = 5;
    short b = 9;
    long c = 12;
    increase (&a,sizeof(a));
    increase (&b,sizeof(b));
    increase (&c,sizeof(c));
    printf( "%d %d %d\n", a,b,c);
    return 0;
  }
```

2.1.6.9 Multiple Indirection – Pointers to Pointers

It is possible in C to have a pointer point to another pointer that points to a target value. This is termed multiple indirection in this case double indirection. Multiple indirection can be carried out to whatever extent is desired but can get convoluted if carried to extremes.

In the normal situation, single indirection, a pointer variable would hold the address in memory of an appropriate variable, which could then be accessed indirectly by de-referencing the pointer using the * operator.

In the case of double indirection, we have the situation where a variable may be pointed to by a pointer as with single indirection, but that this pointer may also be pointed to by another pointer. So we have the situation where we must de-reference this latter pointer twice to actually access the variable we are interested in. De-referencing the *pointer to a pointer* once gives us a normal singly indirected pointer, de-referencing the pointer to a pointer secondly allows us to access the actual data variable. The situation is depicted in Fig. 2.2.

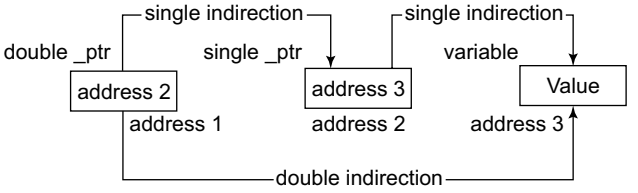


Figure 2.2 Single and double indirection

To declare a pointer to a pointer we include another indirection operator

```
float  ** ptr ;
```

which in this case defines a *pointer to a pointer to type float*.

The following illustrates some valid operations using double indirection.

```
int x = 10, *p, **q ;
p = &x ;
q = &p ;
**q = 20 ;   // de-reference twice to access value
p = *q ;       // de-reference q once to get a pointer to int
…
int array1[] = { 1,2,3,4,5,6,7,8,9,10} ;
int array2[] = {10,20,30,40,50} ;
```

```
int *pointers[2] ;   // an array of pointers to type int
int **ptr ;            // a doubly indirected pointer
ptr = pointers ;   // initialise pointer to array of pointers
*ptr++ = array1 ; // now we simply de-reference the pointer to a pointer
*ptr = array2 ;        // once and move it on like any pointer
**ptr = 100 ;       // ptr is pointing at pointers[1] which in turn is pointing
                  // at array2 so array2[0] is assigned 100
```

For Example: Allocation and initialisation of an m x n matrix using double indirection.

What we require here is to allocate an n x n matrix as a collection of discrete rows rather than just as one block of memory. This format has advantages over a single block allocation in certain situations. The structure we get is shown in Fig. 2.3.

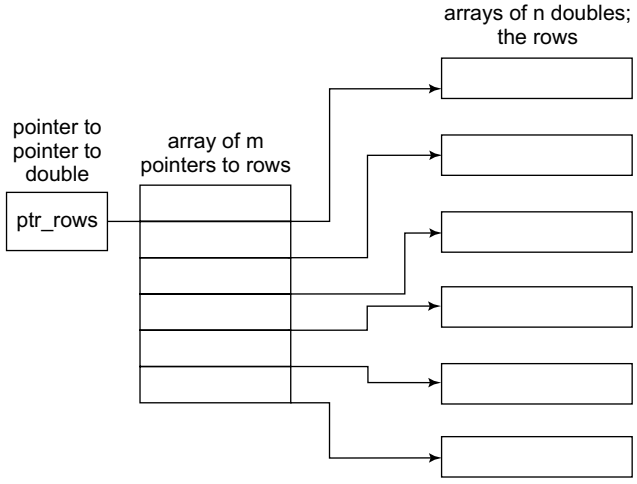


Figure 2.3 Array of Pointers

■ **Example** What is the output for the following program?

```
main(){
int arr2D[3][3];
printf("%d\n", ((arr2D==* arr2D)&&(* arr2D == arr2D[0])) );
}
```

■ **Answer:** This is due to the close relation between the arrays and pointers. N dimensional arrays are made up of (N-1) dimensional arrays. arr2D is made up of a 3 single arrays that contains 3 integers each.

arr2D			
arr2D[1]			
arr2D[2]			
arr2D[3]			

The name arr2D refers to the beginning of all the 3 arrays. *arr2D refers to the start of the first 1D array (of 3 integers) that is the same address as arr2D. So the expression (arr2D == *arr2D) is true (1). Similarly, *arr2D is nothing but *(arr2D + 0), adding a zero doesn't change the value/meaning. Again arr2D[0] is the another way of telling *(arr2D + 0). So the expression (*(arr2D + 0) == arr2D[0]) is true (1). Since both parts of the expression evaluates to true the result is true(1) and the same is printed.

2.1.6.10 Pointers to Functions

A function even though not a variable still has a physical address in memory and this address may be assigned to a pointer. When a function is called it essentially causes an execution jump in the program to the location in memory where the instructions contained in the function are stored so it is possible to call a function using a pointer to a function.

The address of a function is obtained by just using the function name without any parentheses, parameters or return type in much the same way as the name of an array is the address of the array. A pointer to a function is declared as follows

```
return_type (* fptr ) ( parameter list ) ;
```

where fptr is declared to be a pointer to a function which takes parameters of the form indicated in the parameter list and returns a value of type return_type.

The parentheses around * *fptr* are required because without them the declaration

```
return_type  * fptr( parameter list ) ;
```

just declares a function fptr which returns a pointer to type return_type !

To assign a function to a pointer we might simply do the following

```
int (*fptr)( ) ;
fptr = getchar ;  /* standard library function */
```

To call the function using a pointer we can do either of the following

```
ch = (*fptr)( ) ;
ch = fptr( ) ;
```

■ **Example** Which of the following solution are better in assigning 0s to an array elements.

a.
```
for ( k = 0; k < 100; k++ )
array[ k ] = 0.0 ;
```
b.
```
ptr = array ;
for ( k = 0; k < 100; k++ )
*( ptr + k ) = 0.0 ;
```
c.
```
ptr = array ;
for ( k = 0; k < 100; k++ )
*ptr++ = 0.0 ;
```

■ **Answer:** Option C is better answer in this case we just incur the addition of size of (double) onto the address contained in the pointer variable for each iteration.

■ **Example** What will be the output of the following program? Explain.

```
#include<stdio.h>
int main()
{
 int *p, i=10;
 p=&i;
 printf("\n%d %d\n",*p, *&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&i);
 return 0;
 }
```

■ **Answer:** We will be getting output as: 10 10. Here, &i is the address of the variable i. If we apply * for &i, we are referring to i itself. Similarly, & to *&i indicates address of i while * of &*&i refers to i itself. Here we have used & and * operators recursively. Thus, entire expression conveys ***&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&i** the value of i only.

■ **Example** What is the value of
&&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*p
in the above program?

■ **Answer:** It refers to i assuming i address is assigned to pointer variable p. As p is pointer to i, *p refers to i, while &*p refers to address of i. Also, *&*p refers to i. Thus, * followed by a series of &* indicates i.

■ **Example** What is going to happen when we try to run the following program?

```
#include<stdio.h>
int main()
   {
            int Num;
            printf("Enter a Number\n");
    scanf("%d", Num);
    printf("Value of the number l=%d\n", Num);
    return 0;
    }
```

■ **Answer:** In Unix, when we run the above program, we will get core dumped error message and program will be terminated. If we observe the scanf statement, we have not used & to **Num** variable. Actually we have to send format string and a list of addresses to scanf() statement such that it reads values and stores in the given addresses. However, here in this example we did not use address (&) operator to **Num** variable in scanf() function. Thus, scanf() understands that the value read has to be stored in the memory which is pointed by the value of **Num** variable which is trash (garbage) as it is un-initialised. By doing so, we are violating the rules and regulations of operating system thus we will get the error message and then the program will be terminated abnormally.

Note

Same program may run without any runtime error under Turbo C compiler!!!

Some important notations and representations about pointers are summarised below.

int *p;	/* p can be a pointer to an integer quantity or to an integer array */
int *p[10] ;	/* p is a 10 element array of pointers to integer quantities /*
int (*p)[10];	/* p is a pointer to a 10-element integer array /*
int *p(void);	/* p is a function that returns a pointer to an integer quantity /*
int p(char *a);	/* p is a function that accepts an argument which is a pointer to a character returns an integer quantity */
int *p(char *a);	/* p is a function that accepts an argument which is a pointer to a character returns a pointer to an integer quantity */
int (*p)(char *a);	/* p is a pointer to a function that accepts an argument which is a pointer to a character returns an integer quantity */
int (*p(char *a))[10]	/* p is a function that accepts an argument which is a pointer to a character returns a pointer to a 10 element integer array */
int p(char (*a)[]);	/* p is a function that accepts an argument which is a pointer to a character array returns an integer quantity */
int p(char *a[]) ;	/* p is a function that accepts an argument which is an array of pointers to characters returns an integer quantity */
int *p(char a[]);	/* p is a function that accepts an argument which is a character array returns a pointer to an integer quantity */

int *p(char (*a)[]); /* p is a function that accepts an argument which is a pointer to a character array returns a pointer to an integer quantity */

int *p(char *a[]); /* p is a function that accepts an argument which is an array of pointers to characters returns a pointer to an integer quantity */

int (*p)(char (*a)[]); /* p is a pointer to a function that accepts an argument which is a pointer to a character array returns an integer quantity */

int *(*p)(char (*a)[]); /* p is a pointer to a function that accepts an argument which is a pointer to a character array returns a pointer to an integer quantity */

int *(*p)(char *a[]); /* p is a pointer to a function that accepts an argument which is an array of pointers to characters returns a pointer to an integer quantity */

int (*p[10])(void); /* p is a 10 element array of pointers to functions;
Each function returns an integer quantity */

int (*p[10])(char a); /* p is a 10 element array of pointers to functions each functions
Accepts an argument which is a character, and returns an integer quantity */

int *(*p[10])(char a); /* p is a 10 element array of pointers to functions, each function accepts an argument which is a character, and returns a pointer to an integer quantity */

int *(*p[10])(char *a); /* p is a 10 element array of pointers to functions, each function accepts an argument which is a pointer to a character, and returns a pointer to an integer quantity */

■ **Example** Explain the purpose of each of the following declarations

(a) float (*x) (int *a);
(b) float (*x(int *a))[20];
(c) float x(int (*a)[]);
(d) float x(int *a[]);
(e) float *x(int a[]);
(f) float *x(int(*a)[]);
(g) float *x(int *a[]);
(h) float (*x)(int (*a)[]);
(i) float *(*x)(int *a[]);
(j) float (*x[20])(int a);
(k) float *(*x[20](int *a);

■ **Answers:**

(a) **x** is a pointer to a function that accepts an argument which is a pointer to an integer quantity and returns a floating-point quantity.
(b) **x** is a function that accepts an argument which is a pointer to an integer quantity and returns a pointer to a 20 element floating-point array.
(c) **x** is a function that accepts an argument which is a pointer to an integer array and returns a floating point quantity.
(d) **x** is a function that accepts an argument which is an array of pointer to integer quantities and returns a floating point quantity.
(e) **x** is a function that accepts an argument which is an integer array and returns a pointer to a floating point quantity.
(f) **x** is a function that accepts an argument which is a pointer to an integer array and returns a pointer to a floating point quantity.
(g) **x** is a function that accepts an argument which is an array pointers to integer quantities and returns pointer to a floating point quantity.
(h) **x** is a pointer to a function that accepts an argument which is a pointer to an integer array and returns a floating point quantity.
(i) **x** is a pointer to a function that accepts an argument which is an array of pointers to integer quantities and returns a pointer to a floating point quantity.

(j) **x** is a 20 element array of pointers to functions; each function accepts an argument which is an integer quantity and returns a floating-point quantity.

(k) **x** is a 20 element array of pointers to functions; each function accepts an argument which is a pointer to an integer quantity and returns a pointer to a floating-point quantity.

■ **Example** Write an appropriate declaration for the following situations involving pointers.

(a) Declare a function that accepts an argument which is a pointer to an integer quantity and returns a pointer to a six-element character array.

(b) Declare a function that accepts an argument which is a pointer to an Integer array and returns a character.

(c) Declare a function that accepts an argument which is an array of pointer to integer quantities and returns a character.

(d) Declare a function that accepts an argument which is an integer array and returns a pointer to a character.

(e) Declare a function that accepts an argument which is a pointer to an Integer array and returns a pointer to a character.

(f) Declare a function that accepts an argument which is an array of pointers to integer quantities and returns a pointer to a character.

(g) Declare a pointer to a function that accepts an argument which is a pointer to an Integer array and returns a character.

(h) Declare a pointer to a function that accepts an argument which is a pointer to an integer array and returns a pointer to a character.

(i) Declare a pointer to a function that accepts an argument which is an array of pointers to Integer quantities and returns a pointer to a character.

(j) Declare a 12 element array of pointers to functions. Each function will accept two double-precision quantities as arguments and will return a double-precision quantity.

(k) Declare a 12 element array of pointers to functions. Each function will accept two double-precision quantities as arguments and will return a pointer to a double-precision quantity.

(l) Declare a 12 element array of pointers to functions. Each function will accept two pointers to double-precision quantities as arguments and will return a pointer to a double-precision quantity.

■ **Answers:**

(a) char (*p(int *a))[6];

(b) char p(int (*a)[]);

(c) char p(int *a[]);

(d) char *p(int a[]);

(e) char *p(int (*a)[]);

(f) char *p(int *a[]);

(g) char (*p)(int (*a)[]);

(h) char *(*p)(int (*a)[]);

(i) char *(*p)(int *a[]);

(j) double (*f [12])(double a, double b);

(k) double *(*f [12])(double a, double b);

(m) double *(*f [12])(double *a, double *b);

■ **Example** What is the difference between arrays and pointers?

■ **Answer:** Pointers are used to manipulate data using the address. Pointers use * operator to access the data pointed by them. Arrays use subscripted variables to access and manipulate data. Array variables can be equivalently written using pointer expression. Name of array is considered as constant pointer. Like normal pointers, we cannot apply ++,-- operators on them.

■ **Example** What is the purpose of **realloc( )**?

■ **Answer:** To adjust the memory of an dynamic array. The function **realloc(ptr,n)** uses two arguments; first argument ptr is a pointer to a block of memory for which the size is to be altered. The second argument n specifies the new size. The size may be increased or decreased. If n is greater than the old size and if sufficient space is not available subsequent to the old region, the function **realloc( )** may create a new region and all the old data are moved to the new region. Do test the returned value from the **realloc()** before doing any further work.

■ **Example** What is dynamic and static memory allocation?

■ **Answer:** Usually memory is allocated for variables during compilation time itself in the data area of the program. Assigning address of this type of variables to a pointer variable is known as static memory allocation. memory is assigned during compilation time. Dynamic memory allocation uses functions such as **malloc( )** or **calloc()** to get memory dynamically. If these functions are used to get memory dynamically and the values returned by these functions are assigned to pointer variables, such assignments are known as dynamic memory allocation. Memory is assigned during run time. This memory is allocated from heap portion of the memory.

■ **Example** Which are called as stack variables?

■ **Answer:** Memory which is allocated during a function call can be called as stack variables as for them memory is allocated from run-time stack.

■ **Example** How are pointer variables initialised? Can we assign arbitrary number to a pointer variable?

■ **Answer:** Pointer variable are initialised by one of the following two ways 1. Static memory allocation 2. Dynamic memory allocation. We cannot assign arbitrary number to a pointer variable except 0.

■ **Example** Are pointers integers?

■ **Answer:** No, pointers are not integers. A pointer is an address. We know addresses of a system starts from 0 and increases 2^{word}-1, where word is the word size of the computer. Thus pointer is merely a positive number and not an integer.

■ **Example** What is a pointer variable? How does it differ from other variables?

■ **Answer:** A pointer variable is a variable that may contain the address of another variable or any valid address (allocated address) in the memory. That is, for a pointer variable one can assign allocated memory cell number only. In today's, 32-bit operating systems, 4 bytes of memory is allocated for any type of pointer.

■ **Example** What is a pointer value and address?

■ **Answer:** A pointer value is a data object that is in the memory which is referred by the pointer variable. Each memory location is numbered in the memory. The number attached to a memory location is called the address of the location.

■ **Example** Is a pointer variable is an acceptable **lvalue**?

■ **Answer:** Yes.

■ **Example** Is it possible to compare pointers?

■ **Answer:** Yes. If they belong to same chunk of memory, comparison is meaningful.

2.1.7 Structures

A structure is a customised user-defined data type in C. It is by definition a collection of variables (elements) of any type that are referenced under one name, providing a convenient means of keeping related information together.
Defining Structures: We declare a structure like the following manner

```
struct  tag   {
          type  var_1 ;
          type var_2 ;
          ...
          type var_n ;
          }  ;
```

The keyword **struct** tells the compiler that we are dealing with a structure and must be present whenever we refer to the new type, **tag** is an identifier which is the name given to the customised "type". A variable of this new type can now be

defined as follows for example. Note that the keyword struct has to be used in conjunction with our own name for the structure, *tag*.

```
struct tag ABC ;
```

For Example, in the following we are declaring a structure to store a person's details along with the declaration of such a variable.

```
struct RECORD {
        int rollno ;
        char name[30] ;
        char town[40] ;
        char country[ 20 ]
        } ;
struct RECORD RAM ;
```

The compiler will automatically allocate enough storage to accommodate all the elements. To find out how much storage is required one might do the following

```
size = sizeof ( RAM ) ;
```

or

```
size = sizeof( struct RECORD );
```

Note

The name of a structure is not the address of the structure as with array names.

Accessing Structure Elements

The elements of the structure are accessed using the **dot operator, . ,** as follows

```
RAM.rollno  =  109 ;
scanf ( "%s", RAM.name ) ;
```

Thus we treat structure elements exactly as normal variables and view the dot operator as just another appendage like the indirection operator or an array index.

Initialising Structures

Structure elements or fields can be initialised to specific values as follows:

```
struct id {
        char name[30] ;
        int rno ;
        } ;
struct id student = { "Govindu", 4563 } ;
```

Structure Assignment

The name of a structure variable can be used on its own to reference the complete structure. So instead of having to assign all structure element values separately, a single assignment statement may be used to assign the values of one structure to another structure of the same type. For example:

```
struct {
int a, b ;
}  x = {1, 2 }, y ;
y = x ;   //assigns values of all fields in x to fields in y
```

Creating More Complex Structures with Structures

Once again emphasising that structures are just like any other type in C we can create arrays of structures, nest structures, pass structures as arguments to functions, etc.

For example we can nest structures as follows creating a structure employee that has another structure as one of its members.

```
struct time {
        int hour ;
        int min ;
        int sec ;
} ;
struct employee{
        char name[30] ;
        struct time start, finish ;
        } Abhi;
```

To access the hour field of time in the variable Abhi, we have to just apply the dot operator twice

```
Abhi.start.hour = 9 ;
```

If a company needs to keep track of more than one employee, then an array of *employee* would be useful.

```
struct employee workers[100] ;
```

To access specific employees we simply index using square braces as normal, e.g. workers[10]. To access specific members of this structure we simply apply the dot operator on top of the index.

```
workers[10].finish.hour = 10 ;
```

When structures or arrays of structures are not global they must be passed to functions as parameters subject to the usual rules. For example

```
function1( Abhi ) ;
```

implements a call to function1 which might be prototyped as follows

```
void  function1( struct employee emp );
```

Note that a full local copy of the structure passed is made so if a large structure is involved memory the overhead to simply copy the parameter will be high so we should employ call by reference instead as we will see in the next section.

Passing an array of structures to a function also follows the normal rules but note that in this case as it is impossible to pass an array by value no heavy initialisation penalty is paid - we essentially have call by reference. For example

```
function2( workers );
```

passes an array of structures to *function2* where the function is prototyped as follows.

```
function2( struct employee staff[ ] ) ;
```

2.1.7.1 Structure Pointers

As we have said already we need call by reference calls which are much more efficient than normal call by value calls when passing structures as parameters. This applies even if we do not intend the function to change the structure argument. A structure pointer is declared in the same way as any pointer for example

```
struct address {
        char name[20] ;
        char street[20] ;
        } ;
struct address person ;
struct address *ptr ;
```

declares a pointer ptr to data type *struct address*.
To point to the variable person declared above we simply write

```
ptr =  &person ;
```

which assigns the address of person to ptr.
To access the elements using a pointer we need a new operator called the arrow operator or de-referencing operator, ->, which can be used **only** with structure pointers. For example

```
ptr -> name;
```

Note that even though we are not changing any values in the structure variable we still employ call by reference for speed and efficiency.

2.1.7.2 Dynamic Allocation of Structures

The memory allocation functions may also be used to allocate memory for user defined types such as structures. All malloc() basically needs to know is how much memory to reserve. For example:

```
struct coordinate {
          int x, y, z ;
          } ;
  struct coordinate *ptr ;
ptr=(struct coordinate*)malloc(sizeof(struct coordinate ));
To allocate an array of ten coordinate type of variables:
struct coordinate *ptr ;
ptr= (struct coordinate*)malloc(10*sizeof(struct coordinate));
```

Here, ptr[0], ptr[1], etc refers to coordinate type variables while ptr[0].x, ptr[0].y, etc refers to data members of then. Here, ptr[i].x, (*(ptr+i)).x, (ptr+i)->x refers to same where i is in integer.

2.1.7.3 Bit--Fields

Bit--fields are based directly on structures with the additional feature of allowing the programmer to specify the size of each of the elements in bits to keep storage requirements at a minimum. However bit--field elements are restricted to be of type int (signed or unsigned). For example:

```
struct clock {
        unsigned hour : 5 ;
        unsigned minutes : 6 ;
        unsigned seconds : 6 ;
        } time ;
```

This time structure requires 17 bits to store the information now so the storage requirement is rounded up to 3 bytes. Using the normal structure format and 32-bit integer elements we would require 12 bytes so we achieve a substantial saving.

Bit-fields can be used instead of the bitwise operators in system level programming, for example to analyse the individual bits of values read from a hardware port we might define the following bit-field.

```
struct status {
        unsigned bit0 : 1 ;
        unsigned bit1 : 1 ;
        ...
        unsigned bit15 : 1 ;
        } ;
```

If we are interested in bit 15 only we need only do the following

```
struct status {
        unsigned : 15 ;
/* skip over first 15 bits with a "non-member"  */
```

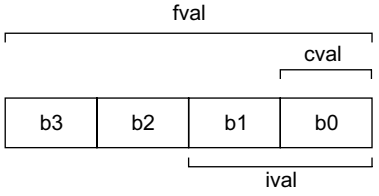
```
unsigned bit15 : 1 ;
} ;
```

There are two further restrictions on the use of bit--fields

- Cannot take the address of a bit--field variable
- Cannot create an array of bit--field variables

2.1.7.4 Unions

A union is data type where the data area is shared by two or more members generally of different type at different times. For example:




```
union u_tag {
   short ival ;
   float fval ;
   char cval ;
   } uval ;
```

The size of uval will be the size required to store the largest single member, 4 bytes in this case to accommodate the floating point member.

Union members are accessed in the same way as structure members and union pointers are valid.

```
uval.ival = 10 ;
uval.cval = 'c' ;
```

When the union is accessed as a character we are only using the bottom byte of storage, when it is accessed as a short integer the bottom two bytes, etc. It is up to the programmer to ensure that the element accessed contains a meaningful value.

A union might be used in conjunction with the bit-field *struct status* in the previous section to implement binary conversions in C. For Example:

```
union conversion {
unsigned short num ;
        struct status bits ;
} number ;
```

We can load number with an integer

```
scanf( "%u", &number.num );
```

Since the integer and bit--field elements of the union share the same storage if we now access the union as the bit--field variable *bits* we can interpret the binary representation of num directly.

i.e.

```
if ( uvar.bits.bit15 )
putchar( '1' ) ;
else
putchar('0') ;
...
if ( uvar.bits.bit0 )
putchar( '1' ) ;
else
putchar('0') ;
```

Admittedly rather inefficient and inelegant but effective.

2.1.7.5 Enumerations

An enumeration is a user defined data type whose values consist of a set of named integer constants, and are used for the sole purpose of making program code more readable.

```
enum tag { value_list } [ enum_var ] ;
```

where tag is the name of the enumeration type, value_list is a list of valid values for the enumeration, and where enum_var is an actual variable of this type. For example:

```
enum colours { red, green, blue, orange } shade;
  //  values red - 0, green - 1, blue - 2, orange - 3
enum day { sun = 1, mon, tue, wed = 21, thur, fri, sat } ;
  enum day weekday ;
//values of sun, mon, tue, etc  are 1, 2, 3, 21, 22, 23, 24
```

Variables declared as enumerated types are treated exactly as normal variables in use and are converted to integers in any expressions in which they are used. For example:

```
int i ;
shade = red ;// assign a value to shade enum variable
i = shade ;// assign value of enum to an int
shade = 3 ;// assign valid int to an enum, treat with care
```

2.1.7.6 The typedef Keyword

C makes use of the **typedef** keyword to allow new data type **names** to be defined. No new type is created, an existing type will now simply be recognised by another name as well. The existing type can be one of the in-built types or a user-defined type.

```
typedef  type  name ;
```

where type is any C data type and name is the new name for this type. For example:

```
typedef  int INTEGER ;
INTEGER i ;
```

// can now declare a variable of type 'INTEGER'

```
        typedef double * double_ptr ;
        double_ptr  ptr ;
//  no need of * here as it is part of the type
        typedef struct coords {
                int x, y ;
                } xycoord ;
//  xycoord is now a type name in C
        xycoord  coord_var ;
```

The use of typedef makes program code easier to read and when used intelligently can facilitate the porting of code to a different platform and the modification of code. For example, in a first attempt at a particular program we might decide that floating point variables will fill our needs. At a later date we decide that all floating point variables really should be of type double so we have to change them all. This problem is trivial if we had used a typedef as follows:

```
typedef float FLOATING ;
```

To remedy the situation we modify the user defined type as follows

```
typedef double FLOATING ;
```

2.1.7.7 Nodes or Self-Referential Structures

A structure that contains a pointer member that points to a structure of the same type as itself is said to be a self-referential structure. For example:

```
struct node {
   char name[20] ;
   struct node *next ;
};
```

This defines a new type, **struct node,** which has a pointer member, next, which points to a structure of the same type as itself. This node pointer allows this current node to be linked to another and so a list of linked nodes can be built up.

Note that even though the structure is not fully defined when the next field is specified the compiler does not have a problem as all it needs to know is that there is such a type.

The above definition of struct node allows us to build a singly linked list, i.e. each node only knows where the node following it is located. A null pointer is used to indicate the end of a linked list structure.

Figure 2.4 illustrates how this node structure would be used to represent our linked list above. Each node contains two fields the actual data and a pointer link to the next node. The final node in the list contains a null pointer link as indicated by the slash in the diagram. The following code fragment may create this list.

```
struct node A, B, C, D;
strcpy(A.name,"Ram");
A.next=&B;
strcpy(B.name,"Ravi");
B.next=&C;
strcpy(C.name,"Abhi");
C.next=&D;
strcpy(D.name,"Sai");
D.next=0;
```

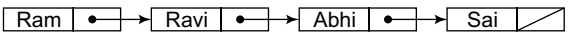


Figure 2.4

2.1.8 File I/O

The C I/O system provides a consistent interface known as **file stream** in C to work with devices such as hard disk, tape drive, the screen, printer port, etc. All I/O through the keyboard and screen that we have employed so far is in fact done through special standard streams called **stdin** and **stdout** for input and output, respectively. So in essence the console functions that we have used so far such as printf(), etc. are special case versions of the file functions. There are two types of streams : text and binary. These streams are basically the same in that all types of data can be transferred through them however there is one important difference between them which we will elucidate now.

Text Streams

A text stream is simply a sequence of characters. However, the characters in the stream are open to translation or interpretation by the host environment. For example, the newline character, '\n', will normally be converted into a carriage return/linefeed pair and ^Z will be interpreted as EOF. Thus the number of characters sent may not equal the number of characters received.

Binary Streams

A binary stream is a sequence of data comprised of bytes that will not be interfered with so that a one-to-one relationship is maintained between data sent and data received.

2.1.8.1 Opening and Closing Files: File Pointers

A stream will be associated with a specific file by performing an open operation on the file. Once a file is opened, information can be exchanged between it and our program. An opened file will have a unique file control structure of type FILE. A **file pointer (or file stream pointer)** is a pointer to this FILE structure which identifies a specific file and defines various things about the file including its name, read/write status, and current position. A file pointer variable is defined as follows

```
FILE  *fptr ;
```

The fopen() function whose prototype is given below opens a stream for use and links a file with that stream returning a valid file pointer which is positioned correctly within the file if all is correct.

```
FILE *fopen( const char *filename, const char *mode );
```

Here first argument is filename and second argument is mode string which indicates in which mode the file has to be opened. Table 2.3 lists acceptable values for mode string.

Table 2.3

r	opens a text file for reading (must exist)
w	opens a text file for writing (overwritten or created)
a	append to a text file
rb	opens a binary file for reading
wb	opens a binary file for writing
ab	appends to a binary file
r+	opens a text file for read/write (must exist)
w+	opens a text file for read/write
a+	append a text file for read/write
rb+	opens a binary file for read/write
wb+	opens a binary file for read/write
ab+	append a binary file for read/write

If fopen( ) cannot open the requested file in the given mode, it will a return a NULL pointer.

The fclose() function is used to disassociate a file from a stream and free the stream for use again. The fclose() function needs file pointer as argument.

```
fclose( fptr ) ;
```

fclose() will automatically flush any data remaining in the data buffers to the file.

2.1.8.2 Formatted Reading and Writing

We can use fprintf(), fscanf() functions which are akin to printf(), scanf() functions with the exception that they will be taking file stream pointer as an additions first argument. Prototypes of these functions can be given as:
fprintf(FILE *, format string, list of variables or expressions whose values to be printed);
fscanf(FILE *, format string, list of addresses into which values to be stored);

Reading & Writing Characters

Once a file pointer has been linked to a file we can write characters to it using the fputc() function.

```
fputc( ch, fp ) ;
```

If successful the function returns the character written otherwise EOF. Characters may be read from a file using the fgetc() standard library function.

```
ch = fgetc( fp ) ;
```

When EOF is reached in the file fgetc( ) returns the EOF character which informs us to stop reading as there is nothing more left in the file.

Working with strings of text

To read and write a string, we can use fputs(), fgets() functions whose prototypes are given as:

```
int fputs( const char *str, FILE *fp ) ;
char *fgets( char *str, int maxlen, FILE *fp );
```

Note

There are several I/O streams opened automatically at the start of every C program. They are:

stdin	---	standard input i.e. keyboard
stdout	---	standard output i.e. screen
stderr	---	again the screen for use if stdout malfunctions

It is through these streams that the console functions we normally use operate. For example in reality a normal printf call such as

```
printf( "%s %d", s, t ) ;
```

is in fact interpreted as

```
fprintf( stdout, "%s %d", s, t ) ;
```

2.1.8.3 Binary reading using fread() and fwrite()

These two functions are used to read and write blocks of data of any type. Their prototypes are as follows where size_t is equivalent to unsigned.

```
size_t  fread( void *buffer,  size_t num_bytes,  size_t count,  FILE *fp ) ;
size_t  fwrite( const void *buffer,  size_t num_bytes,  size_t count,  FILE *fp ) ;
```

where **buffer** is a pointer to the region in memory from which the data is to be read or written respectively, **num_bytes** is the number of bytes in each item to be read or written, and **count** is the total number of items (each num_bytes long) to be read/written. The functions return the number of items successfully read or written.

Note

Unlike all the other functions we have encountered so far fread and fwrite read and write **binary** data in the same format as it is stored in memory so if we try to edit one these files it will appear completely garbled. Functions like fprintf, fgets, etc. read and write displayable data. fprintf will write a double as a series of digits while fwrite will transfer the contents of the 8 bytes of memory where the double is stored directly.

Random Access I/O

The fseek() function is used in C to perform random access I/O and has the following prototype.

```
int fseek ( FILE *fp, long num_bytes, int origin ) ;
```

where **origin** specifies one of the following positions as the origin in the operation

SEEK_SET	---	beginning of file
SEEK_CUR	---	current position
SEEK_END	---	EOF

and where **num_bytes** is the offset in bytes to the required position in the file. fseek() returns zero when successful, otherwise a non-zero value.

For example, if we had opened a file which stored an array of integers and we wish to read the 50th value we might do the following

```
fseek ( fp, ( 49 * sizeof( int ) ), SEEK_SET ) ;
fscanf( fp, "%d", &i ) ;
```

from anywhere in the program.

2.1.8.4 Low -- Level I/O

We can also use system calls to do file operations. For instance, Low level I/O makes use of a file handle or descriptor, which is just a non-negative integer, to uniquely identify a file instead of using a pointer to the FILE structure as in the case of stream I/O. As in the case of stream I/O a number of standard files are opened automatically:

standard input --- 0

standard output --- 1

standard error --- 2

The following table lists some of the more common low level I/O functions, whose prototypes are given in <io.h> and some associated constants are contained in <fcntl.h> and <sys\stat.h>.

open()	opens a disk file
close()	closes a disk file
read()	reads a buffer of data from disk
write()	writes a buffer of data to disk

The open function has the following prototype and returns -1 if the open operation fails.

```
int open ( char *filename, int oflag [, int pmode] ) ;
```

where filename is the name of the file to be opened, oflag specifies the type of operations that are to be allowed on the file, and pmode specifies how a file is to be created if it does not exist.

oflag may be any logical combination of the following constants which are just *bit flags* combined using the bitwise OR operator.

O_APPEND	appends to end of file
O_BINARY	binary mode
O_CREAT	creates a new file if it doesn't exist
O_RDONLY	read only access
O_RDWR	read write access
O_TEXT	text mode
O_TRUNC	truncates file to zero length
O_WRONLY	write only access

pmode is only used when O_CREAT is specified as part of oflag and may be one of the following values

```
S_IWRITE

S_IREAD

S_IREAD | S_IWRITE
```

This will actually set the read / write access permission of the file at the operating system level permanently unlike oflag which specifies read / write access just while your program uses the file.

The close() function has the following prototype

```
int close ( int handle ) ;
```

and closes the file associated with the specific handle.

The read() and write() functions have the following prototypes

```
int read( int handle, void *buffer, unsigned int count ) ;

int write( int handle, void *buffer, unsigned int count ) ;
```

where handle refers to a specific file opened with open(), buffer is the storage location for the data (of any type) and count is the maximum number of bytes to be read in the case of read() or the maximum number of bytes written in the case of

write(). The function returns the number of bytes actually read or written or -1 if an error occurred during the operation.

Low level file I/O also provides a seek function **lseek** with the following prototype.

```
long _lseek( int handle, long offset, int origin );
```

_lseek uses the same origin, etc. as *fseek()* but unlike fseek() returns the offset, in bytes, of the new file position from the beginning of the file or -1 if an error occurs.

■ **Example** To determine the size in bytes of a file, we can simply open the file and call lseek to end of file before calling ftell()

```
handle=open( name,O_BINARY| O_RDONLY, S_IREAD | S_IWRITE );
lseek(  handle, 0L, SEEK_SET ) ;
length = lseek( handle, 0L, SEEK_END ) ; printf( "The length of %s is %ld bytes \n", name, length ) ;
```

2.2 Solved Questions

1. The following recursive function is proposed to calculate (n+2)/n!. Is it correct way of implementation? Do you get correct results? For which values of n, it is going to give correct results?

```
float sum(int n){
if(n<=1) return 1;
else
return ((n+2)/(n*sum(n-1)));
}
```

Answer: No.

We are supposed to get 3, 2, 5/6, 6/24, etc for n values of 1, 2, 3, 4 . See the snap shot of the recursive call n value of 4.

First call	Second call	Third call	Fourth call
n = 4 return((6)/(4*sum(3))	n = 3 return ((5)/(3*sum(2))	n = 2 return((4)/(2*sum(1))	n = 1 return 1
return(6/(4*5/6))	return(5/(3*2))	return(4/(2*1))	

You may find the from the above snap shot for n values of 1, 2 and 3 it is giving correct results while for n value of 4, it is giving wrong value.

2. How many times, we get the message "Hi" from the following code fragment?

```
int i,j,k,l=0, n=5;
for(i=0;i<n;i++)
for(j=0;j<n;j++)
for(k=0;k<n;k++,l++) if(l==100)goto PRINT;
PRINT:
 printf("Hi\n");
```

Answer: Only once. The variable l value is getting incremented by 1 each time. The if condition will be executed each time. When l value becomes 100, loop control goes to PRINT label (That is, out of all the loops) and executes printf statement. Thus, we get the message only once. By this time inner most loop might have executed 100 times.

3. How many times "Hi" message will be printed when we execute the following code fragment? What is the value of l after completing all the loops?

```
int i,j,k,l=0, n=5;
 for(i=0;i<n;i++)
 for(j=0;j<n;j++){
 for(k=0;k<n;k++,l++) if(l%17==0)goto PRINT;
 PRINT:
printf("Hi %d\n",l);
}
```

Answer: 25 times. Initial value of l is 0. Thus, the if condition in inner most loop will be always true and the control comes out of the inner most loop each time and prints "Hi". Never, l++ statement will be executed. Thus, l value will be 0 after completing all the nested loops also.

4. When the following code fragment is executed, we have got the following sequence as output:

1	6	11	16	18	23	28	33
45	50	52	57	62	67	69	74
86	91	96	101	103			

Explain how it is possible.

```
int i,j,k,l=0, n=5;
for(i=0;i<n;i++)
for(j=0;j<n;j++){
for(k=0;k<n;k++) if(l++%17==0)goto PRINT;
PRINT:
printf("%d\t",l);
}
```

Answer: The variable l value will be printed whenever the if condition is true (that is l value is 0, 17, 34, etc multiples of 17) and also whenever control comes out of the inner k loop. That is, whenever k value is 5, l value will be printed. Also, if you observe the if condition, we find l value will be incremented in postfix manner. Thus, when l value is 0, if condition becomes true and automatically l value will be incremented by one. Because of this reason l value 1 is printed first.

5. The following code fragment is executed with interactive input values as 10 60 90, what is the output?

```
int l;
scanf("%d%d%d",&l,&l, &l  );
printf("%d\n", l);
```

Answer: 90. Only last value is assigned to l.

6. What is the output of the following program assuming call-by-reference is used for all variables in procedure P.

```
Program Main(input,output);
Var a,b:integer;
Procedure P(x,y,z:integer);
Begin
   y:=y+1;
   z:=z+x;
   End P;
Begin
   a:=2;  b:=3;
   P(a+b, a,a);
   Write(a);
End.
```

Answer: 6. When function is called from Main, a+b=5 is sent as first argument. Thus,, formal variable x becomes reference to this result. Similarly, formal arguments y and z becomes references to a. After the execution of y=y+1, a value becomes 3. After the execution of z=z+x statement, a value becomes 6.

7. Analyse the following function and program assuming that the arguments of the function are working in passing by reference style.

```
     void  ff(int x, int y){
      x=0;
      y++;
  }
  main(){
  int a[]={10,10,20,20,20}
 ff(a[1],a[1]);
printf("%d\n", a[1]);
}
```

Answer: If we assume passing by reference then both x and y becomes aliases to a[1]. If x is made zero actually a[1] becomes zero. As y is also reference to a[1] its value becomes 1 because of y++ statement. If we assume the method is passing by value-result then x & y values becomes 10 initially. After that x value becomes zero where as y becomes 11. When we return from the function then x value is written into a[1] thus it becomes 1 then y value is written into a[1] thus it becomes 11 (if we assume that sequence).

8. Analyse the following function and program in terms of pass by value style and pass by address style.

```
int a=20;
void fff(int x){
a=1;
x=x+a;
}
     main(){
fff(a);
}
```

Answer: In the case of call by reference, variable a value after function call becomes 2 where as in the case of call by value, result of the value of a after function call becomes 21.

9. In the following fragment, what is the meaning of int (*f)()? Explain what is happening and what will be the output?

```
int x1(){
    printf("1\n");
}
int* x2(){
    int (*f)()=x1;
    return (int*)f;
}
int main() {
    int (*ff)()=(int (*)())x2();
    (*ff)();
return 0;
}
```

Answer: Meaning of int(*f)() is that f is a pointer to a function which takes no arguments. In C language, function cannot return functions. However, we know function names are pointers to the functions. Thus, in function x2() we are typecasting f to int * and returning. In the main, the received address is type casted back and then function is invoked. Thus, the function x1() gets executed. So, we get output as 1.

Exercise Do understand what happens in the following code fragment. Think in the same lines of the above explanation.

```
int x1(int n){
    while(--n)printf("1\n");
}
int* x2(){
    int (*f)(int)=x1;
    return (int*)f;
}
int main() {
    int (*ff)(int)=(int (*)(int))x2();
    (*ff)(10);
return 0;
}
```

10. See the following code fragment having two ways of displaying the elements of a 3-D array? Does the outputs will be same in both approaches?

```
int a[3][2][2] = { 10,2,3,4, 5,6,7,8,9,9,9,9  };
int *p;
for(int i=0;i<3;i++)
for(int j=0;j<2;j++)
for(int k=0;k<2;k++)
printf("%d\t",*(*(*(a+i)+j)+k) );
printf("\n");
p=&a[0][0][0];
for(int i=0;i<3*2*2;i++)
printf("%d\t",*(p+i));
```

Answer: Yes. Second approach is preferred over the first one as it needs less number of indirection operations.

11. The following statement gave output as 65565. Before executing this statement i value is 5. Explain.

```
printf("%d%d%d%d%d%d",i--,i++,--i,++i,i);
```

Answer: While processing functions, variables are pushed into stack left to right then processed. Thus, i value is processed first, then ++i and vice versa. Thus, we get the above output.

12. Does the code fragment gives same output? What happens if we add x[0]='r'; statement at the end?

```
char *x="Rama", *y="Rama";
    printf("%p %p\n",x,y);
```

Answer: Both will give results. While storing constants, if more than one constant is same, only one of them is stored in memory. Thus, only one "Rama" is store in memory and its address is assigned to x and y pointer variables. Thus, x and values are same addresses.

13. Does the code fragment gives same output? What happens if we add x[0]='r'; statement at the end?

```
char x[]="Rama", y[]="Rama";
    printf("%p %p\n",x,y);
```

Answer: No.

14. Does the memory allocated for x and y are same? Or in the same area? If we try to send x and y as arguments to size of method, do we get same values as output?

```
char *x="Rama"; char y[]="Rama";
printf("%d %d\n", sizeof(x), sizeof(y));
```

Answer: No. We get 4 and 5 as output.

15. A smart student represented the following if else statements in a compact form as: `j = i, j?(i,j)?i:j:j;` . Is it equivalent?

```
if(i,j){
if(i,j)
j = i;
else
j = j;
}
else
          j = j;
```

Answer: Yes.

16. Explain how the following statements makes variable i as 1.

```
 int a=5,b=10,c=0;
 int i=b>a>c;
```

Answer: i = (b > a) > c;

i = (10 > 5) > 0;

i = 1 > 0;

i = 1

17. Can the following code fragment be compiled?

```
#define X 0
#define Y 0
switch(5/4/3){
case X: printf("Clinton");
break;
case X+1:printf("Obama");
break;
case X+Y:printf("Hower");
break;
default: printf("Lincoln"); break;
}
```

Answer: No. We have two cases for expression value of 0. Thus, we get error "duplicate case value".

18. What is the output of the following printf statement? **Explain.**

```
printf("%c\n", "AB""CD" "MMM"[strlen("AB" "CD" "MMM"+strlen("AB"))]);
```

Answer: M

In C language, the following style of concatenation of string constants gives a string.

"AB""CD" "MMM"

For example, the above gives a string constant "ABCDMMM". Also, we know string name itself pointer to the string. Thus, "AB""CD" "MMM"+strlen("AB") becomes a string "CDMMM". When this is passed as argument to strlen function, it returns 5. This is used as the index of the string "ABCDMMM", in which 5th character is M.

19. What will be value of z at the end of the following statements?

```
int z, x = 5, int y = -10, a = 4, b = 2;
z = x++ - --y * b /a;
```

Answer: 10

First y is incremented by 1. Thus it becomes 11.

Then y will be multiplied by b. Thus, we get 22

This 22 will be divided by 4. Thus we get 5

This is added to x value that is 5. Thus, expression value becomes 10 and after that x values becomes 6.

Thus z value becomes 10.

20. The format strings %f , %lf are same with both printf and scanf functions?

Answer: No.

Scanf considers them differently.

21. Consider an assignment statement " float f=26;". We know 26.0 is stored in f. When this conversion takes place? During compile time or run time?

Answer: During compile time

22. The following recursive function is proposed to calculate (n+2)/n! for a given value of n. Is it correct implementation?

```
float sum(int i){
if(i<=1) return 1;
else return ((i+2)/(sum(i-1)*i));
}
```

Answer: No

The recursion relation is incorrect. To put it little differently, let's look at it using induction. Suppose sum(n-1) gave us the correct answer, then it would have returned (n – 1 + 2)/(n – 1)! = (n + 1)/(n – 1)!

Now we have defined sum(n) = (n + 2)/(sum(n–1)*n)

Substitute the result from sum(n–1).

We get

sum(n) = (n+2)*(n–1)! / (n+1)!

which is obviously not what we want.

So this shows that the recurrence relation is not correct. To solve for a function f(n) using recursion, we must try to express f(n) as a function of f(n–1) and n.

We can verify this by preparing snap shot also. Consider the function is called with 4.

1st call i=4 return (6/sum(3)*4) after return from fourth call return (6/1*4). That is, it returns 24	2nd call i=3 return (5/sum(2)*3) after return from 3rd call return(5/(2*2)). That is, it returns 1.	3rd call i=2 return (4/sum(1)*2) after return from fourth call return(4/(1*2)). That is, it returns 2.	Fourth call i=1 return (1)

23. Does the following program generates the sequence 5 4 3 2 1?

```
int main(){
static int var = 5;
printf("%d ",var--);
if(var) main();
return 0;
}
```

Answer: Yes.

24. The following gives 64. Explain how.

```
int i=320;
char *ptr=(char *)&i;
printf("%d",*ptr); }
```

Answer: Assuming that the machine uses 2 byte integers. 320 will be stored in memory as

```
0000 0001 0100 0000
See the following code:
#include<stdio.h>
#include<stdlib.h>
int main(){
int i=320;
char *ptr=(char *)&i;
printf("%d\n", &i);
printf("%ld %ld\n", ptr, (ptr+1));
printf("%d %d\n",*(ptr), *(ptr+1));
system("PAUSE");
return 0;
}
```

Output:

2293620

2293620 2293621

64 1

Address of i 2293620

Address of first byte is assigned to ptr.

My program tries to print ptr and ptr+1 values.

We find they are 2293620 and 2293621

That is first value of 320, i.e 01000000 is stored at 2293620 while 00000001 at 2293621. Thus, when we try to print *ptr you are getting 64. We tried to print *(ptr+1) also which gave value as 1.

This computer is little endian style, thus the result is like this. If the computer is big endian style, we may get the reversed values.

25. The following statements giving 115 as output. Explain.

```
char *p="hai";
printf("%d\n",*p+1); (it shows 115)
```

Answer:

Here, p is a pointer variable. String constant "hai" is stored in computer memory and its address is assigned to p.

First cout prints h as *p refers to that character

We are printing *p+1 where *p is h, that is its ASCII code is 114 for which 1 is added and thus we are getting 115.

26. Explain why we are getting 2 as output?

```
enum value{VAL1=0,VAL2,VAL3,VAL4,VAL5} var;
printf("%d\n",sizeof(var));
```

Answer: var is enum type whose values can be either VAL1 or VAL2, etc which are integers. In the given machine, for an integer 2 bytes may be allocated. Thus, you are getting 2.

27. Consider the following function and its sample call. What will be the value of variable i, after the function call? Explain.

```
void fillArray (int array[], int *i,int number) {
if (number < 10 ) {
       array[*i] = number;
       *i = *i +1;
    } else {
       fillArray (array, i, number/10);
       array[*i] = number%10;
       *i = *i +1;
    }
}
main(){
int array[100];
int i = 0, n = 123;
fillArray (array,&i,n);
}
```

Answer: At the end, in i you will have the size of the array. Rather, number of digits of the third argument.

28. The following is a recursive function that determines whether a given integer array is sorted or not. It is missing one line of code, which is a return statement in the recursive case of the code. Complete the function by supplying this missing line of code.

```
public static boolean isSorted(int[] data, int
n){
// base case
if(n == 1) return true;
else{
   // Recursive case
   boolean temp = isSorted(data, n-1);
// Here is the missing line of code
return (temp) && (data[n-2] <= data[n-1]);
   }
}
```

Answer: The above program itself is having the missing return statement.

29. There exists three algorithms p,q, and r for a problem with the following worst case time complexity. Represent the same in big-oh notation and comment which one do you prefer.

p: $T(n)=17$

q: $T(n)=7\log_2 n+20$

r: $T(n)=59N\log_2 n+42n\log_{10} n+12\log_2 n+3$

Answer:

p:O(1)

q:O(logn)

r:O(nlogn)

Algorithm p is preferred over others.

30. Explain how to create a 3-D array of size 5x12x27 dynamically. How to free the same?

```
int i,j, ***array;
  array = (int ***) malloc(5 * sizeof(int **));
   for (i = 0; i < 5; ++i) {
       array[i] =
         (int **) malloc(12 * sizeof(int *));
       for (j = 0; j < 12; ++j)
          array[i][j] =
            (int *) malloc(27 * sizeof(int));
    }
```

To free the array:

```
for(i=0;i<5;i++)
for(j=0;j<12;j++)free(array[i][j]);
for(i=0;i<5;i++)free(array[i];
free(array)
```

31. Below are three variable declarations, followed by 5 code fragments, each ending with a conditional. State for each fragment whether it will print 'True' or not. Assume int a=5, b=7,c=11; is the first statement before each fragment.

```
if( (a <= 5) && (b > 7) ) {
printf("Fragment 1 True\n" ) ;
}
---------
if( ((a < 5) && (b > 7)) || (a < b) || (b > 1000)
) {
printf("Fragment 2 True\n" ) ;
}
-----------
if( !(!(a <= 5) && !(b > 7)) ) {
printf("Fragment 3 True\n" ) ;
}
-------
if( ((a <= 5) && (b > 7)) != 0 || ( c >= a*b )
) {
printf("Fragment 4 True\n" ) ;
}
-----------
if( (c = 1) || (b > 9) || (a < -3) ) {
printf("Fragment 5 True\n" ) ;
}
```

Answer:

Fragment 1 false: true and false is false

Fragment 2 true: false or true or false is true

Fragment 3 true: not (not true or not false) is not (false) is true

Fragment 4 false: (true and false) != false or false is false or false is false

Fragment 5 true : (watch out c=1 means assign 1 to c, and has the value '1', true): true or false or false is true

32. Consider the following six figures I to VI. Assume that you have given x and y coordinates of a point. Write a C statement to check whether the given point is inside the shaded area of each of the figures.

Answer:

I : `(x>=1 && x<=3 && y>=1 && y<=2)`

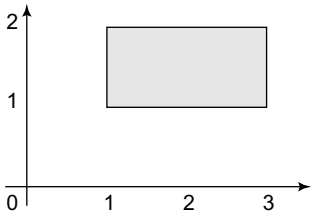
II : `(x-1)*(x-1)+(y-1)*(y-1) <= 1`

III : `!(x>1 && x<3 && y>1 && y<2) && x>=0 && y>=0`

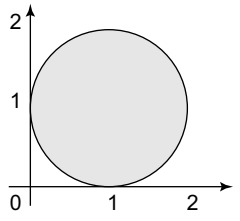
IV : `(y>=x && y >= -x && y<=2-x && y<= x+2)`

V : `(x>=0 && x<=3 && y>=0 && y<=2)`

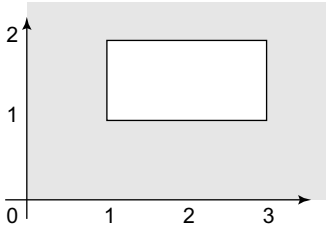
VI : `(x>=1)`



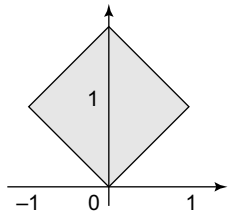
I



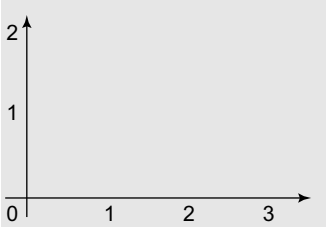
II



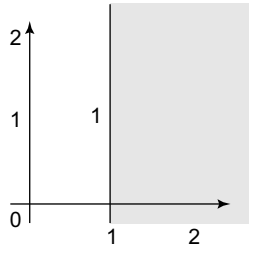
III



IV



V



VI

33. Consider the following functions which are calling others also internally. Assuming that each of them are called initially with 4, give possible outputs of them. Do comment which of them may enter into infinite loop.

```
a(int y) {
while( y >=0 ) {
printf( "%d", y ) ;
y = y - 1 ;
}
}
b(int y) {
if( y >=0 ) {
b( y - 1 ) ;
printf( "%d", y ) ;
}
}
c(int y) {
printf( "%d", y ) ;
```

```
c( y - 1 ) ;
}
d(int y) {
while( y > 0 ) {
printf( "%d", y ) ;
d( y - 1 ) ;
}
printf( "%d", y ) ;
}
e(int y) {
while( y > 0 ) {
printf( "%d", y ) ;
y = y - 1 ;
}
e( y - 1 ) ;
printf( "%d", y ) ;
}
f(int y) {
if( y >=0 ) {
printf( "%d", y ) ;
}
f( y - 1 ) ;
}
```

Answer:

Function a: 43210 TERMINATES

Function b: 01234 TERMINATES

Function c: 43210-1-2-3... DOES NOT TERMINATE

Function d: 43210101010...DOES NOT TERMINATE

Function e: 43210 TERMINATES

Function e: 43210 DOES NOT TERMINATE

34. Is n++ is faster or n=n+1 is faster?

Answer: n++ is fast.

35. Can we apply modulus operator to float type operands?

Answer: No. It is applicable for int, long, char.

36. Can we apply bitwise operators to float type operands?

Answer: No.

37. Can we use string constants in switch cases?

Answer: No. We can use symbolic constants, integer constants, character constants only.

38. Is it possible to use an integer array element as an index of another integer array?

Answer: Yes.

39. Is it possible to use a function name to the right hand side of =.

Answer: Yes. If that function returns address, we can use it.

40. How many ways we can return more than one value from a function?

Answer:

A. Though global variables by storing the values to be returned in thoese global variables.

B. By sending a dummy array and storing the values to be returned in this array.

C. By sending some addresses and storing the values to be returned in those addresses.

D. By creating a dynamic array and storing the values to be returned in that array and return the starting address of this array.

E. By return a structured variable and storing the values to be returned in the data members of that structured variable

41. Is it better to write a function such that it takes addresses of the arrays instead of arrays themselves?

Answer: Functions taking addresses.

42. What is the output of the following program.

```
#include<stdio.h>
int main(){
int i=0;
while (i <= 10){
printf("%d\n",i);
}
return 0;
}
```

Answer: Un-ended zero's.

43. What is the output of the following program?

```
#include<stdio.h>
int main()
{
int i=7;
printf("%d\n", i=7 != 8);
printf("%d\n",i);
return 0;
}
```

Answer: First 7!=8 is evaluated and its value is assigned to i. Thus, we get

1

1

44. What is the output of the following program?

```
#include<stdio.h>
```

```
int main(int z){
printf("%d\n",z);
return 0 ;
}
```

Answer: 1

45. Does the following program compiles and executes?

```
#include<stdio.h>
abc(){
printf("hi");
}
int main(){
printf("%x\n",abc);
return 0;
}
```

Answer: Yes. It prints addess of the function abc.

46. Which is better; navigating an array using a pointer or subscripts?

Answer: C compiler to generate optimal code for traversal using pointers than for subscripts.

47. Is using exit() the same as using return? Mention with respect to main()?

Answer: Yes. However, with respect to call in other functions, the exit() function is used to exit your program and return control to the operating system. It flushes all the opened streams before exiting. The return statement is used to return from a function and return control to the calling function.

48. What is the use of atexit() function?

Answer: To execute code even after the program exits the main() function.

49. What is the heap memory?

Answer: The heap is that area of our program under execution from which malloc(), calloc(), and realloc() get memory. Getting memory from the heap is much slower than getting it from the stack. On the other hand, the heap is much more flexible than the stack. Memory can be allocated at any time and deallocated in any order. Such memory isnt deallocated automatically; you have to call free().

Run-time stack is developed in the heap itself. Recursive data structures are almost always implemented with memory from the heap. Strings often come from there too, especially strings that could be very long at runtime. If we can keep the data in a local variable (and allocate it from the stack), our code will run faster than if we put the data on the heap. Sometimes we can use a better algorithm if we use the heap faster, or more robust, or more flexible. Its a tradeoff. If memory is allocated from the heap, its available until the program ends. Thats great if we remember to deallocate it when we are done with the allocated memory. If we forget, its a problem. A memory leak is some allocated memory thats no longer needed but is not deallocated. If we have a memory leak inside a loop, we can use up all the memory on the heap and not be able to get any more. (When that happens, the allocation functions return a null pointer.) In some environments, if a program does not deallocate everything it allocated, memory stays unavailable even after the program ends. It grows toward lower addresses.

50. What is a dangling pointer. How to avoid it?

Answer: We assign NULL to the elements (pointer) after freeing them such that some catastrophic things will not happen in future. After a pointer has been freed, we can no longer use the pointed-to data. The pointer is said to dangle; it does not point at anything useful. If we NULL out or zero out a pointer immediately after freeing it, our program can no longer get in trouble by using that pointer.

51. What is the output of the following program?

```
int main(int n){
    printf("%d\n",n);
    if(n)main(n-1);
    system("PAUSE");
    return 0;
}
```

Answer: Here, n value becomes 1 when we start the program without any command line arguments. Second time n value is becoming 0, thus main will not be called again recursively. Output of the above program is:

1

0

52. What will be the output of the following program?

```
int v=9;
int main(){
    printf("%d\t",v);
    if(--v)main();
    system("PAUSE");
    return 0;
}
```

Answer:

9 8 7 6 5 4 3 2 1

53. Does the following code gets compiled?

```
int j=22;
while(--j--)
printf("Hi");
```

Answer: No, We get error non lvalue in decrement.

54. How many times the following while loop runs?

```
int j=22;
while(--j,--j)
printf("Hi\n");
```

Answer: 10 times. Value of j is decremented two times each. Thus, j value changes as 20, 18, 16, 14, 12, 10, 8, 6, 4, 2. When it becomes 0, it will not enter into loop.

55. How many times the following while loop runs.

```
int j=20;
while(--j,j--)
printf("Hi\n");
```

Answer: It j value is odd it terminates after j/2 times. Else it becomes as infinite loop.

56. How many levels of pointers can you have? That is, like pointer to pointer (**), etc.,.

Answer: We can have in at least 12 levels like:

```
int i = 0;
int *ip01 = & i;
int **ip02 = & ip01;
int ***ip03 = & ip02;
int ****ip04 = & ip03;
int *****ip05 = & ip04;
int ******ip06 = & ip05;
int *******ip07 = & ip06;
int ********ip08 = & ip07;
int *********ip09 = & ip08;
int **********ip10 = & ip09;
int ***********ip11 = & ip10;
int ************ip12 = & ip11;
int *************ip13 = & ip12;
int **************ip14 = & ip13;
int ***************ip15 = & ip14;
```

The ANSI C standard says all compilers must handle at least 12 levels. On our Blood-shed compiler, we have got 15.

57. Does the following function gets compiled?

```
void f()
  {
    break;
    }
```

Answer: No. We can not use break as it use. It should be used either in loop or switch.

58. Explain why the following code works like an infinite loop.

```
main( ) {
char ch ;
for ( ch = 0 ; ch <= 255 ; ch++ )
printf ( "\n%c %d, ch, ch ) ;
}
```

Answer: Can you believe that this is an indefinite loop? Probably, a closer look would confirm it. Reason is, *ch* has been declared as a char and the valid range of char constant is –128 to +127. Hence, the moment *ch* tries to become 128 (through *ch*++), the value exceeds the character range, therefore the first number from the negative side of the range, i.e. –128, gets assigned to *ch*. Naturally the condition is satisfied and the and the control remains within the loop eternally.

59. Is the following code is acceptable?

```
void f(){
          goto xx;
          }
int main(){
    bool a=true;
    xx:
}
```

Answer: No. We can not have non local jump through goto. We will get compilation error.

60. Is there be any problem in the following code fragement?

```
int x[10]={1,9,1,1};
printf("%d\n", *x);  /* first element will be
printed*/
x++;
printf("%d\n", *x);  /* Second element we want
*/
```

Answer: No. This gives an error during the compiling. We can not increment the x value as it is constant pointer.

61. The following code fragement is giving 5 and 8 as the output. Explain.

```
int x[10]={1,1,9,1};
```

```
    printf("%d\n", *x+2 - *x+3);
    printf("%d\n", *(x+2) - *(x+3));
```

Answer: Here, in the first printf(), *x is first element value, which is 1. Thus, 1+2-1+3 is calculated. Where as in the second one *(x+2) means 9 and *(x+3) means 1. Thus, result is 8.

62. Why the following code fragement is giving output as "Not Same"?

```
int main(){
 float x=10, y;
 y=10/3*3;
 if( x==y)printf("Same\n");
 else printf("Not Same\n");
 system("PAUSE");
 return 0;
}
```

Answer: Because of integer division, y value becomes 9 (not 10).

63. What is NaN?

Answer: It stands for Not a Number. NaN is a value or symbol that is usually produced as the result of an operation on invalid input operands, especially in floating-point calculations.

The following practices may cause NaN:

- All mathematical operations with a NaN as at least one operand
- The divisions 0/0, ∞/∞, ∞/−∞, −∞/∞, and −∞/−∞
- The multiplications 0 × ∞ and 0 × −∞
- The additions ∞ + (−∞), (−∞) + ∞ and equivalent subtractions.
- Applying a function to arguments outside its domain, including taking the square root of a negative number, taking the logarithm of a negative number, taking the tangent of an odd multiple of 90 degrees (or π/2 radians), or taking the inverse sine or cosine of a number which is less than -1 or greater than +1.

64. When we have executed the following program, we have got same as the addresses of the the data members of a union. Explain.

```
#include<stdio.h>
#include<stdlib.h>
union ITEM{
      int A;
      float B;
      char C[20];
      } A;
int main(){
    printf("%p %p %p", &A.A, &A.B, A.C);
    }
```

Output:

```
010 00404010 00404010
```

Answer: For union type variables, memory is allocated commonly. We will be storing only any one of the data member only. Thus, addresses for all the data members will be same.

65. Does the following function returns last digit of 3^n, for given positive value of n?

```
int last_digit_3n(int x)
{
static int d[] = { 1, 3, 9, 7 };
return d[x % 4];
}
```

Answer: Yes. Series 3^n will be 3, 9, 27,81,243,729…. If we observe last digits they will be 3, 9, 7, 1, 3, 9… Thus, the above function can be used for the purpose of finding last digit of 3^n.

66. What is the output of the following C code fragment? Explain.

```
float x=0.085;
int p=x*1000;
if(p==(1000*x))printf("Yes\n");
else
printf("No\n");
```

Answer: We get No. Because, 0.085 when stored in float variable it will be stored like 0.08500000089406 9671630859375. Thus, p value becomes 85 while 1000*x becomes 85.000000894069671630859375. As p is not same as this number, we get No as the output.

67. What is the output of the following code fragment?

```
double x1 = 0.3;
double x2 = 0.1 + 0.1 + 0.1;
if(x1 == x2)printf("Yes\n");
else
printf("No\n");
double z1 = 0.5;
double z2 = 0.1 + 0.1 + 0.1 + 0.1 + 0.1;
if(z1 == z2) printf("Yes\n");
else
printf("No\n");
```

Answer: No and Yes. When 0.3 is represented in float it takes infinite series. Where as 0.5 can be represented in correct form. Thus, we get No in the first case and Yes in the second case.

2.3 Objective Questions

1. Valid fact about goto statement of C language
A. It is used to increase the logical clarity of the program
B. It makes debugging easy
C. It is the only way to come out of innermost loop
D. Its use is highly discouraged

2. Minimum number of temporary variables needed to exchange two unsigned character type variables.
A. 1 B. 2 C. 0 D. 3

3. Assume that we wanted to exchange values of two unsigned character type variables values. Only one operator is permitted to be used. What is it?
A. + B. – C. << D. ^

4. How many times "Hi" message will be displayed when we execute the following for loop?
```
for(i=0; i%2?1:0;i++) printf("Hi\n");
```
A. 5 B. 2 C. 0 D. 1

5. Modulus operator is applicable for
A. double B. float C. int D. None

6. The ternary operator
A. + B. * C. % D. ?:

7. False one in the following.
A. –100 b. 0 C. 1 D. None

8. The – operator is
A. Unary B. Binary C. Both D. None

9. The driver program ___
A. scanf B. main C. printf D. None

10. ____ statement allocates memory in a C program.
A. scanf B. main
C. variable declaration D. variable address

11. The variable type which has range of -128 to 127.
A. char B. unsigned char
C. byte D. int

12. The ?: operator is ___
A. Unary B. Binary
C. Low level D. Ternary

13. A variable acquires meaningful value through
A. Reading B. Assignment
C. Initialisation D. All

14. Garbage value is __
A. Infinite
B. Negative infinite
C. No one can predict what it is going to be
D. Zero

15. The variable type which has the range of 0-255.
A. char B. unsigned char
C. byte D. int

16. C supports byte type variables.
A. True B. False
C. Not applicable D. None

17. A program has to declare a variable which is supposed to store values between 0 to 65535 as
A. int B. Two byte int
C. Long D. None

18. Find uncommon one with respect to variable declaration.
A. int B. long
C. void D. long

19. The data type used to specify the function return types is
A. int B. char C. void D. long

20. Find odd one
A. int B. char C. float D. long

21. Find incorrect one
A. signed int B. unsigned char
C. unsigned float D. None

22. The statement to declare a constant
A. const B. #define C. Both D. None

23. The symbolic constant
A. #define N 10 B. const N=10;
C. Both D. None

24. The keyword which can make a variables value to not to get changed
A. define const B. const
C. volatile D. None

25. Find odd man out.
A. TAB B. New line
C. Carriage return D. Comma

26. Acceptable variable name
A. scanf B. Keywords
C. main D. abc
E. int

27. The scanf needs
A. stdio.h to be included
B. Format string
C. List of addresses
D. All

28. The format to print a float variable value in exponential form is
A. %f B. %lf C. %e D. None

29. The %f is the format string with scanf function, then it can take ___ from the keyboard.
A. Only numbers with decimal point
B. Only integers
C. Both
D. None

30. The error which we get if we use a variable which is not declared
A. No such variable name
B. Use of undeclared variable
C. Variable declaration missing
D. Undefined symbol

31. C is ___
A. Procedural language
B. Not strictly typed language
C. Medium level language
D. All

32. In Indian banking system, while calculating interests we need precision of
A. Huge digits B. Two digits
C. 6 decimals D. None

33. Invalid operator
A. += B. –= C. /= D. >==
E. %=

34. Find odd man with respect to ;
A. printf B. main
C. scanf D. #define
E. #Include

35. What is the return type of printf function?
A. string B. int
C. char D. None

36. What is the output of the following printf statement?
printf("%d\n", printf("%3d %3d\n", x, x));
A. Error B. 8 C. 6 D. None

37. The probable error which we may get with sqrt function
A. Overflow
B. Domain error
C. math.h is not included
D. None

38. f A and B are two integers, then A%B value can be
A. A B. –A C. B-1 D. 0
E. All

39. Fastest operator
A. + B. – C. * D. |
E. None

40. The scanf function returns
A. integer
B. Number of values which it reads
C. Both
D. None

41. The operator +=
A. Sum
B. Implicit assignment statement
C. Binary
D. None

42. With logical AND operator, if first operand is false then
A. Second operand will be evaluated
B. Second operand will not be evaluated
C. The final result will be false
D. None

43. With logical AND operator, if first operand is true then
A. Second operand will be evaluated
B. Second operand will not be evaluated
C. The final result will be false
D. None

44. Logical operators precedence
A. Higher than modulus
B. Higher than *
C. Lower than /
D. None

45. The default storage type
A. Static B. Automatic
C. Register D. External
E. None

46. Is it possible to declare variables in between computational statements? (Y/N).

47. Value of variable i after the execution of the following statement is__. Assume initial value of i is 10.
i = i++ + (i++);
A. 12 B. 22 C. 23 D. None

48. Value of variable i after the execution of the following statement is__. Assume initial value of i is 10.
i = ++i + (i++);
A. 12 B. 22 C. 23 D. None

49. Output of the following program

```
#include<stdio.h>
int main(){
   int i=0, j=0,k=0;
   j=i||++i;
```

```
    k=i&&++i;
    printf("%d %d %d\n", i,j,k);
     return 0;
}
```

A. 0,0,1 B. 1,1,1 C. 2,1,1 D. 2,1,0

50. In C language, we can declare variable wherever we want. (Y/N)

51. In C language, all the variables has to be declared at the beginning of a block. (Y/N)

52. Does the following versions of scanf and printf functions gets compiled? (Y/N)

scanf("");

printf("");

53. Is there any mistake in the following code fragment?

```
#include<stdio.h>
int main(){
   int x=10;
   printf("%d\n", x):
   float a=10.2;
}
```

54. C language uses _____ characters.

A. ASCII B. ISCII

C. UNICODE D. EBCDIC

55. ASCII code is ___ bit code

A. 8 B. 16 C. 12 D. None

56. To convert an uppercase character to lowercase, we have to add ___

A. 32 B. 16 C. 48 D. None

57. The digit 9 is not same as

A. 9 B. '9' C. 00111001 D. b&c

58. The upper case A is not same as

A. 'A' B. 01000001 C. O101 D. None

59. Escape character

A. '\A' B. '\n' C. '\M' D. None

60. The beep sound

A. '\a' B. '\b' C. 'beep' D. None

61. We can assign a character to an integer variables. (Y/N)

62. One can assign an integer to a character variables. (Y/N)

63. To get digit value from digit, we have to subtract ___ from digit.

A. 8 B. 16 C. 32 D. 48

64. In which bit position, upper case and lower case characters differs? Assume least significant bit is referred to 0th bit.

A. 5 B. 6 C. 9 D. 4

65. The range of unsigned integers which we can store in a character variable.

A. –100 to 100 B. –128 to 127

C. –255 D. None

66. The range of integers which we can store in a character variable.

A. –100 to 100 B. –128 to 127

C. –255 D. None

67. In C, number of bytes needed to store any symbol is _

A. 4 B. 2 C. 1 D. None

68. Memory needed to store '\n'

A. 1

B. 2

C. Depends on computer

D. None

69. Memory needed for a character constant.

A. 1 B. 2 C. 3 D. None

70. Value of V after executing the following statements.

char C='P';

++P++;

A. 'R'

B. V

C. Statement will not get compiled

D. None

71. Value of V after executing the following statements.

char C='P';

P++=P++;

A. 'R'

B. V

C. Statement will not get compiled

D. None

72. Value of V after executing the following statements.

char C='P';

++P=P++;

A. 'R'

B. 'Q'

C. Statement will not get compiled

D. None

73. Is it possible to store all the possible human blood groups types in a character variable? (Y/N)

74. Either if or else block contains only single statement, then curly braces are not mandatory. (Y/N)

75. The main block should need to have both the curly braces even if it contains single statement. (Y/N)

76. We can use comma separated list in if condition. (Y/N)

77. Conditional expression of if can be null (i.e., empty) also. (Y/N)

78. Is it possible to use print statement inside if constructs condition ? (Y/N)

79. What is the output of the following code fragment?

```
int i=10;
if(scanf(""))printf("%d\n", i);
```

A. Does not get compiled
B. Garbage
C 10
D. None

80. What is the output of the following code fragment?

```
int i=10;
if(printf(""))printf("%d\n", i);
```

A. Does not gets compiled
B. Garbage
C. 10
D. None

81. What is the output of the following code fragment?

```
int i=10;
if(printf("%d ", i))printf("%d\n", i);
```

A. Does not gets compiled
B. Garbage
C. 10 10
D. None

82. What is the output of the following code fragment?

```
int i=10;
 if(i=scanf("%d", i)){
 printf("%d\n", i);
 }
```

A. 10 B. 0
C. 1 D. What ever we enter

83. What is the output of the following code fragment?

```
int i=10;
 if(i=printf("%d", i)){
 printf("%d\n", i);
 }
```

A. 10 B. 0 C. 102 D. 2

84. We can use any type of constants with case constructs of switch. (Y/N)

85. What is the output of the following code fragment?

A. 10 10
B. Program will not be compiled
C. 10 D. None

86. What is the output of the following code fragment?

A. 10 10
B. Program will not be compiled
C. 10 11
D. None

87. Unreachable code error occurs with

A. switch B. ?: C. If D. None

88. Does the printf statement is going to get executed any time in the following code fragment? (Y/N)

```
switch(i){
        case 10:
                break;
                printf("%d\n", i);
        default:
                break;
    }
```

89. Is it recommended to use goto from switch? (Y/N)

90. Unconditional jump

A. if B. switch C. goto D. break;

91. Is it mandatory to have default case at the end in a switch? (Y/N).

92. Does switch accepts comma separated statements in its expression? (Y/N).

93. What is the output of the following code fragment.

A. Program will not be compiled at all.
B. If we give 11, we get 11 on the screen
C. If we enter any other number, say, x, we get output as x x
D. None

94. Is it recommended to use goto inside a if or else block? (Y/N).

95. One can use break and continue statements inside if or else blocks. Do assume that these if and else blocks are not inside any loop. (Y/N).

96. Does the following if gets compiled? (Y/N).

```
if(continue) {  }
```

97. Does the following if gets compiled? (Y/N).

```
if(break) { }
```

98. What is the output of the following code fragment?

```
int i=10;
      if(i++?i++:-i){
      printf("%d\n", i);
      }
```

99. 10 B. 11 C. 12 D. None

99. Does the statement switch(i); {} gets compiled? (Y/N)

100. How many Hi's are printed the following code fragment if we enter 9?

```
int i;
   scanf("%d", &i);
   switch(printf("%d",i))   {
              case 1: printf("Hi"); break;
```

```
            case 9: printf("HiHi");break;
            default:printf("HiHiHi");break;
}
```

A. 1 B. 2
C. 3 D. None

101. We wanted a program using switch such that if we enter any single digit number, it should print Hi once. Similarly, if we enter ant two digit number, it should print two Hi's. Also, if we enter a three digit number, it should print Hi three times. The following is proposed. Will it work? (Y/N)

```
#include<stdio.h>
#include<stdlib.h>
   int main(){
   int i;
   scanf("%d", &i);
   switch(printf("%d",i)){
            case 1: printf("Hi"); break;
            case 2: printf("HiHi");break;
            case 3:printf("HiHiHi");break;
   }
    return 0;
}
```

102. Is there any mistake in the following code fragment? (Y/N).

```
int i=10;
if(scanf("%d", &i)); printf("Hi");
      else{
      printf("%d\n", i);
      }
```

103. Does practical compilers accepts infinite number of nested if conditions? (Y/N)

104. While loop condition cannot be null statement. (Y/N).

105. While loop condition can accept comma separated statements. (Y/N)

106. The loop which runs for 4 times.

A. `int i=0; while(i<5){ i++;}`
B. `int i=1; while(i<=5){ i++;}`
C. `int i=5; while(i--);`
D. `int i=5; while(--i);`

107. The statement which makes a while loop to skip statements in the current iteration and goes straight to while condition checking.

A. break B. continue
C. skip D. None

108. The code which does not give factorial n (assuming n is positive value) into variable f.

A. `f=1; i=1; while(i<=n) { f=f*i; i++);`
B. `f=1; i=1; while(i<=n) f *=i++;`
C. `f=n; while(--n) f *=n;`
D. `f=n; while(n--)f*=n;`

109. The code which gives into f for n (assuming n is positive value).

A. `f=1; i=1; while(i<=n) { f=f*i; i++);`
B. `f=1; i=1; while(i<=n) f *=i++;`
C. `f=n; while(--n) f *=n;`
D. `f=n; while(n--)f*=n;`

110. Output of the following code fragment.

```
int i=5;
 while(scanf("")){
    printf("%d\n", i);
    }
```

A. None
B. 5
C. Code will not get compiled at all
D. None

111. Output of the following code fragment.

```
int i=5, j=3;
      while(i++<5, j--) printf("Hi");
```

A. Program will not get compiled
B. HiHiHi
C. HiHiHiHiHi
D. HiHHi

112. Output of the following code fragment.

```
int i=5;
    while(printf("")){
    printf("%d\n", i);
    }
```

A. None
B. 5
C. Code will not get compiled at all
D. None

113. What is the output of the following code fragment?

```
     int i=5, j=3;
      while(i++<5&& j--) printf("Hi");
```

A. None B. Hi
C. HiHiHi D. HiHiHiHiHi

114. What is the output of the following code fragment?

```
      int i=5, j=3;
      while(i--<5&& j--) printf("Hi");
```

A. None B. Hi
C. HiHiHi D. HiHiHiHiHi

115. What is the output of the following code fragment?

```
int i=5, j=3;
while(i--<5&& j--) printf("Hi");
```

A. None B. Compilation Error
C. Infinite Hi's D. HiHiHi

116. What is the output of the following code fragment?

```
int i=0, j=3;
    while(i++<5|| j--) printf("Hi");
```

A. None B. HiHiHi
C. HiHiHiHiHi D. HiHiHiHiHiHiHiHi

117. What is the output of the following code fragment?

```
int i=0, j=3;
while(i++<5&&j--) printf("Hi");
```

A. HiHiHiHiHiHiHiHi B. Hi
C. HiHiHi D. HiHiHiHiHi

118. What is the output of the following code fragment?

```
int i=0, j=3;
while(i++<5, j--) printf("Hi");
```

A. None B. Hi
C. HiHiHi D. Hi word 8 times

119. What is the output of the following code fragment?

```
int i=0, j=3;
while(i++<5, j--);
printf("%d %d", i, j);
```

A. Does not get compiled
B. None
C. 4 –1
D. 4 7

120. We have two code fragments with while loops.

Fragment 1:	**Fragment 2:**
`int i=0, j=3;`	`int i=0, j=3;`
`while(i++<5, j--);`	`while(i++<5&& j--);`
`printf("%d %d", i, j);`	`printf("%d %d", i, j);`

A. Two will not gets compiled
B. Second one gets compiled and runs
C. Both give 4 –1 as output.
D. Both runs for 4 times.

121. The following code fragment is said to be running infinite times. Explain.

```
int i=0;
  while(scanf("%d", &i))
 printf("%d\n", i);
```

122. Find odd man out of the following.

A. `int i=0; while(i<5){ i++;}`
B. `int i=1; while(i<=5){ i++;}`
C. `int i=5; while(i--);`
D. `int i=5; while(--i);`

123. Find odd man out of the following.

A. `int i=0, n=5; while(i<n);`
B. `int i=0, n=5; while(i<n){i++;}`
C. `while(1){}`
D. `int i=5; while(printf("%d", i--));`

124. First element index of a 1-D array in C language is

A. 1 B. 2 C. 0 D. None

125. We can store all the possible human blood groups using string variables. (Y/N).

126. Both scanf and printf requires first argument as string. (Y/N).

127. Does the following loop prints the string and ends the loop? (Y/N)

```
char x[]="RAMA";
int i=0;
for(; x[i]; i++) printf("%c",x[i]);
```

128. What is the output of the following code fragment?

A. Program will not be compiled
B. R
C. M
D. None

129. Does the following code fragment prints the string? (Y/N).

```
char x[]="RAMA";
  int i=0;
  for(; i[x];printf("%c",i++[x]));
```

130. What is the output of the following code fragment?

```
char x[]="RAMA", y[]="RAO";
int i=0;
for(; i[x]&&i[y];printf("%c%c",i++[x],i[y]));
```

A. Compilation error B. Run-time error
C. RRAAMO D. ARMAAO

131. What is the output of the following code fragment?

```
char x[]="RAMA", y[]="RAO";
int i=0;
for(; i[x]||i[y];printf("%c%c",i++[x],i[y]));
```

A. Compilation error B. Run-time error
C. RRAAMO D. ARMAAO

132. What is the output of the following code fragment?

```
char x[]="RAMA", y[]="RAO";
int i=0;
for(; i[x]&&i[y];printf("%c%c",i[x],i++[y]));
```

A. Compilation error B. Run-time error
C. RRAAMO D. ARMAAO

133. We can use string constants case labels in a switch statement. (Y/N).

134. While reading data into a string variable using scanf, we need not apply address operator to the string variable as the string variable names itself refers to some memory location in C. (Y/N).

135. If we declare a 4-element character array, we can store more than 4 characters also in practice. (Y/N).

136. The function which is used to store formatted in a string is

A. printf B. formatprintf
C. sprint D. None

137. First argument in sprint is

A. An integer B. A string variable
C. A float D. None

138. The following code fragment is proposed to replace all the occurrences of second character with *. Will it work? (Y/N).

```
char x[80]="Ramamam";
int i = 0;
while(i[x]=(i[x]==1[x])?'*':i[x], i++[x]);
```

139. The following code fragment is proposed to replace all the occurrences of second character with *. Will it work? (Y/N).

```
char x[80]="Ramamam";
int i=0;
while(i[x]=(i[x]=='a')? '*' :i[x], i++[x]);
```

140. The following code output will be

```
char x[80];
int i=0;
sccanf("%s", x);
while( i++[x]);
printf("%d\n", i);
```

A. Does not get compiled at all.
B. Length of the string
C. One more than the number of characters in the string.
D. None

141. Does the following two code fragments gives same value of i?

Fragment 1:

```
char x[80];
int i=0;
sccanf("%s", x);
while( i++[x]);
printf("%d\n", i);
```

Fragment 2:

```
char x[80];
int i=0;
sccanf("%s", x);
while( x[i]!='\0')i++;
printf("%d\n", i);
```

A. Yes
B. No.
C. Second fragment gives i value as 1 more than first fragment.
D. None

142. The following code fragment is proposed to check whether the given string is having '.' Character or not. Will it work correctly? (Y/N).

```
char x[80];
int i=0;
scanf("%s",x);
printf("%s\n",x);
while( i[x]&&i++[x]!='.');
((i-1)[x]=='.')?  printf("Yes"): printf("No");
```

143. The function which converts an integer to string

A. atoi B. itoa C. atof D. ftoa

144. Both while and do-while loops behavior is same always. (Y/N).

145. The loop which runs once before checking the condition is:

A. while B. do-while
C. switch D. If and goto

146. The differences between while loop and do-while loop.

A. In do-while condition is checked after executing loop block once.
B. In do-while loop, after closing parenthesis of the loop condition, we should use ;
C. In while loop, we need not required to put ; after the closing parenthesis of while condition.
D. All

147. What is the output of the following code fragment?

```
do
printf("%d ",i++);
while(i<5);
```

A. It will not get compiled at all.
B. 0 1 2 3 4
C. 1 2 3 4 5
D. None

148. What is the output of the following code fragment?

```
do
;
while(printf("%d ",i++), i<5);
```

A. It will not get compiled at all.
B. 0 1 2 3 4
C. 1 2 3 4 5
D. None

149. What is the output of the following code fragment?

A. It will not get compiled at all

B. 0 1 2 3 4

C. 1 2 3 4 5

D. Runs for infinite times as printf returns a positive number always.

150. Find odd man out of the following.

A. `int i=0; do; while(printf("%d ",++i), i<5 );`

B. `int i=1; do; while(printf("%d ",i++), i<=5 );`

C. `int i=1; do; while(printf("%d ",i), ++i<=5 );`

D. None

151. Out of the following which does not behave like infinite loop?

A. `int i=1; do; while(printf("%d ",i), ++i<=5 );`

B. `int i=1; do; while(printf("%d ",i), i<=5 );`

C. `int i=1; do; while( i++<=5, printf("%d ",i) );`

D. `int i=1; do; while( 1 );`

152. What is the output of the following code fragment?

```
int i=1; do printf("%d ", i);    while( scanf("") );
```

A. It will not be compiled at all

B. Infinite times

C. 1

D. None

153. What is the output of the following code fragment?

```
int i=1; do printf("%d ", i);    while( printf("") );
```

A. It will not be compiled at all

B. Infinite times

C. 1

D. None

154. Is it possible to use comma separated lists inside a do-while loops condition? (Y/N).

155. If do-while block contains no statements at all, then we can remove curly braces and replace them with a single semicolon. (Y/N).

156. When the following do-while loop terminates?

```
int i; do; while(scanf("%d", &i), printf("%d", i)!=4);
```

A. Never

B. If we enter a 4 digit number

C. It will not get compiled at all

D. None

157. C language for loops accepts comma separated lists as its operands. (Y/N).

158. All the three operands in for loop can be null lists. (Y/N).

159. If all the three operands of a for loop are null lists, then for loop works like an infinite loop. (Y/N)

160. If for block contains no statement at all, then we can remove curly braces and replace them with a single semicolon. (Y/N).

161. The following version of for loop works like an infinite loop. (Y/N)

```
for();
```

162. The following version of for loop works like an infinite loop. (Y/N)

```
for(;;);
```

163. Number of semicolons needed in between two parenthesis of for loop are:

A. 1 B. None

C. 2 D. Any number

164. Is it possible to use instructions separated by semicolons in between the two parenthesis of for loop? (Y/N).

165. Value of l after executing the following code fragment.

```
int i=5,j, l=0;
for(i=1;i<=5;i++)
for(j=0;j<i;j++)l++;
```

A. 25 B. 5 C. 15 D. 3

166. Value of l after executing the following code fragment.

```
int i=5,j, l=0;
for(i=1;i<=5;i++)
for(j=0;j;  j++)l++;
```

A. 25 B. 15 C. 0 D. None

167. Loops are meant for

A. Iterative calculations B. Reiteration

C. Recursion D. None

168. What is the most common type of bug in software?

A. The "wrong way" problem, where a two way decision is written incorrectly.

B. The "wrong operator" problem, where an arithmetic expression does not mean what the programmer thought it did.

C. The "unititialized variable" problem, where a variable is used in an expression before its contents have been initialized.

D. The "off by one" problem, where a counting loop executes its body one time too many or one time too few.

169. Which is not loop in C?

A. for B. while

C. do-while D. until-repeat

170. Is it possible to use break in either of the three operands of for loop, i.e initialization, condition checking or modifier? (Y/N)

171. What is the value of i after executing the following code fragment?

```
int i=0;
for(;i++, ++i, i++, i<10;i++);
```

A. 10 B. 8
C. 11 D. 7

172. What is the value of i after executing the following code fragment?

```
int i=0;
for(;i++, ++i, i++, i<10;i++)break;
```

A. 11 B. 3
C. 4 D. 2

173. What is the value of i after executing the following code fragment?

```
int i=0;
for(;i++, ++i, i++, i<10;){
     continue;
     i++;
}
```

A. 11 B. 10 C. 12 D. None

174. Is it possible to use character variables also as loop control variables? (Y/N).

175. Only integer type variables can be used as loop control variables in for loop. (Y/N)

176. Output of the following code fragment

```
char i='A';
for(; i<='Z'; i=i+13){
     printf("%c",i);
}
```

A. AN
B. ABCD
C. None
D. Does not compile as for loop does not accept characters as control variables.

177. Arrays are used when we use no-memory approach. (Y/N).

178. All arrays contains a special character at the end to indicate that no more elements are available after it. (Y/N).

179. We can store integers in a float array without losing their values. (Y/N).

180. The following for loop on an n-element array reverses its elements. That is, 1st and last elements gets exchanged, second and last but one gets exchanged, and vice versa. (Y/N).

```
for(i=0, j=n-1;  i<n; i++, j--)  { int T=a[i];
a[i]=a[j];   a[j]=T; }
```

181. The following for loop on an n-element array reverses its elements. That is, 1st and last elements gets exchanged, second and last but one gets exchanged, and vice versa. (Y/N).

```
for(i=0, j=n-1;  i<j; i++, j--)  { int T=a[i];
a[i]=a[j];   a[j]=T; }
```

182. We want to propose a for loop which is supposed to run on an n-element array and reverses its elements. That is, 1st and last elements gets exchanged, second and last but one gets exchanged, and vice versa. The following for loop is proposed with empty middle operand. We can fill this with

```
for(i=0, j=n-1;  _____ ; i++, j--)  { int
T=a[i];  a[i]=a[j];   a[j]=T; }
```

A. i<n B. i</n/2 C. i<j D. None

183. What is the output of the following code fragment?

```
int a[]={10,20,30,40}, i=1, n=4;
while(a[i]+=a[i-1], ++i<n);
printf("%d\n", a[3]);
```

A. Compile-error B. Run-time error
C. 100 D. 40

184. What is true about the following code fragment?

```
int a[]={10,20,30,40}, i=1, n=4;
while(i[a]+=(i-1)[a], ++i<n);
```

A. Compile error
B. Run time error
C. 100 is the last element value after while loop
D. Element values becomes 10, 30, 60, 100

185. What is the output of the following code fragment?

```
int a[]={10,20,30,40}, i=1, n=4;
while(i[a]+=i-1[a], ++i<n);
printf("%d\n", a[3]);
```

A. Compilation error B. Run time error
C. 42 D. 100

186. We want to assign 100 to first element of the following array. Select not acceptable

```
int a[]={10,20,30,40}, i=1;
```

A. i-1[a]=100; B. 0[a]=100;
C. (i-1)[a]=100; D. [a]0=100;

187. The following code fragment is proposed to store 3,2,1 and 0 in a 4-element integer array. Will it work? (Y/N).

```
int a[4],n=4;
while(n--)a[3-n]=n;
```

188. It is proposed to store 3, 2, 1 and 0 in a 4-element array. Select the loop which does work. Array declaration is as follows.

```
int a[4],n=4;
```

A. while(n--)a[3-n]=n; B. while(--n)a[3-n]=n;
C. while(n--)a[n-3]=n; D. None

189. It is proposed to store 3, 2, 1 and 0 in a 4-element array. The following code is suggested. Which element value does not satisfy our requirement?

```
int a[4],n=4;
while(--n)(3-n)[a]=n;
```

A. 0^{th} element B. 3^{rd} element
C. Compiling error D. None

190. What is the output of the following code fragment?

```
int a[4]={0,0,0,0},i,j,n=4;
for(i=1;i<=n;i++, a[i-1]=a[i-2])
for(j=1;j<=i;j++) a[i-1]+=j;
printf("%d\n", a[3]);
```

A. 10 B. 15
C. 20 D. None

191. What is the output of the following code fragment?

```
int a[4]={0,0,0,0},i,j,n=4;
for(i=1;i<=n;i++)
for(j=1;j<=i;j++) a[i-1]+=i+j;
printf("%d\n", a[3]);
```

A. 20 B. 10
C. Error D. 26

192. An integer n-element array (a) contains values between 0–99 (including). It is proposed to calculate histogram with interval width. The following solution is proposed such that array H contains frequencies. Will it work? (Y/N).

```
int  a[100], i, n, H[10]={0,0,0,0,0,0,0,0,0,0};
for(i=0;i<n;i++) h[a[i]/10]++;
```

193. Is it mandatory to use array to calculate scalar product between two arrays?

194. Any type array element indexes starts from 0. In C language, we cannot change this behavior at all. (Y/N)

195. C language does support content addressable arrays and a special foreach loop is available. (Y/N).

196. Number of 1s in the following programs output.

A. 11 B. 10
C. Compile error D. Run time error

197. We cannot have ragged 2-D character arrays in C language. That is, rows with variable number of column numbers (Y/N).

198. Find the acceptable 2-D character arrays declaration.

A. char a[10][2];
B. char a[][10]={"ram", "ravi"};
C. char a[10][20]={"Ram", "Ravi"};
D. All

199. Acceptable means of accessing ith row jth column element of a 2-D character array.

A. a[j][i] B. a[i][j] C. i[j[a]] D. None

200. If a is a valid 2-D character array then 0[a] refers to

A. Invalid
B. First row of the 2-D character array
C. First row first element of the 2-D character array
D. None

201. If a is a valid 2-D character array then 0[1[a]] is

A. Invalid
B. First row of the 2-D character array
C. First row second column element of the 2-D character array
D. A Character

202. What is the output of the following code fragment?

```
char a[10][20]={"Ram", "Rao", "Raj"};
int i=3, j=3;
while(i--&&j--)
printf("%c", i[j[a]]);
```

A. jaR B. Raj C. Rao D. Ram

203. What is the output of the following code fragment?

```
char a[10][20]={"Ram", "Rao", "Raj"};
int i=3, j=3;
while(i||j)
printf("%c", (--i) [(--j)[a]]);
```

A. jaR B. Raj C. Rao D. Ram

204. Does the following code fragment gets compiled? Its objective is to exchange two strings of the 2-D character array. (Y/N).

```
char a[10][20]={"Ram", "Rao"};
 char s[20];
 s=a[0];
 a[0]=a[1];
 a[1]=s;
 printf("%s %s", a[0], a[1]);
```

205. The following code is proposed to print the addresses of two rows of 2-D character array. What will be the difference?

```
char a[10][20]={"Ram", "Rao"};
printf("%p %p", a[0], a[1]);
```

A. Compilation error B. Run time error
C. 20 D. 16

206. What is the output of the following code fragment?

```
char a[10][20]={"Ram", "Rao"};
 int i=1;
 printf("%s %s", i--[a], i[a]);
```

A. Rao Rao B. Ram Rao
C. Rao Ram D. None

207. We have a 2-D character array (a of size 26 × 26) in which we want to store A–Z in its diagonal elements. The following solution is proposed. Will it work? (Y/N).

```
for(i=0;i<26;i++)i[i[a]]='A'+i;
```

208. We have a 2-D character array (a of size nxn, where n is odd) in which we want to store As in the middle row and middle column. How many loops are needed at least side?

A. 1 B. 2 C. 4 D. None

209. The following code fragment is proposed to store A's in the 26×26 sized 2-D character array. Will it work? (Y/N).

```
char a[26][26];
int i,j;
for(i=0,j=0;i<26; j=0,i++)while(j<26) i[j++[a]]='A';
```

210. What is the output of the following code fragment?

A. Compile time error
B. Run time error
C. It stores all A's in the matrix a
D. It stores A in the first row, BB in the second row, CCC in the third row, and vice versa

211. Is it possible to use a while loop inside a for loops parenthesis. (Y/N)

212. Best suitable structure to store a matrix is

A. Integers B. 1-D arrays
C. 2-D arrays D. Pointers

213. Largest possible odd sized square matrix in a 2-D array of size mxn

A. mxm
B. nxn
C. q=min{m,n}. If q is even q=q-1. Required matrix size is qxq
D. q=min{m,n}. Required matrix size is qxq

214. If a is 2-D array of size nxn then the following for loop calculates its principal diagonal elements sum to the variable s. (Y/N)

```
for(i=0, s=0;  i<n; i++) s+=a[i][i];
```

215. If a is 2-D array of size nxn then the following for loop calculates its other diagonal elements sum to the variable s. (Y/N).

```
for(i=0, s=0;  i<n; i++) s+=a[i][n-1-i];
```

216. If a is 2-D array of size nxn then the following for loop calculates its principal diagonal elements sum to the variable s. (Y/N)

```
for(s=0;  --n; ) s+=a[n][n];
```

217. Output of the following code fragment

```
int a[10][10]={{1,1,1},{2,1,2},{7,2,1}}, i,n=3, x=0,y=0;
 for(i=0;  i<n;  x+=i[i[a]], y+=i[(n-1-i)[a]], i++);
    printf("%d %d\n", x, y);
```

A. 3 3 b 3 9 C. Error D. None

218. Output of the following code fragment

```
int a[10][10]={{1,1,1},{2,1,2},{7,2,1}}, i,n=3, x=0,y=0;
 for(i=0;  i<n;  x+=i[n/2[a]], y+=n/2[i[a]], i++);
    printf("%d %d\n", x, y);
```

A. 3 3 b 3 9 C. Error D. None

219. What is the effect of the following code fragment?

```
int a[10][10]={{1,1,1},{2,1,2},{7,2,1}}, i,n=3, x=0,y=0;
 for(i=0;  i<n;  x+=i[(n/2)[a]], y+=(n/2)[i[a]], i++);
     printf("%d %d\n", x, y);
```

A. 5 4
B. It tries to calculate sum of the elements of middle column and middle row.
C. Compilation error
D. None

220. Does the following code fragments prints the array elements? (Y/N)

```
int a[3][3]={{1,1,1},{2,1,2},{7,2,1}}, i,n=3;
 for(i=0;  i<n*n;  i++)
       printf("%d\n", a[i]);
```

221. The following code fragment is proposed to traverse a square matrix in row-wise raster fashion. That is, in 0^{th} row elements are printed to left to right. In 1^{st} row, elements are printed from right to left; and vice versa. Will this solution works? (Y/N).

```
int a[3][3]={{1,1,1},{2,1,7},{7,2,1}}, i,n=3,j,l,k;
 for(i=0;  i<n;  i++){l=0;
            k=n-1;
  for(j=0;j<n;j++)printf("%d\n", a[i][i%2?k--:l++]);
  }
```

222. The following code fragment is proposed to test whether the square matrix is symmetric or not. Initially, it is assumed as symmetric. This is indicated by setting flag value as 1. When it is not found to be symmetric, the program sets the same to 0. Does the following satisfies this specification? (Y/N).

```
for(i=1;  i<n;  i++){
  for(j=0;j<i;j++)if (a[i][j]!=a[j][i]) { flag=0; exit(-1);}
  }
```

223. Output of the following code fragment.

A. Compilation error B. 3 3
C. 11 9 D. None

224. The following code fragment is proposed to calculate transpose of a square matrix and store in the same memory. Will it work? (Y/N).

```
int a[3][3]={{1,1,1},{2,1,7},{7,2,1}}, i,n=3,j,T;
 for(i=1;  i<n;  i++){
for(j=0;j<i;T=i[j[a]],i[j[a]]=j[i[a]],j[i[a]]=T,j++);
 }
```

225. Minimum number of loops needed two test uniqueness of a 2-D array is: (Without using pointers concept)

A. 2 B. 3 C. 4 D. None

226. Output of the following code fragment.

```
int a[3][3]={{1,1,1},{2,1,7},{7,2,1}}, i=3,j=3,s=0;
  while(i--&&j--) s+=a[i][j];
```

A. 9 B. 3 C. 4 D. None

227. Is the output of the following two fragments are same? (Y/N).

Fragment 1:

```
int a[3][3]={{1,1,1},{2,1,7},{7,2,1}}, i=3,j=3,s=0;
  while(i--&&j--) s+=a[i][j];
```

Fragment 2:

```
int a[3][3]={{1,1,1},{2,1,7},{7,2,1}}, i=3,j=3,s=0;
  while(i--,j--) s+=a[i][j];
```

228. What is the output of the following code fragment?

```
int a[3][3]={{1,1,1},{2,1,7},{7,2,1}}, i=3,j=2,s=0;
  while(i--||j--) s+=a[i][j];
```

A. Compilation error
B. Goes to infinite loop
C. Gives sum of elements
D. None

229. What is the output of the following code?

```
int a[3][3]={{1,1,1},{2,1,7},{7,2,1}}, i=3,n=3,s=0;
  while(n--) s+=a[n][n];
```

A. 9 B. 4
C. 3 D. Sum of elements

230. Minimum number of loops needed to multiply two matrices

A. 2 B. 4 C. 3 D. None

231. We wanted to multiply a vector of size 1xn, matrix of size nxn and a vector of size nx1. Minimum number of loops required for the same.

A. 2 B. 3 C. 4 D. None

232. Valid return type of a function.

A. Integer B. void C. struct D. real

233. A function can explicitly return any number of values. (Y/N).

234. In a function, we have used to return statements. Does it mean that it is returning two values? (Y/N).

235. Does return statement can take an expression also? (Y/N).

236. Is it possible to define a function inside another function? (Y/N).

237. Function names are

A. Key words B. Built-ins
C. Pointers D. None

238. Header files contains

A. Function definitions
B. Function calls
C. Function prototypes
D. Function signatures

239. Is it possible to employ some expressions while sending arguments to functions? (Y/N).

240. Two functions taking integer argument are called one after another. The addresses variables in both the calls are

A. Same B. Different
C. Differs by 4 bytes D. None

241. Formal arguments of a function

A. Can be of same type
B. Cannot have same name as that of other functions formal arguments
C. Can have same name as that of other functions formal arguments
D. Need not to follow naming conventions of variables.

242. Scratch variables

A. Formal arguments
B. Actual arguments
C. Are the ones declared inside the function definition
D. Are automatic type

243. When we send an argument in passing by value, whatever changes takes place on the formal argument will not reflect on the original. (Y/N)

244. Macros

A. Does not have type checking ability
B. Employs blind substitution policy
C. Are used if we wanted to do simple calculations.
D. All

245. We cannot remove curly braces of a function, if we have a single statement. (Y/N)

246. Is it possible to define a function inside main? (Y/N)

247. Find correct one

A. If we send an integer variable to a function, really its value is passed.

B. If we send a string to a function, the changes done by function are visible in the caller function.

C. If we send any array to a function, the changes done by function are visible in the caller function.

D. All

248. Stack variables

A. Formal arguments B. Actual arguments

C. Scratch variables D. None

249. Libraries containes

A. Function declarations

B. Function definitions in source language

C. Function definitions in machine/object languages

D. Function calls

250. Is it possible to call main from other functions? (Y/N)

251. There is limitation on number of arguments to a function while writing function definition. (Y/N)

252. Everything which can be realized through a loop can be converted to a recursive function. (Y/N)

253. Recursive function demands more system resources especially more stack memory. (Y/N)

254. The main can also be called recursively. (Y/N)

255. Tail recursion is employed to reduce stack space requirements of a recursive method. (Y/N)

256. Main problem with recursive functions is that it faces stack overflow situation. (Y/N)

257. Mathematical recurrence relationships in a problem are the basis for realising that problem in recursive manner. (Y/N)

258. We can implement a recursive function like a macro also. (Y/N)

259. Recursive functions also get expanded when they are called. (Y/N)

260. Out of the following statements the one which gives scope for recursive implementation

A. Go on divide n with x till n is not dividable with x

B. Go an add x till n is not zero.

C. Read and add in each iteration

D. None

261. If function A calls function B and B calls A. Then it is

A. Tail recursion B. Straight recursion

C. Self recursion D. Indirect recursion

E. Mutual recursion

262. If a, b are two integers and we want to calculate a modulus b without using % operator. Thus, we go on subtract b from a till a is larger than b. At the end, a is returned as modulus. Does it give a feeling of recursive implementation? (Y/N).

263. For the above problem in question 262, the following code is suggested. Will it work? (Y/N).

```
int f(int a, int b){
  if(a>=b) return f(a-b, b);
  else
  return a;
  }
```

264. What is the output of the following recursive function a and b values of 10 and 3?

```
int f(int a, int b){
  if(a>=b) return (1+f(a-b, b));
  else
  return 0;
  }
```

A. Error B. Infinite loop

C. 3 D. 1

265. In 32-bit computers, pointer variables occupies __ bytes.

A. 2 B. 4 C. 6 D. No

266. Things associated with any variable

A. Its memory location B. Value in the memory

C. Both D. None

267. A variable becomes pointer type if

A. We use * before it in its declaration.

B. We use ** before it in its declaration.

C. We use *** before it in its declaration.

D. All

268. We can use any number of *'s before a variable in its declaration. (Y/N).

269. If we declare only an integer variable and pointer variable in a declaration statement on a machine (which uses 4 byte integers and 4 byte pointer), the difference between their addresses is

A. 2 B. 4 C. 8 D. None

270. Does the following statements get executed? (Y/N)

```
int *p;
printf("%d\n", *p);
```

271. Find in correct one

A. A variable name can be lvalue of an expression involving =.

B. A pointer variable name can be lvalue of an expression involving =.

C. A dereferencing operator applied to a pointer variable can be lvalue of an expression involving =.

D. None

272. A properly defined pointer variable, i.e., for which allocated memory cell number is already assigned, can be used in the scanf statement after format string. (Y/N).

273. Find out odd one out of the following statements assuming an integer variable, a, whose value is already initialised to 10.

A. printf("%d\n", a);

B. printf("%d\n", *&a);

C. printf("%d\n", *&*&*&*&*&*&a);

D. printf("%d\n", *&*&*&&*&*&a);

274. We have an integer pointer variable for which an allocated memory cell number having some meaningful value is assigned. Then, find out odd one out of the following

A. printf("%d\n", p);

B. printf("%d\n",*p);

C. printf("%d\n", *&*&*&*&*&*p);

D. printf("%d\n",*&*p);

275. When we assign a pointer variable, p, value to another pointer variable, q, then whenever p values changes, q also changes. (Y/N).

276. When we assign a pointer variable, p, value to another pointer variable, q, then whenever value changes in the memory pointed by p, q also see the changes. (Y/N).

277. We cannot return address of a scratch variable from a function. Any attempt of dereferencing of such address leads to memory segment violation. (Y/N).

278. Life of the dynamic memory

A. In the block in which it is allocated

B. Entire life of the program

C. Till we free it

D. None

279. What is the output of the following code fragment?

```
int a=10, b=12, c=20;
int *p=&a;
p--;
p--;
printf("%d\n",*p);
```

A. 10 B. 12

C. 20 D. None

280. What is the output of the following code fragment?

A. 10 12 20 B. 10 12 14

C. Compilation error D. None

281. Out of the following, which can print the values of a,b and c correctly.

int a=10, b=12, c=20, *p=&c;

A. printf("%d %d %d\n",*p, *p++, *p++);

B. printf("%d %d %d\n",*p++, *p++, *p);

C. printf("%d %d %d\n",*&*p, *p++, *p++);

D. None

282. What is the output of the following fragment?

```
int a=10, b=12, c=20, *p=&c;
printf("%d\n", *(++(p=(&(*(++p))))) );
```

A. 10 B. 12 C. 20 D. Error

283. Dangling memory

A. Memory which is allocated using dangling function instead of malloc.

B. Memory which is allocated and freed in the same block.

C. Memory which is allocated in a function and control returns from the function with its starting address such that this memory is accessed outside.

D. Memory which is allocated in our account but we don't have freedom to use the same.

284. The function which allocates memory and initialises also.

A. malloc B. free C. realloc D. calloc

285. Dangling pointer

A. Is the one which points to memory which is not currently allocated to us

B. Gives rise a run time error if we try to apply dereferencing operator

C. Is the one which points to stack memory

D. None

286. For a pointer variable (such as int *a), a machine allocates 4 bytes. Then, amount of memory allocated for int ****p;

A. 16 B. 4 C. 8 D. None

287. Memory allocated for the following two pointer variables is same on any machine? (Y/N).

int *****p;

char *****q;

288. The malloc function can be asked which type of array to be created. (Y/N).

289. What is the output of the following code fragment?

```
int a[]={10,12,22,21,31}, *p=a;
printf("%d\n", *(++(p=(&(*(++p))))) );
```

A. 10 B. 12
C. 22 D. 21

290. What is the output of the following code fragment?

```
int a[]={10,12,22,21,31}, *p=a;
++(p=(&(*(++p))));
printf("%d\n", *(++p) );
```

A. 10 B. 12 C. 22 D. 21

291. Is it possible to read a value to a pointer variable through scanf? (Y/N)

292. Language supported arrays names are constant pointers. (Y/N)

293. If a variable a is declared as an integer array and some values are assigned like int a[]={10,12,22,21,31}, then

A. a++ is not valid
B. ++a is not valid
C. printf("%d\n", *(a+4)) is valid
D. All

294. The output of the following code fragment is

A. 12 B. 22
C. 31 D. 21
E. Error

295. If a is an integer pointer to a dynamically created memory on a machine which uses 4 bytes for an integer. In the array, we have stored 0,1,2,3, and vice versa. We observed that a value as 1000. Then, a+4

A. 1004 B. 4
C. 1016 D. None

296. Find the correct one

A. If a variable is sent to a function in passing by value manner, whatever operations on its formal argument counterpart will be really seen on the actual one.
B. If a variable is sent to a function in passing by value manner, whatever operations on its formal argument counterpart will not be really seen on the actual one.
C. Passing by address involves sending values in the memories.
D. None

297. In passing by address, the function refers to the memory of the variables whose addresses are sent, and values in those addresses are manipulated. (Y/N)

298. When can a function call can be used on the left hand side of an equality?

A. If it returns an integer
B. If it return float
C. If it returns void
D. If it returns address

299. Which approach can be used to create a dynamic 2-D array of exactly required size.

A. Malloc approach
B. Pointer to pointer approach
C. Array of pointers approach
D. None

300. When can we a function has a return type of int **?

A. If it returns an integer
B. If it returns address of an integer
C. If it returns address of an integer pointer
D. If It returns the starting address of dynamically created integer pointer array.

301. Is it possible to de-allocate memory of language supported arrays using free method? (Y/N).

302. Advantage of dynamic arrays

A. Memory management will be in the hands of programmer
B. We can create an array of required size only.
C. Programs become flexible
D. All

303. A character pointer variable (which is pointing to 'M') is subjected to postfix increment operator. Its value after increment

A. Incremented by one B. 'N'
C. N D. None

304. If p, q are two integer pointers then

A. p=p+q is legal B. p=p-q is legal
C. p=P*q is legal D. p=p/q is legal
e. All are illegal

305. If p is an integer pointer then we cannot add to p

A. A positive integer constant or variable
B. An integer/long/char constant or variable
C. An octal/hexadecimal constant
D. A float/double constant

306. If two character pointers are declared and initialized to same string constant (like the following), then their values (addresses) are same. That, both will be pointing to same memory locations. (Y/N).

```
char *a="Ram", *b="Ram";
printf("%p %p\n", a, b );
```

307. If two character pointers are declared and initialized to same string constant (like the following), then their addresses are same? (Y/N).

```
char *a="Ram", *b="Ram";
printf("%p %p\n", &a, &b );
```

308. We have stored same integer constants in the memories pointed by the two pointer variables soon after their declaration (see below code). Does it runs? (Y/N).

```
int *a, *b;
*a=12;
*b=12;
```

309. See the following declaration statements. Assume we have an integer pointer p for which variable c address is assigned. Is it possible to access variable a value using pointer variable p? If so, how?

```
int a=10, b=12, c=20;
int *p=&c;
```

310. Not a user defined variable

A. int B. Bit fields
C. struct D. union

311. Padding is not seen in the unions. (Y/N)

312. Memory allocated for a structured pointer variable in a 32-bit computer is

A. 2 B. 4
C. 6 D. 8

313. A structured variable occupies same memory irrespective of the processor. (Y/N)

314. Memory allocated for the members of a structured variable is always consecutive. (Y/N)

315. Is it possible to define a structure inside the main. (Y/N)

316. Is it possible to define a structure in any function? (Y/N)

317. hat is the output of the following program?

```
struct ABC{
int a;
int b;
};
int main(){
  struct ABC X,Y;
printf("%d\n", &X - &Y );
}
```

A. 8 B. 2 C. 1 D. None

318. See the following code fragment. What will be difference in the addresses of X and Y?

```
struct ABC{
int a;
int b;
}X,Y;
```

A. 8 B. 2 C. 1 D. None

319. What is the output of the following code?

```
struct ABC{
int a;
int b;
}X={10,22}, Y={82,121};
int main(){
int *p=&X.a; p++; p++;
printf("%d\n", *(p) );
}
```

A. 10 B. 12 C. 82 D. 121

320. A structure contains a 20-element character array, two integers and a double. The compiler running on this 64-bit computer uses 4 and 8 bytes for integer and double. What will be the increment value for a structured pointer variable of this type?

A. 4 B. 36 C. 16 D. 40

321. Memory for two structured variables are allocated through malloc function. Does the memory allocated for both is guaranteed to be consecutive? (Y/N)

322. Two structured variables are declared. Does the memory allocated for both is guaranteed to be consecutive? (Y/N)

323. Is the following structure definition acceptable?

```
struct ABC{
int a;
struct ABC X;
} ;
```

324. Is it possible to include a function definition inside a structured definition?

325. Standard output operator

A. > B. >> C. < D. None

326. Number of filenames which standard output operator can take.

A. 1 B. 2 C. 0 D. None

327. Standard output re-direction operations will work under Windows also. (Y/N)

328. Main function can take __ arguments.

A. 2 B. 3 C. 2 or 3 D. None

329. Third argument to main can be

A. Int **a B. Char *a[]
C. Char **a D. None

330. Program name

A. First command line argument
B. Will not be list of command line arguments
C. Availability in the list of command line arguments depends on the HW
D. None

331. The function that can be used to manage command line arguments

A. strtok B. main

C. getopt D. None

332. Command line arguments are stored

A. As strings

B. As integers if we enter integers

C. As float if we enter floats

D. None

333. We propose to enter a series of float numbers as command line arguments then first and second arguments of main method are

A. float, float** B. int, float**

C. int, char** D. int, char *[]

334. Does the following style of main gets compiled and executed? (Y/N)

int main(int N)

335. Default streams available for any program

A. Standard input B. Standard output

C. Standard error D. All

336. File descriptors are meaningful in Unix only. (Y/N)

337. Which programs are portable, FILE * based or file descriptors based?

338. The style of I/O in which a variable will occupy same space on disk as that of its space in RAM.

A. Formatted I/O B. Unformatted I/O

C. Binary I/O D. Both

339. On 32-bit machine, value of sizeof (FILE *)

A. 2 B. 4 C. 8 D. None

340. Two file which occupies consecutive areas on the disk are made pointed by two FILE* type variables. Then, these pointers will be also occupying consecutive locations in RAM. (Y/N)

341. A fopen call fails

A. If the file does not exist.

B. If we do not have permissions on the file.

C. If we supply illegal filename.

D. All

342. In order to copy content of a file to another file, the following solution is proposed. Will it work? (Y/N)

```
FILE *A, *B;
A=fopen("XXX", "r");
B=fopen("YYY", "w");
while(!feof(A) && fputc(fgetc(A),B));
```

343. The function which is used to move pointer in a file

A. fseek B. ftell

C. fmove D. fskip

344. We proposed to traverse a file byte by byte. Thus, the following is suggested where a character pointer is used to traverse. Will it work? (Y/N)

```
FILE *A;
char *p;
A=fopen("XXX", "r");
p=(char*)A;
while(*p)printf("%c", *p++);
```

345. Numbers 0, 1 and 2

A. Used as third argument in fseek

B. Refers to standard input, standard output, and standard error

C. Both

D. None

346. To use FILE *

A. We need special header files

B. We need special libraries

C. We need stdio.h and standard library

D. None

347. How many files can we open simultaneously using fopen depends on the operating system. (Y/N)

348. Use of ferror function is

A. To correct I/O errors

B. To recover from I/O errors

C. To find I/O errors

D. No such function exists

349. Find correct one

A. Formatted I/O is used for hard copy purpose

B. Unformatted I/O is used to conserve space.

C. Both

D. None

350. Bitwise operators are also called as low level operators. (Y/N)

351. Bitwise operators are the fastest operators. (Y/N)

352. Which bit position upper case and lower case characters differs?

A. 3 B. 4

C. 5 D. None

353. To double an unsigned integer

A. Left shift by one bit B. Set MSB

C. Set LSB D. Right shift by one bit

354. The bitwise operator used for parity checking

A. OR B. AND

C. << D. XOR

355. We have two properly initialised unsigned char variables x and y. What happens if we apply the following operators: x=x^y; y=x^y; x=x^y;

A. There is no operator ^

B. x and y values gets exchanged.

C. x and y values become 0s.

D. None

356. We wanted to check whether the given number is even or odd using a bitwise operator. Thus, we apply bitwise AND with 1. Will it work? (Y/N)

357. A digit is stored in a character variable. To digit value from character value

A. Extract Four LSB bits

B. Extract Four MSB bits

C. Both

D. Mask with 15

358. The following code fragment is suggested to calculate the binary code length of the given number. Will it work? (Y/N)

```
int n,p=0;
scanf("%d", &n);
while(p++,n=n/2);
printf("%d\n", p);
```

359. The following code fragment is proposed to calculate parity of an integer. Will it work? (Y/N)

```
int n,p=0;
scanf("%d", &n);
while(p^=n%2,n=n/2);
printf("%d\n", p);
```

360. Which of the following scanf statement is valid way of reading data into an integer variable l?

A. scanf(“%d”, l)

B. scanf(“%d”,**&l);

C. scanf("%d",&*&*&l);

D. scanf(“%d”,&*&*l);

361. While reading a value to an integer variable (l) through scanf function, second argument of scanf can be a series of &* with one & before the variable name (l) like &*&*&*&l. (Y/N)

362. While displaying value of an integer variable (l) using printf statement, second argument of printf can be a series of *& before the variable name (l) like *&*&*&l (Y/N)

363. printf (“%d\n”, sizeof sizeof sizeof sizeof sizeof 2+7) gives

A. 4 B. Error C. 11 D. 9

364. Assuming that the following program is syntactically correct. What parameter passing convention would be in force if the program prints 6?

Program test (input, output);

```
Var I,J: integer;
Procedure calc (p1, p2 :integer);
Begin
p2:=p2*2; p1:=p1+p2; p2:=p2-p1;
End; {calc}
Begin {main}
I:=2; J:=3; calc(I, J); Writeln(J)
End.        {main}
```

A. pass by value B. pass by address

C. pass by reference D. None

365. In _____ parameter passing method the actual argument has to be a variable.

A. pass by value B. pass by reference

C. pass by result D. pass by value-result

366. Consider the following Pascal program.

```
Procedure XYZ(A,B,C:integer);
Begin
B :=B-4; C := A+C
End;
Var A,B: ineteger;
A :=10; B:=20; XYZ(A,A,A);
Write (A);
End
```

If this program prints 12 then A,B,C should have been declared as:

A. All are variable parameters

B. Only A, B are variable parameters

C. Only A, C are variable prarameters

D. Only B,C are variable parameters

367. In ths above XYZ if we include write(C); statement before end and if the result is 12, 6 then

A. All are variable parameters

B. Only A, B are variable parameters

C. Only A, C are variable prarameters

D. Only B,C are variable parameters

368. In which of the following cases it is possible to obtain different results for call-by-reference and call-by-name parameter passing?

A. Passing an expression as a parameter

B. Passing an array as a parameter

C. Passing a pointer as a parameter

D. Passing an array element as a parameter

369. Late binding or lazy evaluation

A. Pass by value

B. Pass by result

C. Pass by value result

D. Pass by name
E. Pass by reference

370. Find odd man out. In terms of 0-origin indexing
A. C B. C++
C. Object C D. Ada

371. In C, a function return _____ (Find incorrect ones)
A. An integer B. A 1D array
C. Address of a function
D. Address of a strcture

372. If X is an integer variable, find out which of the following statement gives error
A. X= -2; B. X=- -2; C. X=--2; D. –x;

373. What is the result of the following program?
```
int a[2][2][2] = { 10,2,3,4, 5,6,7,8 };
printf("%d",***a);
```
A. Error B. 10
C. Address of the array D. None

374. Find invalid C statement.
A. 100; B. 100+2;
C. 0<1,2&&3; D. None

375. Is it possible to jump to a statement of a function from another function? (Y/N)

376. Find odd man out of the following with respect to a program's available files by default.
A. stdin B. stdout
C. stderr D. None

377. Assuming that the integer variable i value is 5 initially. Which of the following statements makes value as 12?
A. i=i+++i B. i=i--+i
C. i=--i+i--; D. i=i+++i++;

378. What is the value returned from the function ff() if we send 5 as the argument?
```
int ff(int a){
return ++a,++a,a++,++a,a++;
}
```
A. 6 B. 9 C. 5 D. 6

379. What will be the output of the following program?
```
int main() {
static int i=5;
if(--i){
printf("%d\t",i);main();
}
return 0;
}
```
A. 0 0 0 0 B. 5 4 3 2 1
C. Stack overflow error D. 4 3 2 1

380. See the following C code fragment.
```
int r, c, s;
printf("Input a full number between -10 000 and
10 000:");
scanf("%d",&r);
c = 1;
c = r * c;
s = 1;
if (c <= r) {
s = s + c;
c = c * 2;
}
printf("the variable s is %d\n",s);
```
Which of the following statements is true?
A. The sum of all numbers from 1 to r + 1 is shown.
B. The value r + 1 is shown.
C. The value r + 1 is shown only if r >= 1.
D. The value 2r + 1 is shown.

381. What will be the output of the following program?
```
int succ(int i) {
if (i <= 2)
return(1);
else
return(3*succ(i - 1)+2*succ(i - 2)-succ(i - 3));
}
int main() {
printf("num=%d\n",succ(7));
return 0;
}
```
A. 554 B. 559
C. 567 D. None

382. Find invalid statements syntactically
A. switch(5/4/3) { ……}
B. #define X 0
switch(some valid expr){ case X+1: …..; break;…..}
C. int X=0; switch(some valid expr){ case X+1: …..; break;…..}
D. const X=0; switch(some valid expr){ case X+1: …..; break;…..}

383. Find invalid statement
A. Case expression should be always a character constant
B. Case expression should be always a integer constant

C. Case expression involving a symbolic constants only
D. Case expression should be always a symbolic constant
e. None

384. Find odd man out of the following
A. stdin B. cin
C. System.in D. 1
E. None

385. Find odd man out of the following
A. stdin B. 1
C. – D. None

386. Default file descriptors available for any process
A. 0 B. 1
C. 2 D. All a, b, and c
E. 3

387. What is the value of y after the last statement?

```
x = 3;
y = 3;
switch(x + 3) {
case 6: y = 1;
default: y += 1;
}
```

A. 1 B. 2 C. 6
D. Syntactical error in switch

388. Find incorrect statement about malloc() function
A. It allocates contiguous memory
B. If fail it returns -1
C. If an array is created using malloc in a function, if we return its address then other functions can use this memory.
D. Dangling pointer value will be 0.

389. Dangling pointer
A. Give raises memory segment error
B. Appears if malloc() function fails and returns -1
C. Is same as void or generic pointer
D. None

390. In HEX: if 08 00 00 00 = 8 and 05 00 00 00 = 5 then what does 61 02 00 00 indicates?
A. 509 B. 609
C. 690 D. None

391. In-correct way of referring an element of an array B
A. B[1] B. 1[B]
C. *(B+1) D. *&B[1]
E. None

ANSWER KEY

1. D	**2.** C	**3.** D	**4.** C
5. C	**6.** D	**7.** B	**8.** C
9. B	**10.** C	**11.** A	**12.** D
13. D	**14.** C	**15.** B	**16.** B
17. B	**18.** C	**19.** C	**20.** C
21. C	**22.** C	**23.** A	**24.** B
25. D	**26.** D	**27.** D	**28.** C
29. C	**30.** D	**31.** D	**32.** B
33. D	**34.** B	**35.** B	**36.** B
37. B	**38.** E	**39.** D	**40.** B
41. B	**42.** B,C	**43.** B	**44.** C
45. B	**46.** No	**47.** B	**48.** C
49. C	**50.** No	**51.** Yes	**52.** Yes

53. Yes. We cannot include an executable statements in between declaration statements

54. A	**55.** A	**56.** A	**57.** A
58. D	**59.** B	**60.** A	**61.** Yes
62. Yes	**63.** C	**64.** A	**65.** C
66. B	**67.** C	**68.** A	**69.** A
70. C	**71.** C	**72.** B	

73. No. as some blood groups such as O+ etc., needs two symbols.

74. Yes	**75.** Yes	**76.** Yes	**77.** No
78. Yes	**79.** C	**80.** C	**81.** C
82. C	**83.** C	**84.** No	**85.** C
86. C	**87.** A	**88.** No	**89.** No
90. C	**91.** No	**92.** Yes	**93.** C
94. No	**95.** No	**96.** No	**97.** No
98. C	**99.** Yes.	**100.** A	

101. Yes it works. However, it displays the given number also.

102. Yes. We get error on else. This is, because of; after in if statement.

103. No. Most of the compilers gets confused after third level nesting.

104. Yes	**105.** Yes	**106.** D	**107.** B
108. D	**109.** D	**110.** A	**111.** C
112. A	**113.** A	**114.** A	**115.** C
116. D	**117.** C	**118.** C	**119.** C
120. C			

121. Scanf function returns 1 when we enter an integer. Thus, while condition becomes true. Thus, loop runs for infinite times. Rather it will go on take numbers. We can brake only through control + C.

122. D (This is only one which runs for four times).

123. B (remaining all are infinite loops)

124. C	**125.** Yes	**126.** Yes	**127.** Yes
128. C	**129.** Yes	**130.** C	**131.** A
132. D	**133.** No	**134.** Yes	**135.** Yes
136. C	**137.** B	**138.** No	**139.** Yes
140. C	**141.** C	**142.** Yes	**143.** B
144. No	**145.** B	**146.** D	**147.** B
148. B	**149.** D	**150.** D	**151.** A
152. C	**153.** C	**154.** Yes	**155.** Yes
156. B	**157.** Yes	**158.** Yes	**159.** Yes
160. Yes	**161.** No	**162.** Yes	**163.** C
164. No	**165.** C	**166.** C	**167.** A
168. D	**169.** D	**170.** No	**171.** C
172. B	**173.** C	**174.** Yes	**175.** No
176. A	**177.** No	**178.** No	**179.** Yes
180. No	**181.** Yes	**182.** B, C	**183.** C
184. C, D	**185.** C	**186.** D	**187.** Yes
188. A	**189.** B	**190.** C	**191.** A
192. Yes	**193.** No	**194.** Yes	**195.** No
196. B	**197.** No	**198.** D	**199.** B, c
200. B	**201.** C,D	**202.** A	**203.** A
204. No	**205.** C	**206.** A	**207.** Yes
208. A	**209.** Yes	**210.** D	**211.** No
212. C	**213.** C	**214.** Yes	**215.** Yes
216. No	**217.** B	**218.** C	**219.** A,B
220. No	**221.** Yes	**222.** Yes	**223.** C
224. Yes	**225.** C	**226.** B	**227.** Yes
228. C	**229.** C	**230.** C	**231.** A
232. B	**233.** No	**234.** No	**235.** Yes
236. No	**237.** C	**238.** C,D	**239.** Yes
240. A	**241.** C	**242.** C,D	**243.** Yes
244. D	**245.** Yes	**246.** Yes	**247.** D
248. A,B	**249.** C	**250.** Yes	**251.** No
252. No	**253.** Yes	**254.** Yes	**255.** Yes
256. Yes	**257.** Yes	**258.** No	**259.** No
260. A,B	**261.** D,E	**262.** Yes	**263.** Yes
264. C	**265.** B	**266.** C	**267.** D
268. Yes	**269.** B	**270.** No	**271.** D
272. Yes	**273.** D	**274.** A	**275.** No
276. Yes	**277.** Yes	**278.** B,C	**279.** C
280. A	**281.** A,C	**282.** A	**283.** D
284. 21	**285.** A,B	**286.** B	**287.** Yes
288. No	**289.** C	**290.** D	**291.** No
292. Yes	**293.** D	**294.** C	**295.** C
296. B	**297.** Yes	**298.** D	**299.** B
300. C,D	**301.** No	**302.** D	**303.** A
304. E	**305.** D	**306.** Yes	**307.** No

308. No

309. Increment p value by two times before applying de-referencing operator.

310. A	**311.** Yes	**312.** B	**313.** No
314. Yes	**315.** Yes	**316.** Yes	**317.** C
318. A	**319.** C	**320.** B,D	**321.** No
322. Yes	**323.** No	**324.** Yes	**325.** A
326. A	**327.** Yes	**328.** C,D	**329.** B,C
330. A	**331.** C	**332.** A	**333.** C
334. Yes	**335.** D	**336.** Yes	**337.** file*
338. B,C	**339.** B	**340.** No	**341.** D
342. Yes	**343.** B	**344.** No	**345.** C
346. C	**347.** Yes	**348.** C	**349.** D
350. Yes	**351.** Yes	**352.** C	**353.** A
354. D	**355.** B	**356.** Yes	**357.** A, D
358. Yes	**359.** Yes	**360.** D	**361.** Yes
362. Yes	**363.** C	**364.** C	**365.** B,C,D
366. B	**367.** B	**368.** A	**369.** D
370. D	**371.** B,C	**372.** C	**373.** B
374. D	**375.** N	**376.** D	**377.** D
378. B	**379.** D	**380.** B	**381.** B
382. C,D	**383.** E		

384. E(All are related to standard input stream)

385. D(All are related to standard input stream)

386. D

387. B(First, case 6 will be executed then default statements also executed as break is not used)

388. B **389.** D

390. D (Given examples indicates that the numbers are given in big-endian style machines. Thus, HEX 61 02 00 00 can be visualized normally as 00 00 02 61 = $2*16^2 + 6*16^1 + 1*16^0 = 609$).

391. E (All the given styles are acceptable)

2.4 Data Structures and Algorithms

We now come to the study of Data Structures and Algorithms. These include Analysis, asymptotic notation, notions of space and time complexity, worst and average case analysis; asymptotic analysis (best, worst, average cases) of time and space, upper and lower bounds, basic concepts of complexity classes P, NP, NP-hard, NP-complete, Stacks, queues, linked lists, trees, binary search trees, binary heaps.

Design: Greedy approach, Dynamic programming, Divide-and conquer; Tree and graph traversals, Connected components, Spanning trees, Shortest paths; Hashing, Sorting, Searching.

2.4.1 Comparing Algorithms: Complexity Theory

An **algorithm** may be defined in simple terms as a finite sequence of instructions that solves a problem. A computer program is simply an implementation of an algorithm on a computer. Evidently, there can be a number of algorithms to solve a given problem. Thus, an immediate question that arises is "which is better?" This necessitates the relative **analysis of algorithms** and this area of study is called as **Complexity Theory** in Computer Science.

One way to compare algorithms is to compare their performance in terms of how quickly they solve the problem. Another way of comparing algorithms is to look at the amount of space (memory) they require. Some algorithms require large amounts of space but arrive at a solution quickly. Others require small amounts of space but arrive at a solution less quickly. This behaviour is referred to as **space/time trade-off**. That is, it is observed with many algorithms that if we try to reduce its CPU time requirements it demands more memory; if we try to reduce its memory requirements it becomes computationally intensive.

Algorithm performance, that is **time complexity of an algorithm**, is the relative amount of time an algorithm takes to solve a problem. When we speak of the time complexity we are **not** interested in **absolute** times, i.e. how many seconds it takes to solve a particular problem. The actual absolute time taken to solve a problem depends on a number of factors: how fast the computer is, the quality of code generated by the compiler, the number of users using the computer at that time, etc. If you change any of these, then the absolute time changes.

Thus, absolute time is not useful as a measure of an algorithm's performance.

One reason for computing complexity is to compare algorithms. Thus we need a measure which will allow us to compare two algorithms. Each algorithm consists of a finite sequence of instructions. The more instructions in an algorithm the longer it will take to execute. Thus one way to compare algorithms would be to count the instructions that the algorithm requires to solve a problem. Rather, we compute the number of instructions as a function of the input size.

When comparing algorithms it is important to know both the average and worst case complexity.

Infeasible Algorithms

There are many algorithms whose complexity is such that for even small values of n, they cannot be feasibly used.

Algorithms with complexity O(**n!**) (factorial n) or O(**2n**) (exponential complexity) are **infeasible**.

One such algorithm is one for the **travelling salesman problem.**

A salesman has to travel to say, 50 cities. He wishes to take the cheapest (perhaps shortest) route whereby he visits each city only once. One simple algorithm is to compute all possible routes and pick the cheapest. But how many routes are there?

There are 50! possible routes. This is a 65 digit number, and would take billions of years to compute on a computer. If there were 300 cities then we would have a 600 digit number. (So what ! Well the number of protons in the universe is a 126 digit number).

Thus we can see that the algorithm is infeasible for even small values of n.

It is important to note that this problem occurs frequently in different guises, e.g. laying telephone cables (roads, rail tracks) between a number of towns. A small saving in distance can result in huge financial savings. It is an example of a routing problem.

Another example of an infeasible problem might be a process control situation in a power plant. There are 50 sensors around the plant connected to a computer. For simplicity, we assume each sensor sends a binary signal to the computer, indicating normal/abnormal conditions.

We are interested in testing that the computer program will behave correctly with every possible combination of inputs from the sensors.

Well how many combinations are there? There are 50 sensors each providing yes/no responses thus there are 2^{50} possible combinations. If we carried out 1 instruction per microsecond it would take over 35 years to compute all solutions.

If we had 60 sensors, it would take 366 centuries to compute.

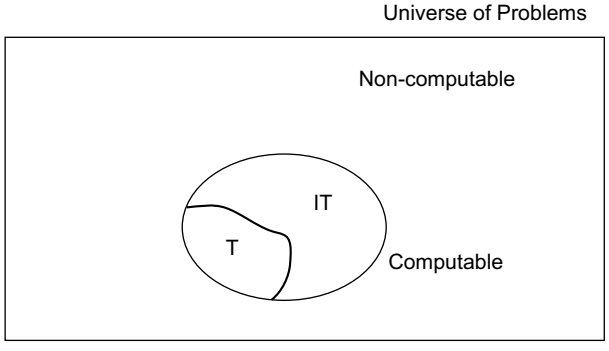
This is another infeasible problem. This last example illustrates another important issue. We cannot test complex computer programs to check if they will work with all possible inputs. We can only partially test programs on a very small subset of the possible inputs. It is also very important to remember that testing in itself does not prove anything. Just because a program works for some test data, in no way proves the correctness of the program. The question arises then as to how we deal with infeasible problems. One approach is to develop algorithms that are not guaranteed to give the best answer, but will on average give a good answer.

Computability

In the field of computability, we are interested in studying what can and cannot be computed. We have seen above some problems that were computable but they were infeasible, i.e. it would take an impossible amount of time to compute the results.

But there are also problems, in fact the majority of problems, which cannot be computed. The computable problems form a very small subset of the universe of problems. The set of feasibly computable problems forms a small subset of the computable problems as shown in the Fig. 2.5.

It is important to distinguish infeasible and non-computable problems from feasible problems, so that we do not waste time seeking solutions that either do not exist, or if they do exist, they are useless.



T : Tractable or Feasibly Computable
IT : Intractable or Infeasible

Figure 2.5

A Non-Computable Problem: Halting Problem

A common error that programmers make is to write programs that contain endless loops, i.e. they do not terminate (halt). When such a program is executed, it must be interrupted by the user to halt it, when the user realises after a while that the program is not going to terminate.

It would be very useful, if we could write a program or modify a compiler so that it could test if a given source program terminates. If we had such a compiler, then we would never execute a program with an endless loop.

The sad fact is that this problem is non-computable. It is impossible to write a program, which given another program P as input, will determine whether P terminates. This problem is known as the halting problem.

Informally, we can show that the halting problem is non-computable as follows.

Assume there exists a function Halt() which takes an arbitrary program P as a parameter. Function Halt() is defined to return True if P halts and False if P does not halt e.g.

```
if Halt( P ) then printout('P halts')
else printout('P does not halt');
```

Consider the following program F which takes an arbitrary program P as input and uses the function Halt() as follows:

```
Label1:     if Halt( P ) then goto Label1
            else Stop ;
```

This program loops forever if program P does halt, i.e. Halt(P) returns True and it stops only if P does not halt.

What happens with program F if we use program F itself as input, i.e. as the parameter to Halt(). F now takes the form:

```
Label1:     if Halt( F ) then goto Label1
            else Stop ;
```

which says that if F halts then it loops forever and if F does not halt it stops !

We thus have a contradiction which is due to our assumption that a function Halt(P) could computed. We thus conclude that the halting problem is non-computable.

Computability is explored in the area of Computer Science called **Theory of Computation.**

Correctness

Most programs in current use contain errors (bugs). Errors in computer programs have serious consequences such as the loss of a spaceship sent to Venus. This is just one of many publicised errors. However, there are very many more unpublicised ones. Every (honest) programmer will recount tales of silly, funny and serious errors they have made in their programs.

It is estimated that as much as 70% of the effort and cost in constructing complex software systems is devoted to error correcting. This may be due to poor program specifications (imprecise, vague, ambiguous problem definitions), extensive debugging to find errors and worst of all rewriting large parts to correct bugs. The later an error is discovered, the more costly it is to correct.

It is important to be aware that compilers and operating systems also have bugs. It is quite common that new versions of compilers and especially operating systems are regularly released, which correct bugs in previous versions. Sadly, the new versions frequently introduce new bugs which in turn are corrected in later versions and the cycle continues.

Thus the importance of writing correct programs cannot be overstated. Beginners, often believe that their algorithms do precisely what they intend them to do. This has no justification. A disciplined or methodological approach to programming must be adopted if errors are to be eliminated. Computer Science courses in Programming Methodology, Formal Specifications and Programming Design and Verification, deal with the notion of developing correct programs.

Methods of producing correct programs can be grouped in two categories: testing and proving. Testing a program consists of executing a program on test data in order to discover the output of the program for that data. The important feature is that the outcome is only tested for that particular set of test data.

Proving a program correct means that the program is correct for all permissible input data. This is obviously a much more desirable property. (It is important to make sure that there are no errors in the proof).

Program testing has been more widely used because it is a much easier technique to apply, requiring less thought than proving. Informal proofs are often used to reason about a program's behaviour. They can increase our confidence in the correctness of a program. Informal proofs can be made more formal and detailed, but this is more difficult and tedious. However, in some situations where the application is very crvcial, a formal proof of correctness may be required.

The method of proof by induction is primarily used in proving the correctness of a program.

What does correctness mean?

It is meaningless to speak of program correctness in isolation, it must be related to the purpose of the program. "The program produces the right answer" is meaningless unless we can specify what is meant by "right" in any particular situation. The purpose of a program is provided in the form of the problem specification, which defines (precisely, unambiguously and clearly) the problem to be solved and the output to be produced.

A program is correct with respect to its specifications if it produces the results specified for those input values defined by the specification. There are a number of facets to correctness. We speak of partial correctness, when we know that if a program terminates it will always produce the correct result, but we do not know if it will always terminate. We speak of total correctness if in addition to always producing the correct result, the program always terminates. Different techniques can be used to prove partial and total correctness. Another facet of correctness is feasibility, which was discussed earlier. Will the program use a reasonable amount of resources, e.g. finish is a feasible time span. In conclusion, the ability to reason about program correctness and produce correctness proofs is likely to become increasingly important in the future. Already, the major cost in computing systems is the development of software. The cost of software is in terms of programming effort. Techniques which can significantly improve programmer productivity will become more important. In addition, the requirement that software in important applications (process control in industry, life critical applications) be formally correct is likely to become a standard requirement.

Time Complexity analysis of algorithms

We have explained the reasons for the study of algorithms in the above paragraphs. There are several factors affecting the running time :

- Computer
- Compiler

- Algorithm
- Input to the algorithm

The *content of the input* affects the running time

Typically, the *input size* (number of items in the input) is the main consideration

- E.g. sorting problem ⇒ the number of items to be sorted.

For this course, it is generally assumed that instructions are executed one after another.

We use RAM (Random Access Model), in which each operation (e.g. +, –, x, /,=) and each memory access take one run-time unit. Loops and functions can take multiple time units.

Problem Size: Usually computational complexities are represented in terms of problem size. For example, in the case of sorting a set of elements, number of elements can be considered as problem size. Similalarly, while estimating the complexity of matrix multiplication problem, matrix size is taken. Thus, this is very much associated with the problem.

The time complexity of an algorithm, T(n), is represented as a function of its problem size, say n.

The time required is simply a count of the **primitive operations** executed. Primitive operations include

1. Assign a value to a variable (independent of the size of the value; but the variable must be a scalar)
2. Method invocation, i.e., calling a function or subroutine
3. Performing a (simple) arithmetic operation (divide is OK, logarithm is not)
4. Indexing into an array (for now just one dimensional; scalar access is free)
5. Following an object reference
6. Returning from a method

Algorithm inner Product

Input: Non-negative integer n and two integer arrays A and B of size n.

Output: The inner product of the two arrays

```
prod ← 0
for i ← 0 to n-1 do
    prod ← prod + A[i]*B[i]
return prod
```

- Line 1 is one op (assigning a value).
- Loop initialising is one op (assigning a value).
- Line 3 is five ops per iteration (mult, add, 2 array refs, assign).
- Line 3 is executed n times; total is 5n.
- Loop incrementation is two ops (an addition and an assignment)
- Loop incrementation is done n times; total is 2n.
- Loop termination test is one op (a comparison i<n) each time.
- Loop termination is done n+1 times (n successes, one failure); total is n+1.
- Return is one op.

The total is thus $1 + 1 + 5n + 2n + (n + 1) + 1 = 8n + 4$.

Improved inner product algorithm

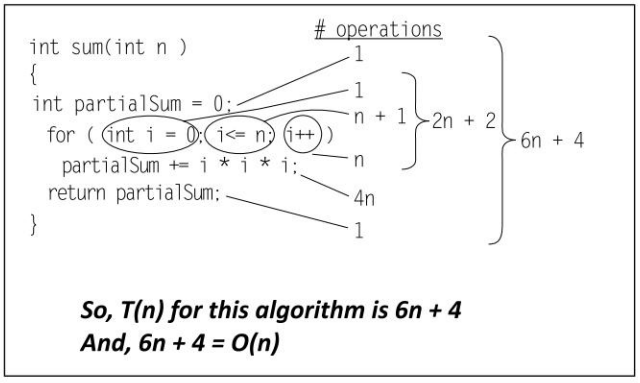
Input: Non-negative integer n and two integer arrays A and B of size n.

Output: The inner product of the two arrays

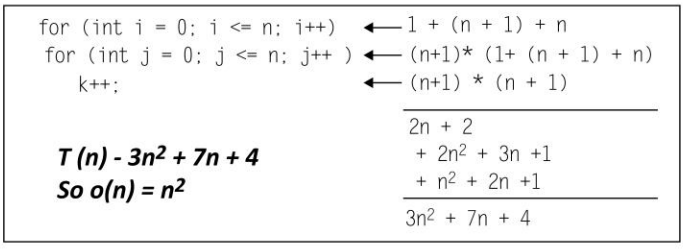
```
prod ← A[0]*B[0]
for i ← 1 to n-1 do
    prod ← prod + A[i]*B[i]
return prod
```

The cost is $4 + 1 + 5(n - 1) + 2(n - 1) + n + 1 = 8n - 1$

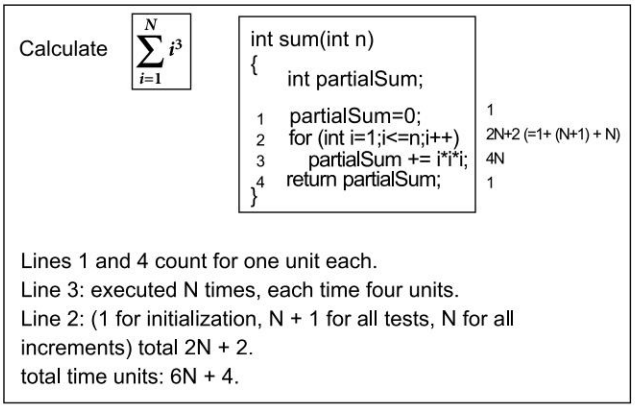
Similarly consider another example of a function call along with the operations involved.



Consider another example involving nested loops. Here, each loop runs for n+1 times. Thus, the inner most k++ statement executes (n + 1) * (n + 1) times.



Consider one more example where we wanted to calculate sum of the cubes of natural numbers between 1 to N.



Analysing Recursive Algorithms

We may employ recursive solutions also in practice. Thus, we do required to know how to estimate the time complexities of recursive solutions. Consider a recursive version of innerProduct. If the arrays are of size 1, the answer is clearly A[0] B[0]. If n>1, we recursively get the inner product of the first n-1 terms and then add in the last term.

Algorithm inner Product (Recursive)

Input: Positive integer n and two integer arrays A and B of size n.

Output: The inner product of the two arrays

if n=1 then

return A[0]B[0]

return innerProductRecursive(n-1,A,B) + A[n-1]B[n-1]

How many steps does the algorithm require? Let T(n) be the number of steps required.

- If n=1 we do a comparison, two fetches, a product, and a return.
- So T(1)=5.

- If n>1, we do a comparison, a subtraction, a method call, the recursive computation, two fetches, a product, a sum and a return.
- So T(n) = 1 + 1 + 1 + T(n-1) + 2 + 1 + 1 + 1 = T(n-1)+8.
- This is called a **recurrence equation.** In general these are quite difficult to solve in **closed form**, i.e. without T on the right hand side.
- For this simple recurrence, one can see that T(n)=8n-3 is the solution.

While comparing the algorithms we can use the above Time complexities. In fact, this can be carried out experimentally also. However, it is not always possible to run the programs and evaluate; especially for large problem sizes like human genome project. Thus, theoretical analysis is carried out widely.

■ **Example** Consider the following algorithm. What is return value from this function? As a function of *n*, what is the exact number of multiplications ("*") performed by the algorithm?

```
(1) s ← 1
(2) for i ← 1 to n do
(3)     if n is even then
(4)         for j ← 1 to n do
(5)             s ← s * 2
(6) else
(7)         s ← s * 2
(8) return s
```

■ **Answer:** For even n, the nested loops (lines 2 and 4) will execute n^2-times the multiplication in line 5. For odd n, the loop (line 2) will execute n-times the multiplication in line 7. Together this yields

$$f(n) = \begin{cases} n^2, \text{ if n is even} \\ n, \text{ otherwise} \end{cases}$$

As a function of n, the value of s returned in line 8 can be represented as:

$$s(n) = \begin{cases} 2^{n\wedge 2}, \text{ if } n \text{ is even} \\ 2^n, \text{ otherwise} \end{cases}$$

2.4.1.1 The Big-Oh Notation

Definition: Let f(n) and g(n) be real-valued functions of a single non-negative integer argument. We write f(n) is O(g(n)) if there is a positive real number c and a positive integer n_0 such that $f(n) \le cg(n)$ for all $n \ge n_0$ as shown in Fig. 2.6.

What does this mean?

For large inputs ($n \le n_0$), f is not much bigger than g ($f(n) \le cg(n)$).

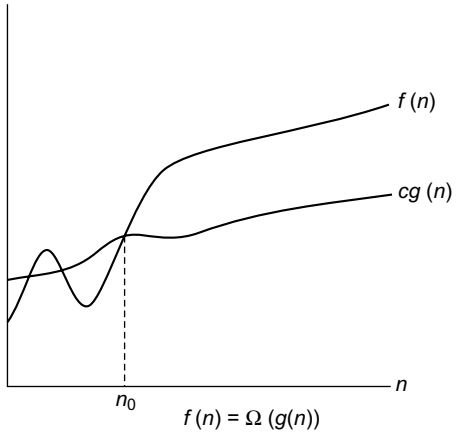


Figure 2.6 Big-Oh definition

To practically explain about this, consider a live problem.

Suppose a company tracks its autos that are shipped around the world on rail, truck, and water vessel. At the end of each day, this company uses an algorithm to summarise all auto movement in some meaningful way. Let us consider that the algorithm does its job in three steps

- The algorithm takes 50,000 msecs to read the data from the database
- The algorithm takes 1 msec to process each auto movement into summarised data
- The algorithm takes 5,000 msecs to write the summarised data back to the database

So, processing *n* auto movements takes

- (50,000 + 1*n + 5,000) msecs
- The n term will become more important as n becomes very large
- **As it turns out, in the real world, n may be in the order of 100,000,000 !!**

Let *n* be the input size (number of autos), thus T(*n*) becomes the time complexity of the processing method 50,000 + 1*n + 5,000.

Let f(*n*) be another function, preferably without constant factors… n, n^2, $\log_2 n$, etc., We can say that T(*n*) is Big-Oh of f(*n*), or, T(*n*) is on the order of f(*n*), or, T(*n*) = *O*(f(*n*)) if:

- T(*n*) ≤ *c**f(n) for some positive constant c, starting at the point where n is ≥ some other positive constant n0
- What we are saying is c*f(n) is bounding the function T(n) asymptotically
- Think of c * *f*(n) like a ceiling for T(n); in other words, we can guarantee that our algorithm will never run in worse time than on the order of *f*(*n*)

Given T(n)= (50,000 + 1*n + 5,000) msecs

T(n) = n + 55,000

Choose f(n) = n to see if T(n) = O(n)

Definition of Big-Oh: T(n) ≤ c*f(n)

n + 55,000 ≤ c*n ← solve for c

1 + 55,000/n ≤ c

As n gets bigger and bigger (and approaches infinity), 55,000/n will approach zero

1 + 55,000/1 ≤ c → c ≥ 55,001 if n = 1

1 + 55,000/2 ≤ c → c ≥ 27,500 if n = 2

1 + 55,000/∞ ≤ c → c ≥ 1 if n =3

We need to show this holds for positive constants c and n0 where n ≥ n0

- Pick n0 = 1
- 1 + 55,000/1 ≤ c , so c = 55,001
- Does this c = 55,001 still work for n = 2 (because n keeps growing)?
- 1 + 55,000/2 ≤ 55,001
- 27,501 ≤ 55,001 TRUE!

So, T(*n*) = *O*(f(*n*)) because we can find *c* and *n0* that hold as *n* grows to infinity. Thus, this algorithm's asymptotic complexity is said to be O(n).

Consider another example:

Suppose we have an algorithm that takes $3n^2$ steps given *n* inputs.

Does $3n^2 = O(n^2)$?

The definition of Big-Oh says :

- T(n) ≤ c*f(n) for c and n0 where n ≥ n0

Given: $3n^2 \leq c\, n^2$

Choose *n0* = 1 and *c* = 3

That works!

Does it still work as *n* grows? Now, let us try for *n* = 2

Yes, so $3n2 = O(n^2)$

To simplify things, we adopt the practice of throwing away leading constants and lower-order terms. We can do this because the highest order term (largest exponent) will dominate how the function grows. That way, we don't have to get too rigorous about finding *c* and *n*0.

■ **Example** $30n^2 + 5000n + 32{,}000$ becomes simply n^2, which is clearly $O(n^2)$

The following theorems give us rules that make calculating big-Oh easier.

Theorem (arithmetic): Let d(n), e(n), f(n), and g(n) be non-negative real-valued functions of a non-negative integer argument and assume d(n) is O(f(n)) and e(n) is O(g(n)). Then

ad(n) is O(f(n)) for any non-negative a

d(n)+e(n) is O(f(n)+g(n))

d(n)e(n) is O(f(n)g(n))

Theorem (transitivity): Let d(n), f(n), and g(n) be non-negative real-valued functions of a non-negative integer argument and assume d(n) is O(f(n)) and f(n) is O(g(n)). Then d(n) is O(g(n)).

Theorem (special functions): (Only n varies)

If f(n) is a polynomial of degree d, then f(n) is $O(n^d)$.

n^x is $O(a^n)$ for any x>0 and a>1.

$\log(n^x)$ is O(log(n)) for any x>0

$(\log(n))^x$ is $O(n^y)$ for any x>0 and y>0.

Relatives of the Big-Oh

Big-Omega and Big-Theta

Recall that f(n) is O(g(n)) if for large n, f is not much bigger than g. That is, g is some sort of **upper** bound on f. How about a definition for the case when g is (in the same sense) a **lower** bound for f?

Definition: Let f(n) and g(n) be real valued functions of an integer value. Then **f(n) is Ω(g(n))** if g(n) is O(f(n)).

Remarks:

1. We pronounce f(n) is Ω(g(n)) as "f(n) is big-Omega of g(n)".
2. What the last definition says is that we say f(n) is not much smaller than g(n) if g(n) is not much bigger than f(n), which sounds reasonable to me.
3. What if f(n) and g(n) are about equal, i.e., neither is much bigger than the other?

Definition: We write **f(n) is Θ(g(n))** if **both** f(n) is O(g(n)) **and** f(n) is Ω(g(n)).

Remarks We pronounce f(n) is Θ(g(n)) as "f(n) is big-Theta of g(n)"

Little-Oh and Little-Omega

Recall that big-Oh captures the idea that for large n, f(n) is not much bigger than g(n). Now we want to capture the idea that, for large n, f(n) is tiny compared to g(n).

If we remember limits from calculus, what we want is that $f(n)/g(n) \to 0$ as $n \to \infty$. However, the definition we give does not use limits (it essentially has the definition of a limit built in).

Definition: Let f(n) and g(n) be real valued functions of an integer variable. We say **f(n) is o(g(n))** if for **any** c>0, there is an n_0 such that $f(n) \le g(n)$ for all $n>n_0$. This is pronounced as "f(n) is little-oh of g(n)".

Definition: Let f(n) and g(n) be real valued functions of an integer variable. We say **f(n) is ω (g(n)** if g(n) is o(f(n)). This is pronounced as "f(n) is little-omega of g(n)".

■ **Example** log(n) is o(n) and x^2 is ω (nlog(n)).

What is "fast" or "efficient"?

If the asymptotic time complexity is bad, say n^5, or horrendous, say 2^n, then for large n, the algorithm will definitely be slow. Indeed for exponential algorithms even modest n's (say n = 50) are hopeless.

Algorithms that are o(n) (i.e., faster than linear, a.k.a. sub-linear), e.g. logarithmic algorithms, are very fast and quite rare. Note that such algorithms do not even inspect most of the input data once. Binary search has this property. When you look a name in the phone book you do not even glance at a majority of the names present.

Linear algorithms (i.e., Θ(n)) are also fast. Indeed, if the time complexity is O(nlog(n)), we are normally quite happy.

Low degree polynomial (e.g., $\Theta(n^2)$, $\Theta(n^3)$, $\Theta(n^4)$) are interesting. They are certainly not fast but speeding up a computer system by a factor of 1000 (feasible today with parallelism) means that a $\Theta(n^3)$ algorithm can solve a problem 10 times larger. Many science/engineering problems fall in this range.

■ **Example** Proove $n^3 - n^2 + 2n\log(n) \in \Omega(n\log(n))$

■ **Answer:** Since $n^3 - n^2 >= 0$ for $n >= 0$ and $2n\log(n)>0$ for $n >1$ we have

$n^3 - n^2 + 2n\log(n) >= 2n\log(n) >0$ for all n >1

ie $n^3 - n^2 + 2n\log(n) \in \Omega(n\log(n))$ with c=2 and n_0=1.

■ **Example** Proove $\sum_{i=1}^{n} i \in O(n^2)$

■ **Answer :** We have $\sum_{i=1}^{n} i = n(n + 1)/2 = n^2/2 + n/2$

Since $0 < n^2/2 + n/2 <= n^2/2+n^2/2 = n^2$ for n > 0 we get

$$0 < \sum_{i=1}^{n} i = n^2/2 + n/2 <= n^2 \quad \text{for all n} > 0$$

ie $\sum_{i=1}^{n} i \in O(n^2)$ with c=1 and n_0=0.

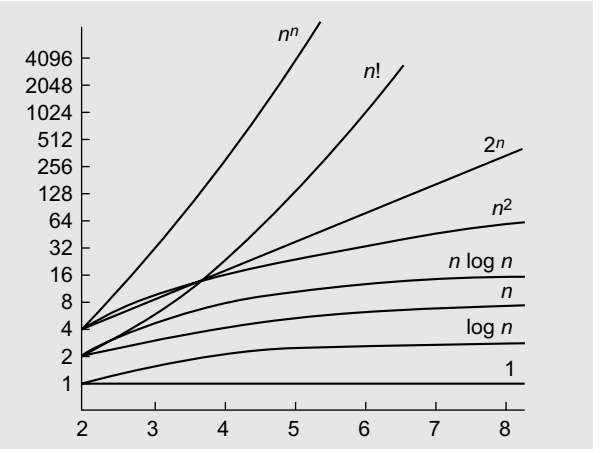
2.4.1.2 The Importance of Asymptotics

It really is true that if algorithm A is o(algorithm B) then for large problems A will take much less time than B.

Definition: If (the number of operations in) algorithm A is o(algorithm B), we call A **asymptotically faster** than B.

■ **Example** The following sequence of functions are ordered by **growth rate**, i.e., each function is little-oh of the subsequent function. log(log(n)), log(n), $(\log(n))^2$, $n^{1/3}$, $n^{1/2}$, n, nlog(n), n^2 (log(n)), n^2, n^3, 2^n.

Function	Name
c	Constant O(1)
log n	Logarithmic
$\log^2 n$	Log-squared
N	Linear
n log n	–
n^2	Quadratic
n^3	Cubic
2^n	Exponential



We do require to know about common program design paradigms.

- Divide-and-conquer
- Iterative
- Recursive
- Recurrence relation based
- Incremental
- Dynamic programming
- Greedy algorithms
- Randomized/probabilistic

We have got exposure to iterative, recurrence, and recursive solutions in the previous chapters. We will confine ourselves to divide-and-conquer based solution in the coming chapters. Other solutions are beyond the scope of this book.

In the following example, we have included a simple example of permutations of a string using both iterative and recursive means. In practice, we may need to study which is better in terms time, space, scalability, and ease of implementation.

Some Points to remember

- log's of different bases grow at the same rate
- 2^n grows much faster than n^2
- $n^2/\ln(n)$ grows just a little slower than n^2, but much faster than n
- n^2 grows faster than n log(n)
- sqrt(n) grows faster than log(n)

■ **Example** We are given a sequence $a_1, \ldots, a_n$ of numbers on input. Give an algorithm that computes the value of

$$F = \sum_{j=1}^{n} \frac{\sum_{i=1}^{j} a_i^a - \sum_{i=1}^{j} i a_i}{j(j+1)}$$

Your algorithm must run in linear time. Present your algorithm in pseudo-code, and briefly explain why its running time is linear.

■ **Answer:**

```
Function F (n : integer)
sum =0, prefix-sum =0
for j  = 1 to n do
prefix-sum = prefix-sum + (a²j − j · aj)
sum =sum + prefix-sum/(j · (j + 1))
end for
return sum
end
```

Since there is only one for loop in the algorithm, it is easy to see the running time is linear.

Some Points about Big O notation

For any **polynomial** $f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_0$, where $a_0, a_1, \ldots, a_n$ are real numbers,

$f(x)$ is $O(x^n)$.

If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$, then $(f_1 + f_2)(x)$ is $O(\max(g_1(x), g_2(x)))$

If $f_1(x)$ is $O(g(x))$ and $f_2(x)$ is $O(g(x))$, then $(f_1 + f_2)(x)$ is $O(g(x))$.

If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$, then $(f_1 f_2)(x)$ is $O(g_1(x)\, g_2(x))$

■ **Example** **If Ω(n lg(lgn)) then f1(n)** must be in **O(n lgn)**? Why or why not?

■ **Answer:** NO. **f1(n)** is in **Ω (n lg(lgn))** only provides a lower bound on **f1(n). n lgn** is in **Ω (n lg(lgn))**. It is possible for **f1(n)** to be larger than **n lgn**; for example, **f1(n) = n2.** In this case, **f1(n)** is not in **O(n lgn)**.

■ **Example** Show that:

i. $(n+1)^5$ is $O(n^5)$

ii. 2^{n+1} is $\Theta(n^5)$

iii. $n + (\log_2 n)^2$ is $O(n\sqrt{n})$

Answer: By rules of limiting: $L = \lim_{n\to\infty} \frac{f(n)}{g(n)}$

Rule 1: if $L = 0$ $f(n) = O(g(n))$

Rule 2: if $L = \infty$ $f(n) = \Omega(g(n))$

Rule 3: if $L =$ any non-zero constant $f(n) = \Theta(g(n))$

i. $(n+1)^5$ is $O(n^5)$

$$L = \lim_{n\to\infty} \frac{(n+1)^5}{n^5}$$

$$= \lim_{n\to\infty} \frac{n^5 + 5n^4 + 10n^3 + 10n^2 + 5n + 1}{n^5}$$

$$= \lim_{n\to\infty} 1 + \frac{5}{n} + \frac{10}{n^2} + \frac{10}{n^3} + \frac{5}{n^4} + \frac{1}{n^5} = 1$$

By Rule 3, $(n+1)^5 = \Theta(n^5)$. So, this also implies $(n+1)^5 = O(n^5)$

ii. 2^{n+1} is $\Theta(n^5)$

$$L = \lim_{n\to\infty} \frac{2^{n+1}}{2^n} = \lim_{n\to\infty} \frac{2 \times 2^n}{2^n} = 2$$

By Rule 3, 2^{n+1} is $\Theta(2^n)$

iii. $n + (\log_2 n)^2$ is $O(n\sqrt{n})$

$$L = \lim_{n\to\infty} \frac{n(\log n)^2}{n\sqrt{n}} = \lim_{n\to\infty} \frac{(\log n)^2}{\sqrt{n}}$$

$$= \lim_{n\to\infty} \frac{2 \times (\log n) \times \frac{1}{n \ln 2}}{\frac{1}{2} n^{-\frac{1}{2}}}$$

$$= \lim_{n\to\infty} \frac{4}{\ln 2} \times \frac{\log n}{\sqrt{n}} = \lim_{n\to\infty} \frac{4}{\ln 2} \times \frac{\frac{1}{n\ln 2}}{\frac{1}{2} n^{-\frac{1}{2}}}$$

$$= \lim_{n\to\infty} \frac{8}{(\ln 2)^2} \times \frac{1}{\sqrt{n}} = 0$$

By Rule 1, $n + (\log_2 n)^2$ is $= O(n\sqrt{n})$

■ **Example** Solve the following recursive relation using the substitution method.

$T(1) = 1$

$T(n) = T\left(\frac{n}{2}\right) + \sqrt{n}$

■ **Answer:** By expanding

$$T(n) = T\left(\frac{n}{2}\right) + \sqrt{n} = T\left(\frac{n}{2}\right) + \sqrt{\frac{n}{2}} + \sqrt{n} = ...$$

$$= T\left(\frac{n}{2^i}\right) + \sqrt{\frac{n}{2^{i+1}}} + ... + \sqrt{n}$$

Let, T(n) = O(√n), meaning T(n) ≤ c√n

Induction base:n = 1, c = 4

T(1) = 1 ≤ c√1 , c > 0

Induction step:

$$T(1) = T\left(\frac{n}{2}\right) + \sqrt{n} \le c\sqrt{\frac{n}{2}} + \sqrt{n} = \sqrt{n}\left(\frac{c}{\sqrt{2}} + 1\right) \le c\sqrt{n}$$

(c/√2 + 1) ≤ c true for c = 4

■ **Example** Assume we have to calculate row sums of an n X n two dimensional array in addition to over all total. Propose code fragment and enumerate how many operations are involved.

■ **Answer:**

```
grand Total = 0;
for (k=0; k<n; k++)  {
rows[k] = 0;
for (j = 0; j <n; j++){
rows[k] = rows[k] + matrix[k][j];
grandTotal = grandTotal + matrix[k][j];
}
}
```

It takes $2n^2$ addition operations

Another solution:

```
grandTotal =0;
for (k=0; k<n; k++){
rows[k] = 0;
for (j = 0; j <n; j++)
```

```
rows[k] = rows[k] + matrix[k][j];
grandTotal = grandTotal + rows[k];
}
```

This one takes $n^2 + n$ operations

■ **Example** What is the complexity of the following code fragment in big-oh notation?

```
Dim iSum,i,j,k As Integer
       For i=1 to n                    \
         For j=1 to n     \              |
           iSum = iSum + 1 | ---  1 x n  |
         Next j           /              |
         For k=1 to 2*n   \              | --- x n
           iSum = iSum + 1 |             |
           iSum = iSum + 1 | ---  3 x 2n |
           iSum = iSum + 1 |             |
         Next k           /              |
       Next i                           /
```

■ **Answer:** If we observe the above code, we find inner j loop runs for n times for each value of i. Thus, the statements inside the j loop will be executed n^2 times. Similarly, k loop contains 3 statements which will be executed 2n times for each value iteration of i. Thus, inner k loop involves $6n^2$ opertaions.

Thus, number of operations: n x (1 × n + 3 × 2n) = $n^2 + 6n^2 = 7n^2$

Since in Big-Oh analysis we ignore leading constants (the 7 in the equation above), this algorithm runs in $O(n^2)$ time.

Complexity: O (n^2)

■ **Example** What is the complexity of the following code fragment in terms of big-oh notation?

```
Dim iSum,i,j,k As Integer
  For i=1 to n                              \
    For j=1 to n                  \          |
      iSum = iSum + 1              |---1 x n |
    Next j                        /          |
    For k=1 to 2*n                \          |---x n
      iSum = iSum + 1     > 1      |         |
    For m=1 to 4*n step 2 \         | --- 2n  |
      iSum = iSum + 1      | 1 x 2n |         |
      iSum = iSum + 1     /         |         |
    Next k                        /          |
  Next i                                     /
```

■ **Answer:** If we observe the above code, we find inner j loop runs for n times for each value of i. Thus, the statements inside the j loop will be executed n^2 times. Similarly, inner k loops first statement which will be executed 2n times for each value iteration of i. Thus, inner k loop involves $2n^2$ opertaions. The innermost m loop runs for 4n times with the step value of 2. Thus, it runs for $2n^2*2n = 4n^3$

Thus, total number of times executed: n (1 x n + 2n (1 + (1 x 2n)))

$= n (n + 2n (1 + 2n))$

$= n (n + 2n + 4n^2)$

$= n^2 + 2n^2 + 4n^3$

$= 3n^2 + 4n^3$

Since, in Big-Oh analysis we ignore leading constants (the 3 and 4 in the equation above) and lower order terms (the n^2 in the equation above), this algorithm runs in $O(n^3)$ time.

Complexity: $O(n^3)$

■ **Example** By using repeated substitution to find the time complexity of the function recurse. Assume only non-negative even values of n are passed in.

```
void recurse(int n) {
    int i, total = 0;
(1) if (n == 0) return 1;
(2) for (i = 0; i < 4; i++) {
(3) total += recurse(n-2);}
(4) return total;
 }
```

■ **Answer:**

- When n = 0 (the base case), this function only executes line 1, doing O(1) work. We will note this constant amount of work as *a*.
- When n >= 2 (the recursive case), this function does O(1) work in executing lines 1, 2, and 4, which we will denote by b, and also makes four recursive calls with parameter values n-2 in the body loop (line 3).

We use these two points to perform the recurrence relation analysis as follows:

From the base case, we have: $T_0 = a$

From the recursive case, we have: $T_n = 4T_{n-2} + b$

We apply repeated substitution:

$T_n = 4T_{n-2} + b$

$T_n = 4(4T_{n-4} + b) + b = 4^2 T_{n-4} + 4b + b$

$T_n = 4^2 (4T_{n-6} + b) + 4b + b$

$= 4^3 T_{n-6} + 4^2 b + 4b + b$

$T_n = 4^i T_{n-2i} + 4^{i-1} b + 4^{i-2} b + \ldots + 4^0 b$

Note that $4^k = 2^{2k}$, yielding:

$T_n = 2^{2i} T_{n-2i} + 2^{2(i-1)} b + 2^{2(i-2)} b + \ldots + 2^0 b$

Set i = n/2, yielding:

$T_n = 2^n T_0 + 2^{n-2} b + 2^{n-4} b + \ldots + 2^0 b$

$= 2^n a + (2^{n-2} + 2^{n-4} + \ldots + 2^0) b$

So, we need to compute the closed form for $(2^{n-2} + 2^{n-4} + \ldots + 2^0) = (2^n - 1)/3$.

Here is a brief digression on how we compute this closed form formula:

Let $x = (2^{n-2} + 2^{n-4} + \ldots + 2^0)$

So, $2^2 x = 4x = 2^2 (2^{n-2} + 2^{n-4} + \ldots + 2^0)$

$= (2^n + 2^{n-2} + \ldots + 2^2)$

$2^2 x - x = 3x = (2^n + 2^{n-2} + \ldots + 2^2) -$

$(2^{n-2} + 2^{n-4} + \ldots + 2^0)$

$3x = 2^n - 2^0 = 2^n - 1$

$x = (2^n - 1)/3 = (2^{n-2} + 2^{n-4} + \ldots + 2^0)$

Substituting this closed form for the sum back in to the equation for T_n yields:

$T_n = 2^n a + (2^{n-2} + 2^{n-4} + \ldots + 2^0) b$

$T_n = 2^n a + ((2^n - 1)/3) b$

Thus, complexity in Big-oh notation is $O(2^n)$.

■ **Example** Solve the following.

$T(a) = \Theta(1)$

$T(n) = T(n-a) + T(a) + n$

Find T(n) using the iteration method.

■ **Answer:**

$T(n) = T(n-a) + T(a) + n$

$= [T(n-2a) + T(a) + (n-a)] + T(a) + n$

$= [T(n-3a) + T(a) + (n-2a)] + 2T(a) + 2n - a$

$= T(n-3a) + 3T(a) + 3n - 2a - a$

$= [T(n-4a) + T(a) + (n - 3a)] + 3T(a) + 3n - 2a - a$

$= T(n-4a) + 4T(a) + 4n - 3a - 2a - a$

...

$= T(n-ia) + iT(a) + in - \sum_{k=1}^{i-1} ka$

$= T(n-ia) + iT(a) + in - ai(i-1)/2$

After ((n/a)-1) steps, the iterations will stop, because we have reached T(a).
Assign i = (n/a)-1:

$T(n) = T(n-((n/a)-1)a) + ((n/a)-1)T(a) + ((n/a)-1)n - a((n/a)-1)((n/a)-2)/2$

$= T(a) + (n/a)T(a) - T(a) + (n^2/a) - n - (a/2)*((n^2/a^2) - 3(n/a) + 2) =$

$= (n/a)T(a) + (n^2/a) - n - (n^2/2a) + (3n/2) - a =$

$= (n/a)T(a) + (n^2/2a) + (n/2) - a =$

$= \Theta(n)\Theta(1) + \Theta(n^2) + \Theta(n) = \Theta(n^2)$

$\boxed{T(n) = \Theta(n^2)}$

■ **Example** Compare the following iterative and recursive methods for calculating Fibnocci number.

```
IterFib ( n) {
   f[0] = 0;
   f[1] = 1;
   for ( i=2 ; i ≤ n ; i++)
      f[i] = f[i-1] + f[i-2];
```

```
}
recFib(n) {
   if (n ≤ 1)
      return n
   else
      return recFib(n-1) + recFib(n-2)
}
```

■ **Answer** Time complexity of iterative version is O(n). Whereas the time complexity of recursive version can be represented a:

$$T(n) = T(n-1)+T(n-2)+1$$

By solving this, we can find its complexity by expanding as

$$T(n) \le 2n^{-1} + 2n^{-1} - 1 = O(2n)$$

■ **Example** Solve the following recursive relation.

$$T(1) = 1$$
$$T(n) = T(n-1) + 1/n$$

■ **Answer:** By expanding the above equation

$$T(n) = \frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \cdots 1 = \sum_{i=1}^{n} \frac{1}{i}$$

$$1 + \sum_{i=2}^{n} \frac{1}{i} \le 1 + \int_{1}^{n} \frac{1}{x} dx$$

$$= 1 + \ln n \Big|_1^n = 1 + \ln n$$

Thus, $T(n) = O(\log n)$

■ **Example** Show that the solution of

$T(n) = T(\lceil n/2 \rceil) + 1$ is $O(\lg n)$.

■ **Answer:** Base case: $n = 2$

$$T(\lceil n/2 \rceil) + 1 \le c \lg 2$$

$$T(2/2) + 1 \le c \lg 2$$

Let $c = 2$

$$1 + 1 \le 2 \lg 2$$

Inductive Hypothesis:

Assume: $T(\lceil n/2 \rceil) + 1 \le c \lg (\lceil n/2 \rceil)$

Inductive step:

$$T(n) \le c (\lg n - \lg 2) + 1 = c \lg n - c \lg 2 + 1$$

$$\le c \lg n$$

■ **Example** Show that the solution of $T(n) = 2T(\lfloor n/2 \rfloor) + n$ is $\Omega(n \lg n)$. Conclude that the solution is $\Theta(n \lg n)$.

■ **Answer:** By induction:

Base case: $n = 2$

$2T(\lfloor 2/2 \rfloor) + 2 \geq c2 \lg 2$

$2 + 2 \geq c2 \lg 2$

Let $c = 1$

$2 + 2 \geq 2$

Inductive Hypothesis:

Assume: $T(\lfloor n/2 \rfloor) \geq c \lfloor n/2 \rfloor \lg (\lfloor n/2 \rfloor)$

Inductive step:

$$\begin{aligned} T(n) &\geq c((n/2) - 1)(\lg n - \lg 2 - 1) \\ &= c(n/2 \lg n - n/2 \lg 2 - n/2 - \lg n + \lg 2 + 1) \\ &= c(n/2 \lg n - n/2 - n/2 + 2) \\ &= c(n/2 \lg n - n + 2) \\ &\geq c(n/2 \lg n) \\ &= c\, n \lg n \end{aligned}$$

Since $T(n) = O(n \lg n)$

and $T(n) = \Omega(n \lg n)$

then $T(n) = \Theta(n \lg n)$

■ **Example** Show that the solution to $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$ is $O(\text{n lg } n)$.

■ **Answer:** Assume

$$\begin{aligned} \text{T(n)} &\leq \text{cn lg n} - \text{b}, \qquad \text{for b} \geq 0 \\ \text{T(n)} &\leq (c\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor + 17 - b) + (c \lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor + 17 - b) \\ &\leq cn\, 2 \lg (n/2) + 34 - 2b \\ &\leq cn \lg n + 34 - 2b \\ &\leq cn \lg n - b \\ &\leq cn \lg n = O(\text{n lg } n) \end{aligned}$$

■ **Example** Determine a good asymptotic upper bound on the recurrence

T(n) $= 3T(\lfloor n/2 \rfloor) + n$ by iteration.

■ **Answer:**

$$\begin{aligned} \text{T(n)} &= 3T(\lfloor n/2 \rfloor) + n \\ &= 3(\lfloor n/2 \rfloor + 3T(\lfloor n/4 \rfloor)) + n \\ &= 3(\lfloor n/2 \rfloor + 3T(\lfloor n/4 \rfloor + 3T(\lfloor n/8 \rfloor))) + n \\ &\leq 3n/2 + 9n/4 + 27n/8 + \ldots + 3^{\lg 3}\, \Theta(1) + n \\ &\leq \sum_{i=0}^{\infty} (3/2)^{\text{i}} + \Theta(n^{\lg 3}) \\ &= \text{O}(n) + \text{O}(n^{\lg 3}) \\ &= \text{O}(\max(\text{O}(n), \text{O}(n^{\lg 3})) \\ &= \text{O}(n^{\lg 3}) \end{aligned}$$

2.4.1.3 Masters Theorem

Theorem 1 (Master Theorem): Let *f* be an increasing function that satisfies the recurrence relation:

$f(n) = a\, f(n/b) + cn^d$

whenever $n = b^k$, where k is a positive integer, a >= 1, b is an integer greater than 1, c is a positive real number, and d is a non-negative real number. Then:

$$f(n) = \begin{cases} O(n^d) & \text{if } a < b^d \\ O(n^d \log n) & \text{if } a = b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Here are some examples to consider:

- $a < b^d$: $f(n) = 2\, f(n/2) + xn^2$. In this example, $2 < 2^2$, and thus $f(n) = O(n^2)$. The xn^2 term is growing faster than the $2\, f(n/2)$ and thus dominates.
- $a = b^d$: $f(n) = 2\, f(n/2) + xn^1$. In this example, $2 = 2$, and thus $f(n) = O(n \log n)$. The two terms grow together.
- $a > b^d$: $f(n) = 8\, f(n/2) + xn^2$. In this example, $8 < 2^2$, and thus $f(n) = O(n^{\log 8}) = O(n^3)$. The $8\, f(n/2)$ term is growing faster than the xn^2 and thus dominates.

■ **Example** Solve the following recurrence relations using Masters theorem.

a. **$T(n) = 7T(n/2) + n^2$**
   $a = 7, b = 2, d=2$. As a is greater than b^d,
   Therefore, $T(n)=O(n^{\log_2 7})$

b. **$T(n) = 4T(n/2) + n^2$**
   $a = 4, b = 2, d=2$. This is case 2 of Masters theorem as a is b^d.
   Therefore, $T(n) =O(n^{\log_2 4}\log n)=O(n^2\log n)$

c. **$T(n) = 2T(n/3) + n^3$**
   $a = 2, b = 3, d=3$
   As a is less than b^d, this is case 1 type. Thus, $T(n) = O(n^3)$

■ **Example** Solve the following

$T(n) = 3T(n/2) + n\log n$

Use the Master-Method to find T(n).

■ **Answer:**

$a = 3, b = 2, f(n) = n\log n$

We assume it is the 1-st case of a Master Theorem, so we need to show that:

$n\log n = O(n^{\log_2 3 - \varepsilon})$

$x = \log_2 3 = 1.585$

$n\log n \le cn^{x - \varepsilon}$

$\log n \le cn^{x - \varepsilon - 1}$

We may operate log on both sides (log is a monotonic increasing function and thus we are allowed to do this):
$\log(\log n) \le (x - \varepsilon - 1) \ln n + \log c = (0.585 - \varepsilon) \log n + \log c$
Next, we need to find values of c, ε, n_0, such that:

$\log(\log n) \le (0.585 - \varepsilon) \log n + \log c$

Let us choose c = 1: $\log(\log n) \le (0.585 - \varepsilon) \log n$

$\log(f(n))$ is smaller then $0.5f(n)$ ($f(n)$ is a monotonous increasing function), for $f(n) > 4$.

Thus, we may choose $\varepsilon = 0.585$

f(n) = logn=4. Hence we set $n_0 = 16$.

$$\boxed{T(n) = \Theta(n^{\log_2 3})}$$

Masters Method Revisited for asymptotic tight bounds
Bound a recurrence of the form:

$$T(n) = aT(n/b) + f(n) \quad a \geq 1, \quad b > 1$$

1. if $f(n) = O(n^{\log}ba^{-\varepsilon})$, $\varepsilon > 0$ then $T(n) = \Theta(n^{\log}ba)$
2. if $f(n) = \Theta(n^{\log}ba)$ then $T(n) = \Theta(n^{\log}ba \log n)$

if $\mathbf{f(n) = \Omega(n^{\log_b a + \varepsilon})}$, $\varepsilon > 0$ and af(n/b) ≤ cf(n) for c < 1 then **T(n) = Θ(f(n))**

■ **Example** Find out upper bound for the following recurrence:

$$T(n) = \begin{cases} 5T\left(\frac{n}{7}\right) + \log_7 n & n > 1 \\ 1 & n = 1 \end{cases}$$

That is, find the smallest function $f(n)$ that we can, such that $T(n) \in O(f(n))$. You may assume that n is a power of 7.

■ **Answer:** The Master Theorem is applicable because the ratio test yields: $\frac{f(n)}{n^{\log_b a}} = \frac{\log_7 n}{n^{\log_7 5}}$ in which there is polynomial dominance in the denominator. That is, $\log_7 n \in O(n^{(\log_7 5)-\varepsilon})$ for some $0 < \varepsilon$. Case 1 applies, so the solution is $T(n) \in \Theta(n^{\log_7 5})$.

We can also use a recursion tree to obtain a looser upper bound. At each level of the tree each problem gives rise to 5 subproblems. The height of the recursion tree is: $\log_7 n$. The summation corresponding to the work is given by:

$$\sum_{i=0}^{\log_7 n} \left(5^i \log_7 \left(\frac{n}{7^i}\right)\right) = \sum_{i=0}^{\log_7 n} \left(5^i \left(\log_7 n - \log_7 7^i\right)\right)$$

$$\sum_{i=0}^{\log_7 n} \left(5^i \left(\log_7 n - i\right)\right) \leq \sum_{i=0}^{\log_7 n} \left(5^i \left(\log_7 n\right)\right) = \log_7 n \sum_{i=0}^{\log_7 n} \left(5^i\right)$$

The closed form solution to the summation is:

$$\sum_{i=0}^{\log_7 n} \left(5^i\right) = \frac{5^{1+\log_7 n} - 1}{4} = \frac{5(5^{\log_7 n}) - 1}{4}$$

$$\frac{5(5^{\log_5 n}) - 1}{4} = \frac{5(n) - 1}{4} \in \Theta(n)$$

The work is therefore ∈ $O(n \lg n)$. (Note that $n^{\log_7 5} < n \Rightarrow n^{\log_7 5} < n \lg n$, so this is a looser upper bound than what we derived using the Master Theorem.)

■ **Example** Use the master method to give tight asymptotic bounds for the following recurrences.

a. $T(n) = 4T(n/2) + n$.

This of CASE 1 type

$a = 4, \quad b = 2, \quad f(n) = n, \quad n^{\log}b^{a} = n^{\lg 4} = n^2, \in = 1$

$\therefore \quad f(n) = O(n^{\log}b^{a-\in}) = O(n^{\lg 4-1}) = O(n)$

$\therefore \quad T(n) = 4T(n/2) + n = \Theta(n^2)$

b. $T(n) = 4T(n/2) + n^2$

This of CASE 2 type

$a = 4, \quad b = 2, \quad f(n) = n^2, \quad n^{\log}b^{a} = n^{\lg 4} = n^2$

$\therefore \quad f(n) = O(n^{\log}b^{a-\in}) = O(n^{\lg 4}) = O(n^2)$

$\therefore \quad \Theta(n^{\log}b^{a} \lg n) = \Theta(n^2 \lg n)$

$\therefore \quad T(n) = 4T(n/2) + n^2 = \Theta(n^2 \lg n)$

c. $T(n) = 4T(n/3) + n^3$

This of CASE 3 type:

$a = 4, \quad b = 2, \quad f(n) = n^3, \quad n^{\log_b a} = n^{\lg 4} = n^2, \ \in = 1$

$\therefore \quad f(n) = \Omega(n^{\log_b a + \in}) = \Omega(n^{\lg 4+1}) = \Omega(n^3)$

and $\quad 4(n/2)^3 \le cn^3$, for $c < 1$

$\therefore \quad T(n) = 4T(n/3) + n^3 = \Theta(n^3)$

■ **Example** The running time of an algorithm A is described by the recurrence $T(n) = 7T(n/2) + n^2$. A competing algorithm A' has a running time of $T'(n) = aT'(n/4) + n^2$. What is the largest integer value for a such that A' is asymptotically faster than A?

$7T(n/2) + n^2$	$aT(n/4) + n^2$
$a = 7, \quad b = 2$	$a = a, \quad b = 4$
$f(n) = n^2$	$f(n) = n^2$
$n^{\log_b a} = \quad n^{\lg 7}$	$n^{\log_b a} = n^{\log_4 a}$

$\therefore \quad \log_4 a = \lg 7$

$\therefore$ The largest integer value for a is $4^{\lg 7}$

■ **Example** Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \le 2$. Make your bounds as tight as possible, and justify your answers.

a. $\boldsymbol{T(n) = 2T(n/2) + n^3}$

This is CASE 3 of Master Theorem

$a = 2, \quad b = 2, \quad f(n) = n^3, \quad n^{\log_b a} = n^{\lg 2} = n$

$n^3 = \Omega(n^{1+1}) = \Omega(n^2)$

$2(n/2)^3 \le cn^3, \quad c < 1 \ \Rightarrow \ c = ¼$

$T(n) = \Theta(n^3)$

b. $\boldsymbol{T(n) = T(9n/10) + n}$

This CASE 3 of Master Theorem

$a = 1, \quad b = 10/9, \quad f(n) = n, \quad n^{\log_b a} = n^{\log_{10/9} 1} = n^0 = 1$

$n = \Omega(n^{0+1}) = \Omega(n)$

$9n/10 \le cn, \ c < 1 \ \Rightarrow \ c = 9/10$

$T(n) = \Theta(n)$

c. $\boldsymbol{T(n) = 16T(n/4) + n^2}$

CASE 2 of Master Theorem

$a = 16, \quad b = 4, \quad f(n) = n^2, \quad n^{\log_b a} = n^{\lg_4 16} = n^2$

$T(n) = \Theta(n^2 \lg n)$

d. $\boldsymbol{T(n) = 7T(n/3) + n^2}$

This CASE 3 of Master Theorem

$a = 7, \quad b = 3, \quad f(n) = n^2, \quad n^{\log_b a} = n^{\lg_3 7}$

$n^2 = \Omega(n^{\log_3 7 + c})$

$7(n/3)^2 \le cn^2, \quad c < 1 \ \Rightarrow \ c = 7/9$

$T(n) = \Theta(n^3)$

e. $\boldsymbol{T(n) = 7T(n/2) + n^2}$

This CASE 1 of Master Theorem

$a = 7, \quad b = 2, \quad f(n) = n^2, \quad n^{\log_b a} = n^{\lg 7}$

$n^2 = O(n^{\lg 7 - c})$

$T(n) = \Theta(n^{\lg 7})$

■ **Example** Whose upperbound is more out of the following two recursive relations:

$$T(n) = T(n/2) + \Theta(1)$$

$$T(n) = T(n/2) + \Theta(n/2) = T(n/2) + \Theta(n)$$

■ **Answer:**

First one: This of CASE 2 of Master's Theorem

$$a = 1, \quad b = 2, \quad n^{\log_b a} = n^{\lg 1} = n^0 = 1,$$

$$f(n) = \Theta(1)$$

$$T(n) = \Theta(f(n) \lg n) = \Theta(\lg n)$$

Upper bound = $O(\lg n)$

Second one: This is CASE 3 of Master's Theorem

$$a = 1, \quad b = 2, \quad n^{\log_b a} = n^{\lg 1} = n^0 = 1,$$

$$f(n) = \Theta(n/2)$$

$$af(n/b) \le cf(n) \Rightarrow c(n/2)/2 = cn/4 \le cn/2$$

$$T(n) = \Theta(n/2) = \Theta(n)$$

Upper bound = $O(n)$

Thus, second one upper bound is more.

1. $T(n) = 3T(n/2) + n^2 \Rightarrow T(n) = \Theta(n^2)$ (case 3)
2. $T(n) = 4T(n/2) + n^2 \Rightarrow T(n) = \Theta(n^2 \log n)$ (case 2)
3. $T(n) = T(n/2) + 2n \Rightarrow \Theta(2^n)$ (case 3)
4. $T(n) = 2^n T(n/2) + 2n \Rightarrow$ Does not apply (a is not constant)
5. $T(n) = 16T(n/4) + n \Rightarrow T(n) = \Theta(n^2)$ (case 1)
6. $T(n) = 2T(n/2) + n \log n \Rightarrow T(n) = n \log^2 n$ (case 2)
7. $T(n) = 2T(n/2) + n \log n \Rightarrow$ Does not apply (non-polynomial difference between $f(n)$ and $n^{\log}$, a)
8. $T(n) = 2T(n/4) + n^{0.51} \Rightarrow T(n) = \Theta(n^{0.51})$ (case 3)
9. $T(n) = 0.5T(n/2) + 1/n \Rightarrow$ Does not apply ($a < 1$)
10. $T(n) = 16T(n/4) + n! \Rightarrow T(n) = \Theta(n!)$ (case 3)
11. $T(n) = \sqrt{2T}(n/2) + \log n \Rightarrow T(n) = \Theta(\sqrt{n})$ (case 1)
12. $T(n) = 3T(n/2) + n \Rightarrow T(n) = \Theta(n^{\lg 3})$ (case 1)
13. $T(n) = 3T(n/3) + \sqrt{n} \Rightarrow T(n) = \Theta(n)$ (case 1)
14. $T(n) = 4T(n/2) + cn \Rightarrow T(n) = \Theta(n^2)$ (case 1)
15. $T(n) = 3T(n/4) + n \log n \Rightarrow T(n) = \Theta(n \log n)$ (case 3)
16. $T(n) = 3T(n/3) + n/2 \Rightarrow T(n) = \Theta(n \log n)$ (case 2)
17. $T(n) = 6T(n/3) + n^2 \log n \Rightarrow T(n) = \Theta(n^2 \log n)$ (case 3)
18. $T(n) = 4T(n/2) + n/\log n \Rightarrow T(n) = \Theta(n^2)$ (case 1)
19. $T(n) = 64T(n/8) + n^2 \log n \Rightarrow$ Does not apply ($f(n)$ is not positive)
20. $T(n) = 7T(n/3) + n^2 \Rightarrow T(n) = \Theta(n^2)$ (case 3)
21. $T(n) = 4T(n/2) + \log n \Rightarrow T(n) = \Theta(n^2)$ (case 1)
22. $T(n) = 4(n/2) + n(2 - \cos n) \Rightarrow$ Does not apply. We are in Case 3, but the regularity condition is violated.(Consider $n = 2\pi k$, where k is odd and arbitrarity large. For any such choice of n. you can show that $c \ge 3/2$, there by violating the regularity condition.)

■ **Example** Solve the following recurrence relationship and represent the time complexity in Big-oh notation.

$T(n) = T(\sqrt{n})+1$

$T(1) = 1$

■ **Answer:**

Assuming n is integer power of 2 like 2^m for some integer value of m. Recurrence relation becomes:

$T(2^m) = T(2^{m/2}) + 1$

The same thing can be represented as:

$S(m) = T(m/2)+1$

Where $m = \log_2 n$

By using Masters theorem, that is comparing with

$T(n) = aT(n/b)+f(n)$, we can write

Using the master theorem, $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$

and $f(n) = 1$. Since $1 = \Theta(1)$,

case 2 applies and $S(m) = \Theta(\lg m)$. Therefore,

$T(n) = \Theta(\lg \lg n)$.

■ **Example** Solve the recurrence

$$T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \lg n,$$

Take m = lg *n*, That is, $n = 2^m$. We can rewrite the equation as become

$$T(2m) = 2T(2m^{/2}) + m,$$

Now take S(m) = T (2*m*) to produce the new recurrence

S(m) = 2S(m/2) + m,

Which is very much like the previous recurrence.

Indeed, this new recurrence has the same solution.

S(m) = O(m lg m).

Changing back from S(m) to T(n), we obtain T(n) = T(2m) = S(m) = O(m lg m) = O(lg n lg lg n).

■ **Example** Solve the following recurrence

Consider

Here, we $T(n) = 9T(n/3) + n$. have a = 9, b = 3, f(n) = n, and thus we have that $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$. Since $f(n) = O(n^{\log_3 9-\epsilon})$ where $\epsilon = 1$, we can apply case 1 of the master theorem and conclude that the solution is $T(n) = \Theta(n^2)$.

■ **Example** Solve the following recurrence

$$T(n) = T(2n/3) + 1,$$

Here, a = 1, b = 3/2, f(n) = 1, and $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$ case 2 applies, since, $f(n) = \Theta\left(n^{\log_b a}\right) = \Theta(1)$ and thus the solution to the recurrence is $T(n) = \Theta(\lg n)$.

■ **Example** Arrive at asymptotic bound for the following recurrence relation.

$$T(n) = 3T(n/4) + n \lg n,$$

We have a = 3, b = 4, $f(n) = n \lg n$, and $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$ Since $f(n) = \Omega\left(n^{\log_4 3+\epsilon}\right)$, where $\epsilon \approx 0.2$, case 3 applies if we can show that the regularity condition holds for f(n). For sufficiently large n, af(n/b) = 3(n/4) lg (n/4) ≤ (3/4) n lg n = cf(n) for c = 3/4. Consequently, by case 3, the solution to the recurrence is $T(n) = \Theta(n \lg n)$.

■ **Example** Arrive at asymptotic bound for the following recurrence relation.

$$T(n) = 2T(n/2) + n \lg n,$$

The master method does not apply to the recurrence even though it has the proper form: a = 2, b = 2, f(n) = n lg n, and $n^{\log_b a} = n$. It seems that case 3 should apply, since f(n) = n lg n is asymptotically larger than $n^{\log_b a} = n$.

The problem is that it is not polynomially larger.

The ratio f(n)/ $n^{\log_b a} = (n \lg n)/n = \lg n$ is asymptotically less than $n^{\in}$ for any positive constant $\in$. Consequently, the recurrence falls into the gap between case 2 and case 3.

■ **Example** Represent the complexity of the following code fragment in big-oh notation.

```
int main(){
    int i, doyouwant, A[100], n;
    i = 0;
    printf("Enter 0 for no search 1 for search\n");
    scanf("%d", &doyouwant);
    while ((A[i] != doyouwant) && (i < n)) i++;
    if (i > n) printf("not found"\n);
    else printf("found\n");
return 0;
}
```

Analysis: This code has a comparison: `(A[i] != doyouwant)`. We count comparisons to determine the running time. This means the running time is essentially the number of times the while loop is executed. This depends on the input and the value of doyouwant which we do not know. So we must consider the worst possible case. In the worst case the value of doyouwant is not found and the while loop is executed n times. Therefore the worst-case running time is O(n).

■ **Example** Solve the following recurrence relationship using Master theorem.

$T(n) = 7T(n/4) + O(n^2)$

$a = 7, b = 4, f(n) = n^2$

$n^{\log_4^7} = n^{<1}$, Compare it with $f(n) = n^2$, as f(n) is greater so case III of Mastertheorem will be applied.

$af(n/b) < c\, f(n)$ where < 1

$7 (n/4^2) \le cn^2 \rightarrow (7/16)n^2 \le cn^2 \rightarrow$ so it will be true for any $7/16 < c < 1$.

So the solution is $O(n^2)$

■ **Example** Solve the following recurrence relation:

$T(n) = 3T(n/2) + \theta(n)$.

The solution to this recurrence is $T(n) = \theta(n\text{^}(\log_2 3))$, which is approximately $T(n) = \theta(n^{1.585})$

■ **Example** Derive tight asymptotic bounds for T(n) in each of the following recurrences. Assume that T(n) is a constant for n = O(1).

1. $T(n) = T(2n/3) + 6n$ Here,

   $a = 1,\ b = \frac{3}{2},\ f(n) = 6n$

   $n^{\log_b a} = n^0 = 1$

   $\Rightarrow f(n) = \Omega(n^{\log_b a}) \Rightarrow case\ 3:$

   $T(n) = \Theta(f(n)) = \Theta(n)$

This is 3rd case, regularity of n was shown in class and therefore $T(n) = \Theta(n)$

2. $T(n) = T(5n/7) + 9$

   Here,

   $a = 1,\ b = \frac{7}{5},\ f(n) = 9$

   $n^{\log_{7/5} 1} = n^0 = 1$

$\Rightarrow f(n) = \Theta(n^{\log_b a}) \Rightarrow case\,2:$

$T(n) = \Theta(n^{\log_b a} \log n) = \Theta(\log n)$

This is case 2 of Master theorem. Each level requires constant time and therefore the total running time is log (n) .

3. $T(n) = 2T(\sqrt{n}) + n^2$

Here, we solve the equation by substituting m = log n as it is not in the standard form.

$T(2m) = 2T(3^{m/2}) + 2^{2m}$

let $S(m) = T(2^m)$

$S(m) = 2S(m/2) + 2^{2m}$

$m^{\log_2 2} = m = O(2^{2m})$ $(case\,3)$

$\Rightarrow S(m) = \Theta(2^{2m})$

$\Rightarrow T(n) = \Theta(n^2)$

to complete the solution you need to show the regularity of $f(n) = n^2$

4. $T(n) = 2T(n/2) + (\dfrac{n}{2\log n})$

If we observe, the function f(n) is not polynomial smaller than $n^{\log_b a}$ so case 1 fails. Thus, we will build a recursion tree. The depth of the tree is $\log_2 n$.

The amount paid at each level can be summed using the equation for a harmonic series.

$$\frac{n}{2\log n} + \frac{n}{2\log(n/2)} + \frac{n}{2\log(n/4)} + \ldots \frac{n}{2\log(n/2^{\log(n)})} =$$

$$= \frac{n}{2}\left(\frac{1}{\log n} + \frac{1}{(\log n)-1} + \frac{1}{(\log n)-2} + \ldots \frac{1}{2} + 1\right) =$$

$$= \frac{n}{2}\sum_{i=1}^{\log n}\frac{1}{i} = \frac{n}{2}\ln\log n + O(1) = \Theta(n\log\log n)$$

5. $T(n) = T(n/2) + 2T(n/4)$

This recurrence relation is the same as $T(n)= 2T(n/2)$ since if you iterate a single iteration it becomes $T(n)= T(n/2) + T(n/2) = T(n/2) + 2T(n/4)$ so the solution is the solution for $T(n) = 2T(n/2)$, which by case 1 of master theorem is $T(n) = \Theta(n)$.

6. $T(n) = 2T(n/2) + \log(n!)$

We can not use Master theorem, this is the same as $T(n) = 2T(n/2) + \Theta(n \log n)$

So the solution is $\Theta(n \log^2 n)$

■ **Example** Find the time complexity of the following functions:

(a) T(N) = 2 • T(N/4) + 7 • N – 15

(b) T(N) = 9 • T(N/3) + 3 • N^2 + 5 • N + 16

(c) T(N) = 8 • T(N/2) + 15

■ **Answers:**

(a) T(N) = 2 • T(N/4) + 7 • N – 15

Here a = 2, b = 4, F(N) ∈ $\Theta(N^d)$ for d = 1

$b^d = 4^1 = 4$, so a < b^d and **T(N)** ∈ **Θ(N1)**

(b) T(N) = T(N) = 9 • T(N/3) + 3 • N^2 + 5 • N + 16

Here a = 9, b = 3, F(N) ∈ $\Theta(N^d)$ for d = 2

$b^d = 3^2 = 9$, so a = b^d. **T(N) ∈ Θ(N2 • log(N))**

(c) T(N) = 8 • T(N/2) + 15, which can be written as

T(N) = 8 • T(N/2) + 15 • N^0

Here a = 8, b = 2, $F(N) \in \Theta(N^d)$ for d = 0 (it is a constant)

$b^d = 2^0 = 1$, so $a > b^d$. Let $P = \text{Log}_b(a) = \text{Log}_2(8) = 3$ and $\mathbf{T(N) \in \Theta(N3)}$

■ **Example** Solve the following recurrence

$T(N) = 8 \bullet T(N/2) + N \bullet \log(N)$. Assume N > = 10.

■ **Answer:** If we observer the recurrence relation we may find that we can not apply the Master Theorem. We can use multiple applications of that theorem to make a statement on the growth rate of T(N).

From the given fact that for $N \geq 10$, we can find $N \leq N \bullet \log(N) \leq N^2$.

Now, consider $T(N) = 8 \bullet T(N/2) + N$. Here a = 8, b = 2, and d = 1.

With $b^d = 2^1 = 2$, we have $a > b^d$, so $T(N) \in O(N^P)$, $P = \text{Log}_b(a) = \text{Log}_2(8) = 3$.

For this case, we conclude that $T(N) \in O(N^3)$.

Now, consider $T(N) = 8 \bullet T(N/2) + N^2$. Here a = 8, b = 2, and d = 2.

With $b^d = 2^2 = 4$, we have $a > b^d$, so $T(N) \in O(N^P)$, $P = \text{Log}_b(a) = \text{Log}_2(8) = 3$.

For this case also, we again conclude that $\mathbf{T(N) \in O(N3).}$

■ **Example** Consider the following algorithm. Determine its time complexity.

```
Function xyz ( A[1 .. N], N )
  If (N > 1) Then
   For J = 1 to N Do A[J]= (A[J] + [N - J]) / 2
      N2 = N / 2            // Integer division
      xyz ( A[1 .. N], N2 )
      xyz ( A[(N2 + 1) .. N], (N - N2) )
      For J = 1 to N2 Do A[J] = A[N - J] - A[J]
  End If
End Algorithm
```

■ **Answer:** We count the number of loops. The top loop is executed N times; the bottom one N/2 times. It does not matter what the loop does, just how many time it does it. Thus, $F(N) = 3 \bullet N/2$. Note that there are two recursive calls, each on an array of size N/2.

Thus, we have $T(N) = 2 \bullet T(N / 2) + 3 \bullet N/2$

a = 2, b = 2, d = 1 so $a = b^d$ and $T(N) \in \Theta(N^d \bullet \log(N))$, here $\mathbf{T(N) \in \Theta(N \bullet log(N))}$

■ **Example** Given the recurrence $T(N) = 4T \bullet (N/2) + N^K$ what is the largest value of exponent K such that $T(N) \in O(N^3)$? Assume that $K \geq 0$.

■ **Answer:** Here we have $T(N) = 4 \bullet T(N / 2) + N^K$. Here $a = b^2$, and $P = \log_2(4) = 2$.

If K < 2, then $a > b^K$, and $T(N) \in O(N^2)$, hence $T(N) \in O(N^3)$.

If K = 2, then $a = b^K$ and $T(N) \in O(N^2 \bullet \log(N))$, hence $T(N) \in O(N^3)$.

If K = 3, then $a < b^K$ and $T(N) \in O(N^3)$.

If K > 3, then $a < b^K$ and $T(N) \in O(N^D)$, for D > 3.

Hence $T(N) \notin O(N^3)$.

Hence, we conclude that this holds for $K \leq 3$.

■ **Example** Arrive at upper bound for the following recurrence.

$$T(n) = 8T\left(\frac{n}{2}\right) + 3n^2$$

We apply the Master Theorem. That is,

$$a = 8; b = 2; f(n) = 3n^2$$

$$3n^2 = f(n) = O(n^{\log_b a - \varepsilon}) = O(n^{\log_2 8 - \varepsilon}) \rightarrow \varepsilon = 0.9 \Rightarrow$$

$$\Rightarrow T(n) = \Theta(n^{\log_2 8}) = \Theta(n^3)$$

■ **Example** Solve the following recurrence.

$$T(n)=T\left(\frac{9n}{10}\right)+n=T\left(\frac{n}{\frac{10}{9}}\right)+n$$

We apply the Master theorem.

$$a=1; b=\frac{10}{9}; f(n)=n$$

$$n=f(n)=O(n^{\log_b a+\varepsilon})=O(n^{\log_{\frac{10}{9}} 1+\varepsilon})=O(n^{\varepsilon})\rightarrow \varepsilon=1.1$$

$$af\left(\frac{n}{b}\right)\le cf(n)$$

$$\frac{9n}{10}\le cn \qquad \frac{9}{10}\le c$$

As c<1, we choose c to be $\frac{95}{100}$. Thus,

$$\Rightarrow T(n)=\Theta(f(n))=\Theta(n)$$

■ **Example** Solve the following recurrence.

$$T(n)=2T\left(\frac{n}{4}\right)+\sqrt{n}$$

By applying Masters theorem,

$$a=2; b=4; f(n)=\sqrt{n}$$

$$\sqrt{n}=f(n)=\Theta(n^{\log_b a})=\Theta(n^{\log_4 2})=\Theta(\sqrt{n})\Rightarrow$$

$$\Rightarrow T(n)=\Theta(n^{\log_b a}\log n)=\Theta(\sqrt{n}\log n)$$

■ **Example** Solve the following recurrence.

$$T(n)=T(n-1)+\frac{1}{n}$$

This is not suitable for applying Masters theorem directly.

$$T(n)=T(n-1)+\frac{1}{n}=T(n-2)+\frac{1}{n-1}+\frac{1}{n}=\ldots=$$

$$=T(n-(k+1))+\frac{1}{n}+\frac{1}{n-1}+\ldots+\frac{1}{n-k}$$

Let us find the depth of the recursion:

$$n-(k+1)=1$$

$$n-k-1=1$$

$$k=n-2$$

Now we substitute and get:

$$T(n)=\Theta\left(T(1)+\frac{1}{n}+\frac{1}{n-1}+\ldots+\frac{1}{2}\right)=\Theta(\log n)$$

■ **Example** Solve the following recurrence and find upper and lower bounds.

$$T(n) \quad 2T\left(\frac{n}{2}\right)+n\log^4 n$$

This relation is not directly suitable for applying Masters theorem. First let us develop it a little:

$$T(n)=2T\left(\frac{n}{2}\right)+n\log^4 n=2(2T\left(\frac{n}{2^2}\right)+\frac{n}{2}\log^4\frac{n}{2})+n\log^4 n=$$

$$=2^2T\left(\frac{n}{2^2}\right)+n\log^4 n+n\log^4\frac{n}{2}=...$$

$$=2^{K+1}T\left(\frac{n}{2^{K+1}}\right)+n\log^4 n+n\log^4\frac{n}{2}+...+n\log^4\frac{n}{2^K}$$

Let us find the depth of the recursion:

$$\frac{n}{2^{K+1}}=1$$

$$n=2K^{+1}$$

$$K_0=\log n$$

Meaning we have log(n) members in the series.
Let us find an upper and lower bound.
Lower bound:
Let us take half of the biggest members and find the smallest among them:

$$\frac{n}{2^{\frac{\log n}{2}}}=n\sqrt{\frac{1}{2^{\log n}}}=n\frac{1}{\sqrt{n}}=\sqrt{n}$$

Now we substitute $\frac{n}{k}$ with $\sqrt{n}$ and we get:

$$T(n)\geq\frac{\log n}{2}n\log^4\sqrt{n}=\frac{\log n}{2}n\left(\frac{1}{2}\log n\right)^4$$

$$=\frac{\log n}{2}n\frac{\log^4 n}{16}=\frac{n\log^5 n}{32}\Rightarrow$$

$$\Rightarrow T(n)=\Omega\left(n\log^5 n\right)$$

Now let us find an upper bound. We substitute $\frac{n}{k}$ with n and get:

$$T(n)\leq n\log^4 n+n\log^4 n+n\log^4 n+....=$$

$$=\log n\cdot n\log^4 n=n\log^5 n\Rightarrow$$

$$\Rightarrow T(n)=O\left(n\log^5 n\right)\Rightarrow$$

$$\Rightarrow T(n)=\Theta\left(n\log^5 n\right)$$

■ **Example** Given the following to recurrence relations for what value of a, upper bound of first one will be subset of upper bound of second relation.

$$T(n)=7T(n/2)+n^2$$

$$T'(n)=aT'(n/4)+n^2$$

First we compute the upper bounds for each of the given recurrence relations:

$$T(n)=7T\left(\frac{n}{2}\right)+n^2$$

By applying Master Theorem:

$$a=7;b=2;f(n)=n^2$$

$$n^2 = f(n) = O(n^{\log_2 7-\varepsilon}) \rightarrow [\varepsilon = 0.7] \Rightarrow$$

$$\Rightarrow T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 7})$$

Now, applying Masters theorem to second recursive equation:

$$T'(n) = aT'\left(\frac{n}{4}\right) + n^2$$

$a = a$; $b = 4$; $f(n) = n^2$

$n^2 = f(n) = O(n^{\log_4 a-\varepsilon})$

if $a > 16$ then $\exists \varepsilon > 0$ such that :

$n^2 = O(n^{\log_4 a-\varepsilon}) \Rightarrow$

$\Rightarrow T(n) = \Theta(n^{\log_4 a})$

So, we have:

$$n^{\log_4 a} > n^{\log_2 7}$$

$$\log_4 a > \log_2 7$$

$$\frac{\log_2 a}{\log_2 4} > \log_2 7$$

$$\log_2 a > 2 \log_2 7$$

$$\log_2 a > \log_2 7^2$$

$$a > 7^2$$

$$a > 49$$

Thus, for a value of 49, first one becomes subset of second one.

■ **Example** Find a tight upper bound on the closed-form solution for the following recurrence:

$$T(n) = 3T(n-1) + n$$

where $T(n)$ is constant for sufficiently small n. That is, find a function $g(n)$ such that $T(n) \in O(g(n))$.

■ **Answer:** The Master Theorem does not apply here. A recursion tree can be used. The tree has $n + 1$ levels if $T(0)=1$. $3i(n-i)$ work is done at the i th level, except for the bottom level, where $3n$ $T(0) = 3n$ work is done The total work is:

$$\sum_{i=0}^{n-1}(3^i(n-i)) + 3^n = \left(\sum_{i=0}^{n-1} 3^i n\right) - \left(\sum_{i=0}^{n-1} 3^i i\right) + 3^n$$

$$= n\sum_{i=0}^{n-1} 3^i - \sum_{i=0}^{n-1} 3^i i + 3^n =$$

$$\frac{n(3^n-1)}{2} - \frac{(n-1)3^{(n+1)} - n3^n + 3}{4} + 3^n$$

$$= \frac{7}{4}3^n - \frac{n}{2} - \frac{3}{4} \in O(3^n)$$

Thus, $T(n) \in O(3n)$

■ **Example** Analyse the following pseudocode and arrive at its complexity in big-oh notation.

*XYZ*1 (*n*)

```
if n ≤ 1
        then return
for i ←1 to 3
     do XYZ2 (n/4)
 return
```

```
XYZ2 (n)
        if n ≤ 1
              then return
          XYZ1 (n/4)
         return
```

■ **Answer:** In the worst case, let n be a power of 4.

$T(n) = \Theta(1) + 3(\Theta(1) + T(n/16) = 3T(n/16) + \Theta(1)$. Case 1 of the Master Theorem applies, yielding $T(n) = \Theta(n^{\log_{16} 3})$. So, a tight upper bound is $T(n) = O(n^{\log_{16} 3})$.

■ **Example** Arrive at the complexity for the following recurrence.

$$T(n) = 2T(n/4) + \sqrt{n} = \Theta(\sqrt{n} \lg n).$$

■ **Answer:** Applying Master theorem, $a = 2$, $b = 4$, $f(n) = \sqrt{n}$, and $n^{\log_b a} = n^{\log_4 2} = \sqrt{n}$. Since $\sqrt{n} = \Theta(n^{\log_4 2})$, case 2 of the master theorem applies, and $T(n) = \Theta(\sqrt{n} \lg n)$.

■ **Example** Arrive at the complexity for the following recurrence.

$$T(n) = 4T(n/2) + n^2\sqrt{n}$$

■ **Answer:** We have $f(n) = n^2\sqrt{n} = n^{5/2}$ and $n^{\log_b a} = n^{\log_2 4} = n\lg 2$. Since $n^{5/2} = \Omega(n^{\lg 2+3/2})$, we look at the regularity condition in case 3 of the master theorem. We have

$a f(n/b) = 4(n/2)^2\sqrt{n}/2 = n^{5/2}/\sqrt{2} \le cn^{5/2}$ for $1/\sqrt{2} \le c < 1$.

Case 3 applies, and thus $T(n) = \Theta(n^2\sqrt{n})$.

■ **Example** Arrive at the complexity for the following recurrence.

$$T(n) = T(n-1) + n$$

■ **Answer:** Using the recursion tree shown here, we get a guess of $T(n) = \Theta(n^2)$.

```
n
n-1
n-2
n-3
.
.
.
2
1
-----
Θ(n²)
```

First, we prove the $T(n) = \Omega(n^2)$ part by induction. The inductive hypothesis is $T(n) \ge cn^2$ for some constant $c > 0$.

$$\begin{aligned} T(n) &= T(n-1) + n \\ &\ge c(n-1)^2 + n \\ &= cn^2 - 2cn + c + n \\ &\ge cn^2 \end{aligned}$$

if $-2cn + n + c \ge 0$ or, equivalently, $n(1 - 2c) + c \ge 0$. This condition holds when $n \ge 0$ and $0 < c \le 1/2$.

For the upper bound, $T(n) = O(n^2)$, we use the inductive hypothesis that $T(n) \le cn^2$ for some constant $c > 0$. By a similar derivation, we get that $T(n) \le cn^2$ if $-2cn + n + c \le 0$ or, equivalently, $n(1 - 2c) + c \le 0$. This condition holds for $c = 1$ and $n \ge 1$.

Thus, $T(n) = \Omega(n^2)$ and $T(n) = O(n^2)$, so we conclude that $T(n) = \Theta(n^2)$.

■ **Example** Solve $T(n) = 3T(n/5) + \lg^2 n$

■ **Answer:** Here, $a = 3$, $b = 5$ and $d = 2$. By using Case 1 of the Master Method, we have $T(n) = \Theta\left(n^{\log_5(3)}\right)$.

■ **Example** Solve the following recurrence relation and arrive asymptotic complexity.

$$T(n) = T(\sqrt{n}) + \Theta(\lg \lg n)$$

Change of variables: let $m = \lg n$. Recurrence becomes $S(m) = S(m/2) + \Theta(\lg m)$. Thus, case 2 of Master's theorem applies, so $T(n) = \Theta((\lg \lg n)^2)$.

■ **Example** Solve the following recurrence relation and arrive asymptotic complexity.

$$T(n) = 10T(n/3) + 17n^{1.2}$$

■ **Answer:** Since $\log_3 9 = 2$, so $\log_3 10 > 2 > 1.2$. Case 1 of Master's theorem applies, $\Theta\left(n^{\log_3 10}\right)$.

■ **Example** Solve the following recurrence relation and arrive asymptotic complexity.

$$T(n) = 7T(n/2) + n^3$$

■ **Answer**: By Case 3 of the Master Method, we have $T(n) = \Theta(n^3)$.

■ **Example:** Solve the following recurrence relation and arrive asymptotic complexity.

$$T(n) = T(n/2 + \sqrt{n}) + \sqrt{6046}$$

■ **Answer:** By induction, $T(n)$ is a monotonically increasing function. Thus, for large enough n, $T(n/2) \le T(n/2+\sqrt{n}) \le T(3n/4)$. At each stage, we incur constant cost $\sqrt{6046}$, but we decrease the problem size to atleast one half and at most three-quarters.

Therefore $T(n) = \Theta(\lg n)$.

■ **Example** Solve

$$T(n) = T(n-2) + \log n$$

■ **Answer:** T (n) = $\Theta(n \log n)$, This is T(n) = $\sum_{i=1}^{n/2} \lg 2i \ge \sum_{i=1}^{n/2} \lg i > (n/4)(\lg n/4) = \Omega(n \lg n)$. For the upper bound note that $T(n) < S(n)$, where $S(n) = S(n-1) + \lg n$, which is clearly O ($n \lg n$).

■ **Example** Solve the following recurrence relation and arrive asymptotic complexity.

$$T(n) = T(n/5) + T(4n/5) + \Theta(n)$$

■ **Answer:** Master's theorem doesn't apply here. Draw recursion tree. At each level, do $\Theta(n)$ work. Number of levels is $\log_{5/4} n = \Theta(\lg n)$ so guess $T(n) = \Theta(n \lg n)$ and use the substitution method to verify guess. In the $f(n) = \Theta(n)$ term, let the constants for $\Omega(n)$ and $O(n)$ be $n0$, c0 and c1, respectively. In other words, let for all $n \ge n_0$, we have $c_0 n \le f(n) \le c_1 n$.

- First, we show T(n) = O(n).

For the base case, we can choose a sufficiently large constant d1 such that T(n) < d1n lg n.
For the inductive step, assume for all k < n, that T(k) < d1n lg n. Then for k = n, we have

$$\begin{aligned} T(n) &\le T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + c_1 n \\ &\le d_1 \frac{n}{5} \lg\left(\frac{n}{5}\right) + d_1 \frac{4n}{5} \lg\left(\frac{4n}{5}\right) + c_1 n \\ &\le d_1 n \lg n - \frac{d_1 n}{5} \lg 5 - \frac{4 d_1 n}{5} \lg\left(\frac{5}{4}\right) + c_1 n \\ &\le d_1 n \lg n - n\left(\left(\frac{\lg 5 + 4 \lg(5/4)}{5}\right) + d_1 - c_1\right) \end{aligned}$$

The residual is negative as long as we pick d1 > 5c1/(lg 5+4 lg(5/4)). Therefore, by induction, $T(n) = O(n \lg n)$.

- To show that $T(n) = \Omega(n)$, we can use almost the exact same math.

For the base case, we choose a sufficiently small constant d0 such that $T(n) > d_0 n \lg n$.
For the inductive step, assume for all k < n, that $T(k) > d_0 n \lg n$. Then, for k = n, we have

$$\begin{aligned} T(n) &\geq T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + c_0 n \\ &\geq d_0 \frac{n}{5} \lg\left(\frac{n}{5}\right) + d_0 \frac{4n}{5} \lg\left(\frac{4n}{5}\right) + c_0 n \\ &\geq d_0 n \lg n + n\left(c_0 - \left(\frac{\lg 5 + 4\lg(5/4)}{5}\right) d_0\right) \end{aligned}$$

The residual is positive as long as d0 < 5c0/(lg 5 + 4 lg(5/4)). Thus, $T(n) = \Omega(n \lg n)$.

■ **Example** Solve the following recurrence relation and arrive asymptotic complexity.

$$T(n) = \sqrt{n} T(\sqrt{n}) + 100n$$

■ **Answer:** Master's theorem does not apply here directly. Pick $S(n) = T(n)/n$. The recurrence becomes $S(n) = S(\sqrt{n}) + 100$. The solution of this recurrece is $S(n) = \Theta(\lg \lg n)$. (You can do this by a recursion tree, or by substituting $m = \lg n$ again.) Therefore, $T(n) = \Theta(n \lg \lg n)$.

■ **Example** Solve the following recurrence relation and arrive asymptotic complexity.

$$T(n) = 3*T(n/3) + n + \log(n)$$

■ **Answer:** Using Master's Theorem: $a = 3$, $b = 3$, $f(n) = n + \log(n)$, $n^{\log_b a} = n$
Master Theorem case 2 applies and thus we get: $T(n) = \Theta(n \log n)$.

■ **Example** Solve the following recurrence relation and arrive asymptotic complexity.

$$T(n) = 4*T(n/3) + n \log(n)$$

■ **Answer:** Using Master's Theorem: $a=4$, $b=3$, $f(n) = n \log(n)$, $n^{\log_b a} = n^{1.2\ldots}$

Since for ε =.1 we get $f(n) \in O(n^{\log_b a - \varepsilon} (= n^{1.2\ldots - \varepsilon}))$, Master Theorem case 1 applies and thus we get: $T(n) = \Theta(n^{\log_3 4})$.

■ **Example** Solve the following recurrence relation and arrive asymptotic complexity.

$$T(n) = 2*T(n-2),\ T(0) = T(1) = 0$$

■ **Answer:** Forward iteration yields T(n)=0 for all n. Hence we have $T(n) = \Theta(1)$.

■ **Example** Answer true or false.

■ **Answer:** Correct answers are in bold

(a) [**T** F] If $f \in o(g)$, then $f(n) \leq g(n)$ for infinitely many n.

(b) [T **F**] If $f \in O(g)$, then $f(n) \leq g(n)$ for infinitely many n. (It should be >=).

(c) [**T** F] $x^{(\lg y)} = y^{(\lg x)}$.

(d) [**T** F] $n! \in O(n^n)$.

(e) [**T** F] A MAX-Heap can be transformed into a MIN-Heap in linear time.

■ **Example** Given the following function, select the following statements are true or false.

$$f(n) = \begin{cases} n^3, \text{ if } n \text{ is even} \\ \\ n, \text{ otherwise} \end{cases}$$

Answers are given in bold.

(a) [**T** F] $f(n) = O(n^3)$ This is the worst out of two terms n^3 and n.
(b) [T **F**] $f(n) = o(n^3)$ It should be o(n).
(c) [T **F**] $f(n) = \Theta(n^3)$.
(d) [T **F**] $f(n) = \Omega(n^3)$ It should be o(n).

■ **Example** Consider the following recursive divide-and-conquer algorithm. How many multiplications ("*") are performed to compute DILEMMA (2, n) for n = 1, 2, 4, 8? . Write the recurrence equation for the number of multiplications ("*") needed to compute DILEMMA (2, n). What is the asymptotic Theta growth class of DILEMMA's worst-case running time? (Use multiplications ("*") as basic operation.) Justify your answer! What value does DILEMMA (a, n) return?

```
Function  DILEMMA( a, n ){
if (n == 1) return a
m = n/2
return DILEMMA( a, m )   *   DILEMMA ( a, m )
}
```

■ **Answer:** Number multiplications ("*") that are performed to compute DILEMMA(2, *n*) for *n* = 1, 2, 4, 8 are given as:

n = 1: 0, *n* = 2: 1, *n* = 4: 3, *n* = 8 : 7

The recurrence equation for the number of multiplications ("*") needed to compute DILEMMA (2, *n*) is given as:

$$T(n) = \begin{cases} 0, \text{ if } n = 1 \\ 2*T(n/2) + 1, \text{ otherwise} \end{cases}$$

Using Master's Theorem: a=2, b=2, f(n)=1, $n^{\log_b a} = n$
Since for ε =.5 we get f(n) ∈ O($n^{\log_b a - \varepsilon}$(= $n.5$)) the Master Theorem case 1 applies and thus we get: T(n)= Θ(n).
This DILEMMA function returns a^n

■ **Example** It is proposed to sort n elements by dividing into k sub lists of n/k elements and each of them insertion sorting is carried out separately. These sub lists are merged using standard merge sorting approach. Discuss about the complexity of this algorithm.

■ **Answer:** Sorting a list of k elements using Insertion Sort takes $\Theta(k^2)$ worst-case time, so sorting n/k such lists takes $n/k \cdot \Theta(k^2) = \Theta(n/k \cdot k^2) = \Theta(nk)$ worst-case time. Now, we propose to merge the lists pairwise, then merge the resulting lists pairwise until there is only one list. the pairwise merging requires Θ(n) time at each level. Thus, total running time for merging becomes Θ (n log(n/k)). Final asymptotic complexity of this method can be said as: Θ (nk + n log(n/k)) . That is: Θ (n logn).

■ **Example** Solve the following recurrence relations and arrive at their asymptotic complexity.

a. $T(n) = 2T(n/2) + n^4$. If we choose ε = 0.01, we have $n^{\lg 2} = O(n^{4-\varepsilon})$, so by the 3rd case of the Master Theorem, we get $T(n) = \Theta(n^4)$.
b. $T(n) = T(7n/10) + n$. If we choose ε = 0.01, we have $n^{\log_{10/7} 1} = 1 = O(n^{1-\varepsilon})$, so by the 3rd case of the Master Theorem, we get $T(n) = \Theta(n)$.
c. $T(n) = 16T(n/4) + n^2$. If we choose ε = 0.01, we have $n^{\log_4 16} = n^2$, so by the 2nd case of the Master Theorem, we get $T(n) = \Theta(n^2 \lg n)$.
d. $T(n) = 7T(n/3) + n^2$. If we choose ε = 0.01, we have $n^{\log_3 7} \approx n^{1.75} = O(n^{2-\varepsilon})$, so by the 3rd case of the Master Theorem, we get $T(n) = \Theta(n^2)$.
e. $T(n) = 7T(n/2) + n^2$. If we choose ε = 0.01, we have $n^{\log_2 7} \approx n^{2.8} = \Omega(n^{2+\varepsilon})$, so by the 1st case of the Master Theorem, we get $T(n) = \Theta(n^{\log_2 7})$.
f. $T(n) = 2T(n/4) + \sqrt{n}$. If we choose ε = 0.01, we have $n^{\log_4 2} = \sqrt{n}$, so by the 2nd case of the Master Theorem, we get $T(n) = \Theta(\sqrt{n} \lg n)$.

g. $T(n) = T(n-2) + n^2$. We have:
$T(n-2) = T(n-4) + (n-2)^2$,
$T(n-4) = T(n-6) + (n-4)^2$, and so on.
So $T(n) = n^2 + (n-2)^2 + (n-4)^2 + \ldots$.
If n = 2k, k integer, $T(n) = \Sigma_{i=1}^{k}(2i)^2 = 4\Sigma_{i=1}^{k}(i^2) = 4\,k(k+1)(2k+1)/6 = n(n/2+1)(n+1)/3 = \Theta(n^3)$.
If n = 2k+1, k integer, $T(n) = \Sigma_{i=1}^{k}(2i+1)^2 = 4\Sigma_{i=1}^{k}(i^2) + k = 4k(k+1)(2k+1)/6 + k = n(n/2+1)(n+1)/3 + n/2 = \Theta(n^3)$.
So $T(n) = \Theta(n^3)$.

■ **Example** Analysing complexity of a divide and conquer method.
Consider that we want to multiply two n-bit numbers A, B. This can be visualised as multiplication of a series of n/2 bit numbers. For example, an 8-bit number 11000101(197) can be represented as two 4-bit numbers as: $1100*2^4+0101=12*16+5=197$. In the same manner, we assume A, B are made up of n/2 bit numbers as shown here.

$$\mathbf{(2^{n/2}X + Y)\,(2^{n/2}W+Z) = 2^{n}XW + (XZ + YW)\,2^{n/2} + YZ}$$

That is we are converting the multiplication of two n-bit numbers into multiplication of four n/2 bit numbers, plus some extra work involved in additions. We are recursively calling multiplication and performing some additions in every recursion. If T (n) be the running time of multiplying two n-bit numbers, the same can be represented now as:

$$T(n) = 4T(n/2) + O(n)$$

- Four multiplications of n/2 bit numbers
- Addition is going to be between numbers that have atmost 2n bits. Thus addition can be O (n).

By applying Master's theorem:

$$a = 4,\ b = 2$$

$$\log_2 4 = 2,\ d = 1$$

The running time complexity becomes $O(n^2)$.

But this is as good as our traditional multiplication algorithm. Since, we now know that multiplications dominate the running time, if we can reduce the number of multiplications to three, which can bring down our T(n) by 25%. To calculate product, we just need the following 3 multiplications separately:

1. (X+Y)(W+Z) 2 additions and one multiplication
2. XW 1 multiplication
3. YZ 1 multiplication

Then we can calculate

$$XZ + YW = (X + Y)(W + Z) - XW - YZ$$

Thus we use three multiplications at the expense of some extra additions and subtractions, which run in constant time(each of O(n) time). Thus,

$$T(n) = 3T(n/2) + O(n)$$

Applying Master's theorem,

$$A = 3,\ b = 2,\ k = 1$$

Thus, $T(n) = O(n^{\log_2 3})$

Since $\log_2 3 \sim 1.5$,

We have reduced the total number of recursive calls in our program.

■ **Example** Give asymptotic complexity of the following function.

```
Function XYZ(n : integer)
for i = 1 to 2n do
for j  =1 to i + 3 do
print("Hi");
```

■ **Answer:** How many times print is executed is given as:

$$\sum_{i=1}^{2n}\sum_{j=1}^{i+3}=\sum_{i=1}^{2n}(i+3)=\sum_{i=1}^{2n}i+\sum_{i=1}^{2n}3=\frac{2n(2n+1)}{2}3.2n$$

$$= 2n^2 + 7n$$

$$2n^2 + 7n = \Theta\,(n^2)$$

■ **Example** Order the functions below according to their order of growth (that is a function f(n) should be listed before g(n) if f(n) = O(g(n)). Enclose functions with same complexity between square brackets.

n^8+n $\quad n+n^2\log^2 n$ $\quad \log^2 n-1$ $\quad \log\log\log n$ $\quad 3^{2^n}$ $\quad 4^{n-1}-30n^2$

$3^{\log^3 n}$ $\quad 2^{\sqrt{\log n}}$ $\quad 17\sqrt{n}$ $\quad n^2+7n$ $\quad 1+1/\sqrt{n}$ $\quad 3^{7n}$

$236n^3$ $\quad 3^{2^{n+1}}$ $\quad \sqrt{n}+\log^2 n$ $\quad 3n+\log^2 n$ $\quad \sqrt{n}\log n$ $\quad \log^2 n+2\log n$

■ **Answer :**

$1+1/\sqrt{n}$ $\quad \log\log\log n$ $\quad [\log^2 n-1$ $\quad \log^2 n+2\log n]$ $\quad 2^{\sqrt{\log n}}$ $\quad [17\sqrt{n}$

$\sqrt{n}+\log^2 n]$ $\quad \sqrt{n}\log n$ $\quad n^2+7n$ $\quad n+n^2\log^2 n$ $\quad 236n^3$ $\quad n^8+n$

$3n^{\log\log n}$ $\quad 3^{\log^3 n}$ $\quad 4^{n-1}-30n^2$ $\quad 3^{7n}$ $\quad 3^{2^n}$ $\quad 3^{2^{n+1}}$

■ **Example** Given an example problem whose complexity is O(!n).

■ **Answer:** A good example is permutations of a string. See the following Java function which has to be invoked with a string whose characters has to be permuted as first argument and an empty string as second argument like: rPrintPerms("rama","");

```
private static void rPrintPerms (String charList, String soFar) {
final int N = charList.length();
if (N==0) {
System.out.println (soFar);
} else {
for (int i=0; i<N; i++) {
String charListMinusIthChar = charList.substring (0,i) + charList.substring(i+1,N);
rPrintPerms (charListMinusIthChar, soFar+charList.charAt(i));
}
}
```

This function is a recursive function with a parameter soFar which keeps track of the current state of the computation. At each step, a for loop adds each possible next character to so Far at the same time it removes it from the original list charList. We may quickly notice that the computational complexity of rprintPerms is factorial in the length of theargument. That is, if string charList has length N, then the running time is *O*(N!).

Is it possible to have two designs for the same data structure that provide the same functionality but are implemented differently?

Yes; it is possible to have multiple implementations of the same data structure that provide the same functionality.

What is the difference between the logical representation of a data structure and the physical representation?

The logical representation is the way that we think of the data being stored in the computer. The physical representation is the way the data is actually organised in the memory cells. For example, we might think of a list of names as being organised in a column. However, when the list is stored in the computer the names are actually strings that follow one after another in a row.

If we have two lists of integers in the range 1..100, describe a method for determining if they are actually the same lists or not (although possibly reordered). Do comment about its time complexity.

Allocate an array of size 100. Run through the first list and count the number of times each value occurs using this array

(thus if we find a value of 42 we increment the 42nd value in the array). Repeat with the second list, only decreasing the values this time. Finally, run through the array we allocated and make sure everything in it is 0. If anything is non-zero, return false. Otherwise, return true. Its time complexity is O(n).

■ **Example** Tromino Tiling

A tromino is a figure composed of three 1 × 1 squares in the shape of an L. Given a $2^n \times 2^n$ checkerboard with 1 missing square, we can recursively tile that square with trominoes.
Here's how we do it:

(1) Split the board into four equal sized squares.
(2) The missing square is in one of these four squares. Recursively tile this square since it is a proper recursive case.
(3) Although the three other squares are not missing squares, we can "create" these recursive cases by tiling one tronimo in the center of the board, where appropriate:

Now, let us do the analysis. Let T(n) be the running time of tiling a nxn square, where n is a perfect power of 2. Then we form the following recurrence relation:
T(n) = 4T(n/2) + O(1), since the extra work involves putting a tile in the middle. Using the master theorem, we have a = 4, b= 2, d=0, and $b^k = 1 < A$. Thus, the running time is $O(n^2)$. This makes sense since we have n^2 to tile and tile at least once each recursive call.

■ **Example** Skyline problem

We have to design a program to assist an architect in drawing the skyline of a city given the locations of the buildings in the city. To make the problem tractable, all buildings are rectangular in shape and they share a common bottom (the city they are built in is very flat). The city is also viewed as two-dimensional. A building is specified by an ordered triple (*Li*, *Hi*, *Ri*) where *Li* and *Ri* are left and right coordinates, respectively, of building *i* and *Hi* is the height of the building. In Fig. 2.7, buildings are shown on the left with triples (1,11,5), (2,6,7), (3,13,9), (12,7,16), (14,3,25), (19,18,22), (23,13,29), (24,4,28) the skyline, shown on the right, is represented by the sequence: (1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)

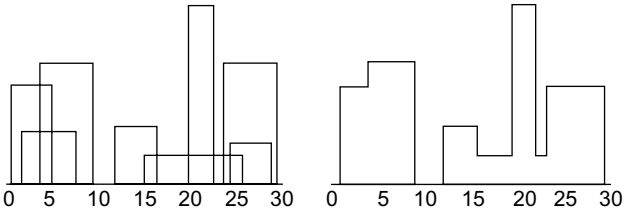


Figure 2.7

We need to merge two skylines—similar to the merge sort
For instance: there are two skylines,

Skyline A: $a_1, h_{11}, a_2, h_{12}, a_3, h_{13}, \ldots, a_n, 0$

Skyline B: $b_1, h_{21}, b_2, h_{22}, b_3, h_{23}, \ldots, b_m, 0$

merge (list of a's, list of b's) form into $(c_1, h_{11}, c_2, h_{21}, c_3, \ldots, c_{n+m}, 0)$

Clearly, we merge the list of a's and b's just like in the standard Merge algorithm. But, it addition to that, we have to properly decide on the correct height in between each set of these boundary values. We can keep two variables, one to store the current height in the first set of buildings and the other to keep the current height in the second set of buildings. Basically, we simply pick the greater of the two to put in the gap.

After we are done, (or while we are processing), we have to eliminate redundant "gaps", such as 8, 15, 9, 15, 12, where there is the same height between the x-coordinates 8 and 9 as there is between the x-coordinates 9 and 12. (Similarly, we will eliminate or never form gaps such as 8, 15, 8, where the x-coordinate does not change.)

Since merging two skylines of size n/2 should take O(n), letting T(n) be the running time of the skyline problem for n buildings, we find that T(n) satisfies the following recurrence:

$$T(n) = 2T(n/2) + O(n)$$

Thus, just like Merge Sort, for the Skyline problem T(n) = O(nlgn).

■ **Example** Write a recursive divide and conquer algorithm for finding the minimum number in an array of unsorted numbers. (b) Analyse the complexity of your algorithm.

(a) modify mergesort: instead of merge just choose min of the two returned scalar.

(b) T(n) = 2T(n/2) + c, (master's theorem) 2>2^0:: T(n) = O(N^(log_2 (2)))=O(N)

```
Function min(array a)
  If (a.size()==1) return a[0]
  Else
        Int x = min(number[0..a.size()/2])
        Int y = min(number[a.size()/2+1..a. size()]
        If (x<y) return x else return y
  End if
End function
```

■ **Example** Analyse the worst case running time of the following program. Here, arrays A and B having integers with more than 1 element. This program outputs the numbers of elements in B equal to the sum of prefix sums in A

```
c ← 0
for i ← 0 to n–1 do
  s ← 0
  for j ← 0 to n–1 do
    s ← s + A[0]
    for k ← 1 to j do
      s ← s + A[k]
  if B[i] = s then
   c ← c+1
return c
```

■ **Answer:** Computational complexity of each line is annoted below.

```
c ← 0                          O(1)
for i ← 0 to n-1 do            O(n)
  s ← 0                        O(n)
  for j ← 0 to n-1 do          O(n^2)
    s ← s + A[0]               O(n^2)
    for k ← 1 to j do          n*(1+2+…+n) = O(n^3)
      s ← s + A[k]             O(n^3)
  if B[i] = s then             O(n)
   c ← c+1                     O(n)
return c                       O(1)
```

By observing the above analysis, the worst case running time can be said as $O(n^3)$.

■ **Example** Suppose that each row of an $n \times n$ array A consists of 1's and 0's such that, in any row of A, all the 1's come before any 0's in that row. That is any row contains a series of 1s followed by a series of 0's as shown below.

11110000

11111100

11000000

11100000

Design and implement an algorithm for finding the row of A that contains the most 1's.

■ **Answer:** Start at the upper left corner of the matrix. Walk across the matrix until a *0* is found. Remember column index of 0. In the case of given matrix, this is 4 (4^{th} indexed column. Here, this four also indicates that there are four 1s in that row). Then walk down wards to next row and traverse in that row until a *1* is found. Compare this 1s column index with previous one save the largest one. This is repeated until the last row is encountered.

■ **Example** Suppose you are given an *n*-element array *A* containing distinct integers that are listed in increasing order. Given a number *k*, describe a recursive algorithm to find two integers in *A* that sum to *k*, if such a pair exists. What is the running time of your algorithm?

■ **Answer:** The following function takes an array A, integers i, j and k. Here, i and j are the two elements whose sum is required to be k. If such a pair exists the function has to return true else false. We know that array is having elements in ascending order. We call this function with i value as the index of first element (0) and j value as the index of the last element (n-1). In the function, we sum ith and jth elements. If it is less than k then we have to search for another element whose value is greater than ith element. Thus, we search for the pair between i+1 and j. If the sum is greater than k, then we have to consider another element that is less than jth element. Thus, we search for the pair between i and j-1 recursively. This, we repeat recursively till we find the solution.

```
Algorithm FindPair(A, i, j, k)
    if i = j then
       return false
    else
       if A[i] + A[j] < k then
              return FindPair(A, i+1, j, k)
       else
              if A[i]+A[j] > k then
                     return FindPair(A, i, j-1, k)
              else
                     return true
              endif
       endif
    endif
```

For each call, the running time is O(1).
For each call, at most one recursive call is activated.
Since the input size decrease by 1 at each recursive call, we can represent the running time as:

$$T(n) = T(n-1) + O(1)$$

Obviously, $T(n) = O(n)$.

■ **Example** The following algorithm is proposed for calculating a^n for an integer value of n. Analyse its worst-case running time, and express it using Big-Oh notation. Do explain how it works by preparing a snap shot.

```
Algorithm power(a,n)
k =n
b =1
c =a
while k > 0 do
    if k mod 2 = 0 then
       k ← k/2
       c ← c*c
    else
       k ← k-1
       b ← b*c
return b
```

■ **Answer:** The running time is O(log*n*) for the following reasons. The initialisation and the **if** statement and its contents take constant time. So, the running time is determined by how many iterations in the **while** loop. Note that if *k* is even, it gets halved, and if it is odd, it gets decremented, and halved in the next iteration. So at least every second iteration of the **while** loop halves *k*. One can halve a number *n* at most log*n* times before n <= 1. So, this suggest that the while loop will run for O(log*n*) iterations.

Snap shot assuming n as 19.

k	c	b	
19	a	1	
			k is odd
18		a	
			k is even
9	a^2		
			k is odd
8		a^3	
			k is even
4	a^4		
			k is even
2	a^8		
			k is even
1	a^{16}		
			k is odd
0		a^{19}	
			Now result is available in b.

■ E**xample** Assume that we have two arrays A and B which contains roll numbers of the students who has registered in two courses. The following algorithm is proposed to count the number of male students enrolled in both of 2 courses (A and B). Assume that the class sizes of both courses are n, n > 50.

```
function AlgorithmA(A,B)
1  result = 0;
2  for i=1 to n
3    if A[i] is male
4       for j=1 to n
5               if A[i]=B[j]
6               result = result + 1;
7       break; //stop iterating the inner for-loop.
8  return result;
```

What are the best, average, and worst cases of this algorithm [Assume that the courses are equally welcomed by male and female students, and all students enrolled in A will enroll or not enroll in B with equal possibility]?
Best case: All course A students are female

Average case: Half of course A students are male and half of these male students are found in course B's list, evenly distributed.
Worst case: All course A students are male and none of them study course B.

Best case: n Worst case: n^2 Average case: n^2

By direct observation, state the order of growth of the running time of each of the 3 cases.
Formulate the running times with abstract coefficients to prove your observation.
Let t_{3a} = number of male students in course A.
Let t_4 = number of times line 4 is executed.
Let t_5 = number of times line 5 is executed.
Let t_6 = number of times line 6 is executed.
$T(n)=C1+C2(n+1)+C3(n)+C4(t_4)+C5(t_5)+C6(t_6)+C7(t_6)+C8$

Best case: $t_{3a}= 0$, $t_4 = t_5 = t_6 = 0$

$\Rightarrow T(n)= C1 + C2 (n +1) + C3(n) + C8 = An + B$

Worst case: $t_{3a}= n$, $t_4= n (n+1)$,$t_5 = n^2$, $t_6= 0$

$\Rightarrow T(n)= C1+C2(n+1) + C3(n) + C4(n^*(n+1)) + C5(n^2) + C6(0) + C7(0) + C8 = An^2 + Bn + C$

Average case: $t_{3a}= n/2$, $t_6 = n/4$, $t_5 = n/4^*(n) + n/4^*n/2 = (3/8)n^2$, $t_4 = n/4^*(n + 1) + n/4^*n/2 = (3/8)n^2 + n/4$,

$T(n) = C1 + C2 (n+1) + C3(n) + C4((3/8)n^2 + n/4) + C5^* ((3/8)n^2) + C6(n/4) + C7(n/4) + C8 = An^2 + Bn + C$

Describe the running times in terms of

a. Asymptotic Upper Bound,
b. Asymptotic Lower Bound, and/or
c. Asymptotic Tight Bound

Best case: (c) Θ (n) Worst case: (c) Θ (n^2) Average case: (c) Θ (n^2) AlgorithmA: (a) O(n^2), (b) Ω (n)

2.4.1.4 A Note on Reducing Space Complexity

We know that space complexity of an algorithms is also an important criterion while comparing and selecting the algorithms for practical problems. Also, it is vital to reduce the space requirements of algorithms. In a nutshell, we may employ some alterantive physical storage schemas to reduce memory space requirements. However, it poses another difficulty. That is, if we propose another schema of storage; then we also need to propose (new) mechanisms to do the operations. For example, if we propose to store a 2D matrix in a 1-D array, then all the operations such as additions, subtractions, etc. which we carry on 2-D matrix has to modified such that they assume matrix is in 1-D array. In the following, we propose some examples which illuminates the reader about reducing memory space requirements of algorithms.

■ **Example** Propose a method to map a symmetric matrix into a 1-D array along with methods to add, subtract, multiply symmetric matrices which are in the 1-D array fashion.

We know that symmetric matrices will be having their upper and lower triangular portion elements as same. Thus, to conserve memory, we propose to store only lower trainangular part of the matrix including main diagonal elements. If one observes, we can find that ith row jth column element of the symmetric matrix is stored in location i*(i+1)/2+j of 1-D array. For example, 2nd row 2nd column element, i.e., x5 will be available at: 2*(2+1)/2+2=3+2=5.

	0	1	2	3
0	x0	x1	x3	x6
1	x1	x2	x4	x7
2	x3	x4	x5	x8
3	x6	x7	x8	x9

x0	x1	x2	x3	x4	x5	x6	x7	x8	x9
0	1	2	3	4	5	6	7	8	9

Total number of elements in the 2-D symmtric matrix = n^2

Number of elements needed in the 1-D representation= 1+2+…+n = n(n+1)/2
Thus, a saving of almost 50%.

If we want ith row jth column element of the symmetric matrix, its location in the 1-D array can be calculated as: i*(i+1)/2+j. However, this will not work for upper triangular portion elements. Thus, if we want upper triangular portion elements of the 2-D array, we simply exchange their row (i) and column(j) indexes and then apply the above formula to get the required element. This became possible because of symmetry.

The following program demonstrates the storage of a symmetric matrix in a 1-D array and accessing the same. Remember, really 2-D array information is in 1-D fashion. However, users can still work at 2-D notation by using the above mapping function.

```
#include<stdio.h>
int Element(int s[], int i, int j){
  return s[i*(i+1)/2+j];
 }
int main(){
int i,j,k,s[200],n,a[10][10];
printf("Enter Size of the matrix\n");
scanf("%d", &n);
for(i=0;i<n;i++)
for(j=0;j<n;j++) scanf("%d", &a[i][j]);
for(i=0,k=0;i<n;i++)
for(j=0;j<=i;j++) s[k++]=a[i][j];
/* here k becomes the total elements of 1-D array */
printf("Symmetric Matrix Data From 1-D Array\n");
for(i=0;i<k;i++)printf("%d\n", s[i]);
printf("Enter Row and Columen Element of Any element\n");
scanf("%d%d", &i, &j);
if(j>i) {
           int t=i;
           i=j;
           j=t;
          }
printf("Element Value=%d\n", Element(s,i,j));
  return 0;
}
```

Now, let us discuss about how to add two symmetric matrices which are represented in 1-D array fashion as explained above.

Consider First matrix A and its 1-D array representations are:

1	9	1	9
9	**2**	2	8
1	**2**	**3**	1
9	**8**	**4**	**1**

1	9	2	1	2	3	9	8	4	1

Consider Second symmetric matrix and its 1-D representations are:

1	7	1	9
7	**2**	7	4
1	**7**	**3**	1
9	**4**	**4**	**1**

1	7	2	1	7	3	9	4	4	1

Now their sum matrix and its 1-D representations are:

1+1	9+7	1+1	9+9
9+7	**2+2**	2+7	8+4
1+1	**2+7**	**3+3**	1+1
9+9	**8+4**	**4+4**	**1+1**

1+1	9+7	2+2	1+1	2+7	3+3	9+9	8+4	4+4	1+1

By observing the above workout, we can say that adding two 2-D symmetrix matrixes in this representation is same as adding their resective 1-D representations element by element. It is true with the subtraction of two symmetric matrices. The following function allows us to do addition of two symmetric matrices which are in their1-D representation.

We know that in nxn elements, total n*(n+1)/2 elements are stored in the 1-D array. Thus, in the function, we allocate a dynamic array to store n*(n+1)/2 elements. The address of this array is returned as the resultant matrix in 1-D representation.

```
int * AddSymMat(int a[], int b[], int n){
int i, *c;
C=(int*) malloc( n*(n+1)/2*sizeof(int));
for(i=0; i< n*(n+1)/2; i++) C[i]=a[i]+b[i];
return C;
}
```

We can carry subtraction also in the same fashion. For multiplication, we propose the following function. Verify whether it will give the expected results are not. Remember, we need to calculate only lower triangular portion of the product of two symmetric matrices.

```
int * ProdSymMat(int a[], int b[], int n){
int i, j,k,l,*c;
C=(int*) malloc( n*(n+1)/2*sizeof(int));
l=0;
for(i=0; i<n; i++)
for(j=0;j<=i;j++){
C[l]=0;
for(k=0;k<n;k++) C[l] += Element(a, i,k) * Element(b,k,j);
l++;
}
return C;
}
```

■ **Example** Now consider storing two symmetric matrices of same size (nxn) in a 2-D array to conserve space.

A

1	9	1	9
9	2	2	8
1	2	3	1
9	8	4	1

B

1	7	1	9
7	2	9	4
1	9	3	1
9	4	4	1

1	1	4	4	9
9	2	3	9	1
1	2	3	2	7
9	8	4	1	1

Resultant Matrix C

As we know that the syemmtric matrices will be having redundancy, we proposed to store two nxn symmetric matrices lower triangular portions in a nx(n+1) matrix. Here, we may find a saving of almost 50%. That is, as such for both the matrices A and B together we need $2n^2$ elements. If we store both in a 2-D array like C, we need n^2+n elements. This is 2-D array to 2-D array mapping.

Probable mapping steps are:

```
for(i=0;i<n;i++)
for(j=0;j<=I;j++) {
                    C[i][j]=A[i][j];
                    C[n-1-i][n-j]=B[i][j];
                  }
```

Of course, if we want to ith row j'th column element of Matrix A, it can be accessed simply as ith row jth column element of C as it is stored like that way. Similarly, if we want ith row jth column element of matrix B, the same can be accessed as C[n-1-i][n-j] as it is stored like that way as shown in the above code fragment. That is, what ever way we have stored the element, the same way we can access. However, with both the matrices A and B, if we want upper triangular portion elements, then we can exchange their row and column elements and then access from C.

Sparse Matrices

In Engineering, we may encounter a special type of matrix known as sparse matrix. From the name itself, one can guess what it is.

An *m* x *n* matrix (table) is said to be *sparse* if "many" of its elements are zero (empty). A matrix which is not sparse is *dense*. The boundary between a dense and a sparse matrix is not precisely defined. Diagonal and tridiagonal *n* x *n* matrices are sparse since they have O(n) nonzero terms and $O(n^2)$ zero terms.

Definition: A matrix *M* is *diagonal* iff M(i, j) = 0 for $i \neq j$.

Definition: A matrix *M* is *tridiagonal* iff M(i, j) = 0 for $|i - j| > 1$.

Both of these special matrices are special cases of the more general square band matrix in which the non-zero elements are on a band which is centered about the main diagonal. The following are sample 6x6 diagonal and tri-diagonal matrices.

2	0	0	0	0	0
0	1	0	0	0	0
0	0	4	0	0	0
0	0	0	6	0	0
0	0	0	0	5	0
0	0	0	0	0	3

Diagonal Matrix

4	2	0	0	0	0
1	3	1	0	0	0
0	4	5	2	0	0
0	0	2	9	4	0
0	0	0	6	3	3
0	0	0	0	1	2

Tridiagonal Matrix

■ **Example** Propose means of reducing memory requirements of a diagonal matrix. Also, mention about accessing matrix elements.

■ **Answer:** We propose to store only diagonal elements in a 1-D array. For example, the above diagonal matrix is stored in a 1-D array with 6 elements as shown below:

2	1	4	6	5	3

Thus, we can save lot of space. That is, to store a nxn diagonal matrix we need n^2 locations, whereas in this representation we need a 1-D array with n elements only.

Accesing the elements is also easy. Say, we want ith row jth column element of diagonal matrix A, then first we compare row and column indexes i and j. If they are same then simply we print ith element from the 1-D array else we print element value as 0. Thus, we have converted a 2-D system into 1-D storage organisation to save memory space.

■ **Example** Discuss what happens if we propose to store a diagonal matrix with its off diagonal elements only as non-zero elements in a 1-D array as shown below. This type of matrices are called as antidiagonal matrices.

0	0	0	x0
0	0	x1	0
0	x2	0	0
x3	0	0	0

1-D representation:

x0	x1	x2	X3

Here also accesing the elements is easy. Say, we want ith row jth column element of diagonal matrix, then first we compare row and column indexes i and j. If j is n-1-i, where n is the size of the matrix then simply we print ith element from the 1-D array else we print required element value as 0.

■ **Example** Let we have proposed to store tridiagonal elements in a 1-D array. Propose a code fragment which does it. Assume A is the tridiagonal matrix of size nxn and X is a 1-D array in which A's non-zero elements are proposed to be stored as shown below. Analyse memory saving. Also, explain how to access ith row jth column element of A from the 1-D array X.

4	2	1	3	1	4	5	2	2	9	4	6	3	3	1	2

■ **Answer:** If we observe the tridiagonal matrix, we may find that except top and bottom rows all the remaining elements will be having 3 non-zero elements. Thus, total number of non-zero elements which we need to store in the 1-D array are: 2+2+(n-2)*3=3n-2. Thus, instead of n^2 locations of matrix A, with 3n-2 locations itself A's details are stored in the 1-D array.

The following code fragment does the required mapping of 2-D to 1-D.

```
int k=0,lower,upper,i,j;
for(i=0;i<n;i++){
lower=i-1;
if(lower<0)lower=0;
upper=i+1;
if(upper>=n)upper=n-1;
for(j=lower;j<=upper;j++)X[k++]=A[i][j];
}
```

Now, let us discuss about how to access ith row jth column element from the 1-D array X. If one observes the tridiagonal matrix, we may find that only non-zero elements are along the principal diagonal. Non-zero elements in any row are: principal diagonal element and one element of its previous column and one more element in the next column. However, first end last rows are having only two elements.

Thus, if absolute difference of row and column indexes of an element is greater than 1 then its value is 0 and is not available in the 1-D array X. Also, 0^{th} row elements that A[0][0], A[0][1] are available at X[0], X[1], respectively. Thus, jth

column element of 0^{th} row will be at jth location in the 1-D array X. Similarly, the following pseudo code illustrates how to access ith row jth column of matrix A from 1-D array X.

```
if( abs(i-j)>1) print("0\n");
else if(i==0) print("%d\n", X[j]);
else print("%d\n", X[ i*3-1+(j -(i-1)) ]);
```

■ **Example** Propose how to store a tridiagonal matrix whose non-zero elements are along the other diagonal in a 1-D array. Also, arrive at logic to access ith row jth column element of tridiagonal matrix from 1-D array.

0	0	0	0	x0	x1
0	0	0	x2	x3	x4
0	0	x5	x6	x7	0
0	x8	x9	x10	0	0
x11	x12	x13	0	0	0
x14	x15	0	0	0	0

These non-zero elements are stored in 1-D array as shown below:

x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15

The following code fragment does the required mapping of 2-D to 1-D.

```
int k=0,lower,upper,i,j;
for(i=0;i<n;i++){
lower=n-1-i-1;
if(lower<0)lower=0;
upper=n-i;
if(upper>=n)upper=n-1;
for(j=lower;j<=upper;j++)X[k++]=A[i][j];
}
```

The following pseudo code illustrates how to access ith row jth column of matrix A from 1-D array X.

```
if( abs((n-i-1)-j)>1) print("0\n");
else if(i==0) print("%d\n", X[j-n+2]);
else if(i<n-1) print("%d\n", X[ i*3-1+(j -(n-1-i-1)) ]);
else printf("%d\n", X[i*3-1+j] );
```

■ **Example** Propose a method to store a band matrix of band width w (w is an odd number) in a 1-D array. Also, explain how to access ith row jth column element of band matrix from the 1-D array.

X0	X1	X2	X3	0	0	0	0	0
X4	X5	X6	X7	X8	0	0	0	0
X9	X10	X11	X12	X13	X14	0	0	0
X15	X16	X17	X18	X19	X20	X21	0	0
0	X22	X23	X24	X25	X26	X27	X28	0
0	0	X29	X30	X31	X32	X33	X34	X35
0	0	0	X36	X37	X38	X39	X40	X41
0	0	0	0	X42	X43	X44	X45	X46
0	0	0	0	0	X47	X48	X49	X50

The following code fragment stores non-zero element of the above band matrix in a 1-D array X.

```
int k=0,lower,upper,i,j,w,d;
scanf("%d", &w); //w should be odd number
d=w/2;
for(i=0;i<n;i++){
lower=i-d;
if(lower<0)lower=0;
upper=i+d;
if(upper>=n)upper=n-1;
for(j=lower;j<=upper;j++)X[k++]=A[i][j];
}
```

The following code fragment can be used to access ith row jth column element of the band matrix.

```
if( abs(i-j)>d) print("0\n");
else
{
if(i<d)k=i*d+i*(i+1)/2+j;
else if(i<n-d) k=d*d+d*(d+1)/2+ (i-d)*w + (j -(i-d));
else
{
 l=j-(n-d);
 k=d*d+ d*(d+1)/2+ (n-2*d)*w + w*l -l*(l+1)/2 + (j-(i-d));
}
print("%d\n", X[k]);
}
```

■ **Example** Propose a method to store the non-zero elements of the following band matrix of band width w (w is an odd number) in a 1-D array. Also, explain how to access ith row jth column element of band matrix from the 1-D array.

0	0	0	0	0	X0	X1	X2	X3
0	0	0	0	X4	X5	X6	X7	X8
0	0	0	X9	X10	X11	X12	X13	X14
0	0	X15	X16	X17	X18	X19	X20	X21
0	X22	X23	X24	X25	X26	X27	X28	0
X29	X30	X31	X32	X33	X34	X35	0	0
X36	X37	X38	X39	X40	X41	0	0	0
X42	X43	X44	X45	X46				
X47	X48	X49	X50					

The following code fragment stores non-zero element of the above band matrix in a 1-D array X.

```
int k=0,lower,upper,i,j,w,d;
scanf("%d", &w); //w should be odd number
```

```
d=w/2;
for(i=0;i<n;i++){
lower=n-1-i-d;
if(lower<0)lower=0;
upper=n-1-i+d;
if(upper>=n)upper=n-1;
for(j=lower;j<=upper;j++)X[k++]=A[i][j];
}
```

The following code fragment can be used to access ith row jth column element of the band matrix.

```
if( abs(n-1-i-j)>d) print("0\n");
else
{
if(i<d)k=i*d+i*(i+1)/2+(j-(n-1-i-d));
else if(i<n-d) k=d*d+d*(d+1)/2+ (i-d)*w + (j -(n-1-i-d));
else
{
 l=j-(n-d);
 k=d*d+ d*(d+1)/2+ (n-2*d)*w + w*l -l*(l+1)/2 + j;
}
print("%d\n", X[k]);
}
```

■ **Example** Is an *n* x *n* triangular (either upper or lower) matrix sparse? A triangular matrix will have at least *n(n-1)/2* zero terms and at most *n(n+1)/2* nonzero terms. For the representation schemes that we are about to examine to be competitive over the standard two-dimensional array representation, it will turn out that the number of nonzero terms will need to be less than *n2/3* and in some cases less than *n2/5*. Thus, in the context of the representation schemes we are about to see, a triangular matrix is considered to be dense rather than sparse.

Regular Sparse Matrices

Sparse matrices which are either diagonal or tridiagonal have sufficient structure in their nonzero regions to allow fairly simple representation schemes to be developed whose space requirements equal the size of the nonzero region. Previous examples illustrates this.

Irregular Sparse Matrices

An irregular sparse matrix has a nonzero region in which no discernable pattern exists.

Usually irregular sparse matrix contains very less number of meaningful (non-zero) elements compared to the total number elements in the matrix. Thus, there will be wastage of memory. In order to save memory to store the details of irregular sparse matrix, we use two prominent approaches:

1. Array
2. Linked list

Array Based Sparse Matrix Representation

Here, we propose to store each meaningful element value along with its row and column indexes in a 2-D array. As number of meaningful elements are very less, we will be saving memory considerably. For example, consider the following sparse Matrix:

1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	7	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	8	0	0	0	6	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	3	0	0	0

Its alternative representation:

12	10	5
0	0	1
4	5	7
5	4	8
5	8	6
11	6	3

In the first row we have stored sparse matrix size (rows, columns) and number of meaningful elements. Next rows contains row and column indexes and value of each of meaningful element in the sparse matrix. We are sure that user has observed the saving in the memory requirements in this storage schema.

■ **Example 5** Calculating sparse matrix representation of the transpose of a sparse matrix

■ **Answer:** We are sure that if we transpose the sparse matrix, the resultant also sparse matrix. Also, total number of elements will not change.

We do not want to calculate first original sparse matrix given its representation, then its transpose followed by its sparse representation. We want a direct step approach. Of course, users should remember that all in sparse matrix representation, we have stored meaningful elements of each row one after another. Thus, if we want sparse matrix representation of original sparse matrix then we have to find first those elements which are of 0^{th} column first, then second column, and vice versa and store.

Assuming A is the sparse matrix representation, we know there will be A[0][2] meaningful elements. Also, transpose matrix rows and columns gets reversed. Thus, transpose matrix B top row becomes:

```
B[0][0]=A[0][1];
B[0][1]=A[0][0];
B[0][2]=A[0][2];
k=1;
for(i=0; i<A[0][1]; i++)   /* column by column we traverse on the source matrix*/
{
for(j=1;j<=A[0][2];j++)
if(A[j][1]== i){
                  B[k][0]=i;              /* column index is made as row index */
                       B[k][1]=A[j][0];    /* row index is made as column index*/
                       B[k][2]=A[j][2];
                       k++;
}
}
```

■ **Example 6** Adding two sparse matrices and getting the sparse matrix representation of resultant sparse matrix. Of course, here also, we can add two sparse matrices if they are of same size.

Also, if two matrices are having m and n number of meaningful elements, then the resultant matrix will be having at most m+n elements. Why? Guess.

We assume A,B are sparse matrix representations as shown here. We want the result to be available in another matrix C for which sufficient memory is already allocated.

A

12	10	5
0	0	1
4	5	7
5	4	8
5	8	6
11	6	3

B

12	10	4
0	0	1
4	5	9
5	0	8
5	8	6

If we observe the size of A and B, we may find they are matching. Thus, addition operation is permitted. Also, if we observe, both the matrices A and B are having elements at 4^{th} row 5^{th} column. If we add two sparse matrices, we really get one element with their sum as the element value. Thus, total number of meaningful elements in the resultant matrix will be less than their total.

```
C[0][0]=A[0][0];
C[0][1]=A[0][1];
K=1;
m=A[0][2]; /* Number of meaningful elements in A*/
n=B[0][2;   /* Number of meaningful elements in B*/
for(i=1;i<=m;i++)
for(j=1;j<=n;j++)
{
if( A[i][0] < B[j][0]) {
                              for(l=0;l<3;l++) C[k][l]=A[i][l];
                                     k++;
                                     i++;
                                  }
else if( A[i][0] > B[j][0]) {
for(l=0;l<3;l++) C[k][l]=B[J][l];
                              k++;
                               j++;
                                           }
else if( A[i][1] < B[j][1]) {
for(l=0;l<3;l++) C[k][l]=A[i][l];
                             k++;
                             i++;
                                              }
else if( A[i][1] > B[j][1]) {
for(l=0;l<3;l++) C[k][l]=B[J][l];
                             k++;
                             j++;
                                   }
else
```

```
{
for(l=0;l<3;l++) C[k][l]=A[i]l]+B[J][l];
                                k++;
               i++;
                                 j++;
}
}
C[0][2]=k /* K is the total number of element */
```

Linked List based Solution

Without regularity in the nonzero region it is highly unlikely that a standard representation, such as a two-dimensional array, would provide an efficient representation of the matrix. On the other hand, if there is a high degree of regularity or structure in the nonzero region, then an efficient representation structure of the nonzero region can typically be developed using standard linked lists that will require space equal in size to the nonzero region. We will not examine these highly regular sparse matrices, our concern is finding a suitable representation scheme for an irregular sparse matrix. Consider the following irregular 4 x 8 sparse matrix shown in Table 2.3.

Table 2.3 Irregular 4 × 8 Sparse Matrix

0	0	0	2	0	0	1	0
0	6	0	0	7	0	0	3
0	0	0	9	0	8	0	0
0	4	5	0	0	0	0	0

Notice in the irregular sparse matrix shown in Table 2.3 of the 32 total elements in the matrix that fully 23 or 71.8% of the cells have zero value. Clearly this is a sparse matrix. Further notice that there appears to be no obvious regularity to where the nonzero elements occur.

The nonzero elements of the sparse matrix in Table 2.3 can be mapped into a linear list. If this list is organized in row-major order we would have the following: 2, 1, 6, 7, 3, 9, 8, 4, 5. To be able to reconstruct the matrix structure, the original row and column must be recorded for each nonzero element in the matrix. The linear list would contain nodes that look like the one shown in Fig. 2.8.

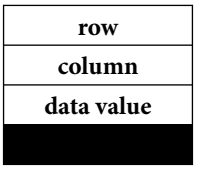


Figure 2.8 Node for representation of the sparse matrix

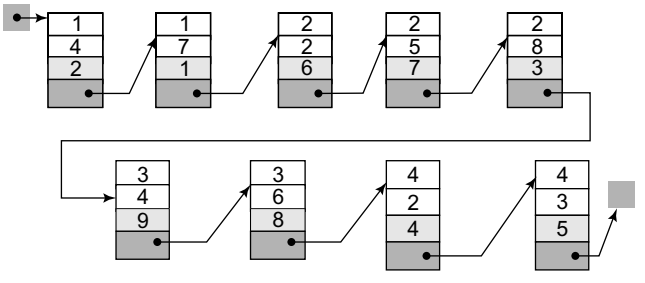


Figure 2.9 Linear list representation of sparse matrix of Table 2.3

Notice that the linked list (linear list) representation of the sparse matrix, while efficient in terms of space when compared with a two-dimensional array will not be particularly efficient for insertion and retrieval operations (although it is better than if a two-dimensional array is used).

Question to think about: Do you think a skip list would improve this implementation enough to warrant the additional overhead?

The sparse matrix representation illustrated in Fig. 2.9 is a fairly common technique for representing sparse matrices and provides fairly efficient behavior for algorithms such as matrix transpose, addition, and multiplication.

Table 2.4 Transpose of the matrix shown in Table 2.3

0	0	0	0
0	6	0	4
0	0	0	5
2	0	9	0
0	7	0	0
0	0	8	0
1	0	0	0
0	3	0	0

Notice how easy it is to transpose the sparse matrix of Table 2.3 when it is represented in the linear list format shown in Fig. 2.9. The transposed matrix of Table 2.4 is shown in its linear list representation in Fig. 2.10. What was done to this list to produce this result? What is the time complexity of this task?

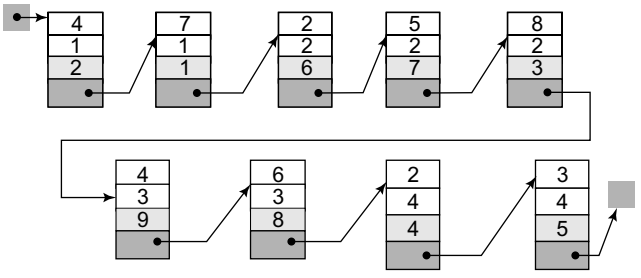


Figure 2.10 Linear list representation of the transpose of the list represented in Figure 2.9

Using the linear list to represent the sparse matrix provided us a very fast way to perform the transpose of the matrix, but at what cost to the retrieval process? Notice that the representation of the transposed sparse matrix is no longer ordered in row-major fashion, but is now ordered in column-major fashion.

Our concern with the representation in Fig. 2.9 is that access operations into the sparse matrix are not efficient for random access into the matrix. Any application performing predominantly access (look-up) operations into the sparse matrix will not perform with anywhere near optimal behavior using this representation. What we need is a better representation when considering random access into the sparse matrix as the dominant operation. To illustrate the access problem this representation presents, consider the sparse matrix addition illustrated below:

0	0	0	0
4	0	2	0
0	1	0	0
0	0	2	0
0	0	0	4

Matrix A

0	0	0	0
4	0	0	1
0	0	3	0
0	0	2	0
0	0	0	1

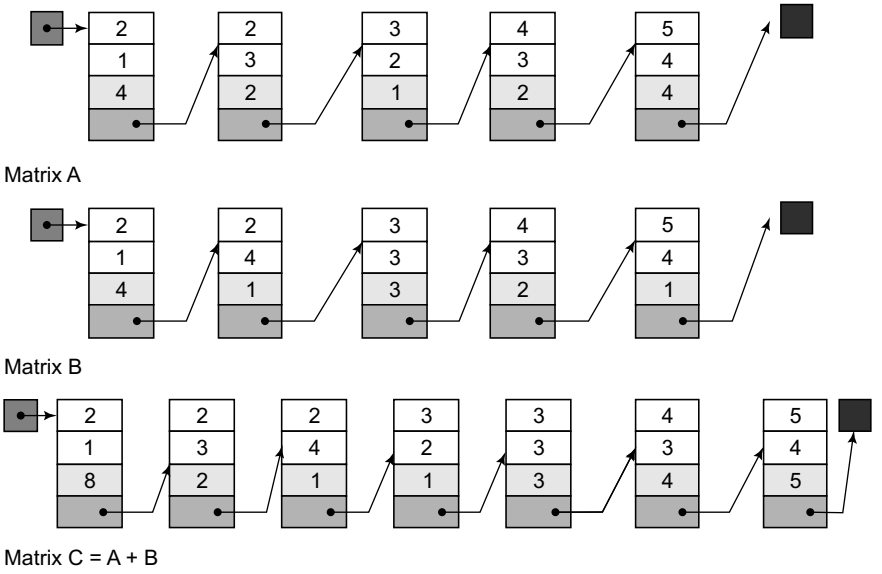
Matrix B

The result of A + B is matrix C shown here, recall that matrix addition is defined only when the two operand matrices have the same dimensions and is defined as $C(i,j) = A(i,j) + B(i,j);\ 1 \le i \le m,\ 1 \le j \le n$ where A and B are $m \times n$ matrices:

0	0	0	0
8	0	2	1
0	1	3	0
0	0	4	0
0	0	0	5

Result matrix of A + B

Consider the representation of the matrices *A* and *B* in linear list format shown below:



Think about the operational aspects of performing the addition of matrices A and B using the linear list representation. Since the two matrices are stored in a row-major fashion a simple iteration through each list will suffice for the addition operation. For example, beginning in matrix A, we find the first nonzero element to be in position (2, 1), this too happens to be the first nonzero element in matrix B, so this sum is computed an stored in the first element of the list which represents the sum. Advancing iterators in both the lists we would find that the next nonzero element in A is in position (2,3), but there is no corresponding element in B since the iterator in B is on a node which corresponds to position (2,4). Thus, a single pass through each list will produce the sum of the two sparse matrices. Once again, we can see that this is a fairly efficient way in which to produce the sum of two sparse matrices. Multiplication is only slightly more difficult, try it yourself to see.

There is some possibility that the addition (or multiplication) of two sparse matrices may produce a dense matrix, but we won't worry about that case here as we are only concerned with the representation of irregular sparse matrices.

It turns out that the representation we desire does not represent a major change from what we have seen above. The representation scheme that will be the structure of choice for representing sparse matrices when random access is the dominant operational activity is a representation using many linear lists. Notice that none of the operations of transpose, addition, or multiplication required a truly random access to the matrix (although multiplication was getting there) and this is why the single linear list representation was reasonably efficient. However, for truly random searches the single linear list will not be efficient enough and a more suitable representation will be found in the multiple linear list representation. A multiple linear list implementation of the irregular sparse matrix is shown in Table 2.3.

Figure 2.11 illustrates a multiple linear list representation of the irregular sparse matrix shown in Table 2.3. The linear list across the top of the diagram represents the columns of the matrix while the linear list farthest to the left represents the rows of the matrix. The multiple lists in the interior of the diagram illustrate the nonzero elements of the sparse matrix. Notice that each element participates in both a row and column list. Thus access to an element may be either through its row or column address.

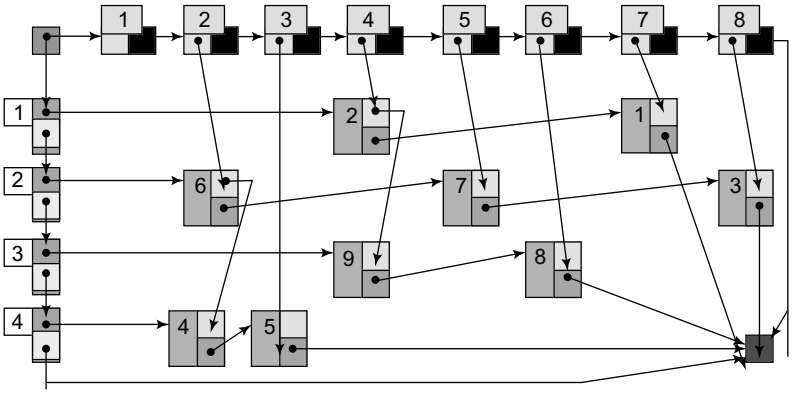


Figure 2.11 Multiple linear list representation of Irregular sparse matrix

Notice that the transpose of the matrix from Table 2.3 already exists in the representation of Figure 2.11. Simply interchange the row and column lists and you have the transpose matrix. This takes O(1) time.

Toeplitz Matrices

An *nxn* matrix *X* is a Toeplitz matrix if *X(i,j) = X(i-1, j-1)* for all *i* and *j* where *i >1* and *j > 1*. Figure 2.12 illustrates a 4x4 Toeplitz matrix.

6	2	4	5
1	6	2	4
3	1	6	2
8	3	1	6

Figure 2.12 A 4 x 4 Toeplitz Matrix

A Toeplitz matrix can be represented by a sparse matrix. The reason for this is that the number of distinct elements in a Toeplitz matrix is at most $2n - 1$. Recall that the definition given at the beginning of these notes for a sparse matrix indicated that a sparse matrix contained O(n) nonzero elements. In the case of a Toeplitz matrix rather than nonzero elements we need only to represent the distinct elements, that the Toeplitz matrix can be represented as a sparse matrix. Figure 2.13 illustrates the linear list implementation (in row major order) of the Toeplitz matrix shown in Fig. 2.12.

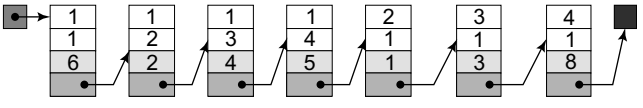


Figure 2.13 Linear list representation of the Toeplitz matrix of Figure 2.12

C-matrices

An *nxn* C-matrix is one in which all elements other than those in row 1, row *n*, and column 1 are zero. A C-matrix contains at most *3n–2* nonzero elements and is thus considered to be a sparse matrix. Figure 2.14 illustrates a 6x6 C-matrix.

4	3	8	1	2	6
5	0	0	0	0	0
8	0	0	0	0	0
2	0	0	0	0	0
4	0	0	0	0	0
3	5	2	8	22	3

Figure 2.14 A 6x6 C-matrix

Notice that Figure 2.13 contains exactly 16 nonzero elements which is (3(6) – 2).

As before, this sparse matrix is easily represented using the linear list structure.

A note on Row and Column Major Order Storage

Column-major and row-major storage

When arrays are physically stored in memory, the elements have to be somehow laid out linearly as RAM is linearly accessible. There are two main ways to do this, by row major order and by column major order.

We say that a matrix(2-D array) is stored in **row-major** order if it is stored row by row. The entire first row is stored first, followed by the entire second row, and so on. Consider for example the matrix

$$A = \begin{bmatrix} 8 & 2 & 2 & 9 \\ 9 & 1 & 4 & 4 \\ 3 & 5 & 4 & 5 \end{bmatrix}$$

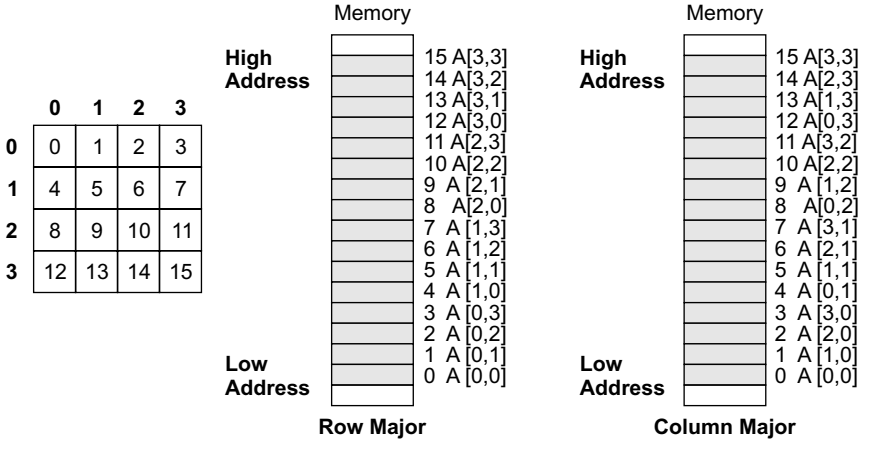
If this matrix is stored in row-major order, then the entries are laid out in memory as follows:

8 2 2 9 9 1 4 4 3 5 4 5

On the other hand, a matrix is stored in **column-major** order if it is stored column by column, starting with the entire first column, followed by the entire second column, and so on. If the above matrix is stored in column-major order, it is laid out as follows:

8 9 3 2 1 5 2 4 4 9 4 5

FORTRAN language uses column major order of storage while C family of languages use row major order. The following figure illustrates how a 2-D array is physically stored in RAM.



While accessing elements of arrays, we need to know offset of the element with respect to first element. By adding this offset to base address, we can calculate the address of the element such that it can be accessed.

For a 2-D arrays, to know offset of an element we can use the following formulas. Here, we are making an important assumption that in any dimension indexes starts from 0.

Row-Major order

offset = rowIndex*NUMCOLS + columnIndex

Column-Major order

offset = rowIndex + columnIndex*NUMROWS

Similarly, for a 3-D array with given number planes where each plane contains some number of rows and columns. For example, int **a[number of planes][number of rows][number of columns]** is a 3-D array in C language. Here, offset in row major order is given as:

=(planeIndex*rows+rowIndex)*cols+columnIndex

That is, if a is an 3-D array of size N1xN2xN3 then a[i][j][k] can be calculated as:
(i*N2+j)*N3+k

=i*N2*N3+j*N3+k

Similarly, if a is an 4-D array of size N1xN2xN3xN4 then a[i][j][k][l] can be calculated as:
((i*N2+j)*N3+k)*N4+l.

Similarly, if a is an 5-D array of size N1xN2xN3xN4xN5 then a[i][j][k][l][m] can be calculated as:
(((i*N2+j)*N3+k)*N4+l)*N5+m.

We can calculate the same as:

```
   i*N2*N3*N4*N5
 + j*N3*N4*N5
 + k*N4*N5
 + l*N5
 + m
```

Generalisation to Higher Dimensions

It is possible to generalise both of these concepts to arrays with greater than two dimensions. For higher-dimensional arrays, the ordering determines which dimensions of the array are more consecutive in memory. Any of the dimensions could be consecutive, just as a two-dimensional array could be listed column-first or row-first. The difference in offset between listings of that dimension would then be determined by a product of other dimensions. It is uncommon, however, to have any variation except ordering dimensions first to last or last to first. These two variations correspond to row-major and column-major, respectively.

More explicitly, consider a d-dimensional $N_1 \times N_2 \times .... \times N_d$ array with dimensions N_k (k=1...d). A given element of this array is specified by a tuple (n_1,n_2, ..., n_d) of d (zero-based indices $nk \in [0, N_k - 1]$.

In **row-major order**, the *last* dimension is contiguous, so that the memory-offset of this element is given by:

$n_d + N_d . (n_{d-1} + N_{d-1} . (n_{d-2} + N_{d-2} . (.... + N_2 n_1)...)) =$

$$\sum_{k=1}^{d} \left(\prod_{\ell=k+1}^{d} N_\ell \right) n_k$$

In **column-major order**, the *first* dimension is contiguous, so that the memory-offset of this element is given by:

$n_1 + N_1 . (n_2 + N_2 . (n_3 + N_3 . (.... + N_{d-1} n_d)...)) =$

$$\sum_{k=1}^{d} \left(\prod_{\ell=1}^{k-1} N_\ell \right) n_k$$

Note that the difference between row-major and column-major order is simply that the order of the dimensions is reversed. Equivalently, in row-major order the rightmost indices vary faster as one steps through consecutive memory locations, while in column-major order the leftmost indices vary faster.

Depending on the ordering method the elements are stored in the memory, we will get different positions of an element in the linear memory and consequently a different address for each method. To calculate the address we use the following procedure:

Step 1: Get the offset value of the element under consideration, make sure you use the correct forumula.

Step 2: Multiply the offset with the size of the element's datatype (Like int is of 4 bytes for 32-bit, Earlier compilers like TurboC worked on 16-bit platform and for them size of int is 2 bytes).

Step 3: Add this to the base address to get the final address.

Thus, address if a[i][j][k] element of a 3-D array a[N1][N2][N3]= a+ (i*N2*N3 + j*N3 +k)*sz
where sz is the memory needed for any element of array a.

■ **Example** A computer that uses 2 bytes for integer is used to store a 2-D array a. Address of a[3][2] and a[1][1] is observed to be 2028 and 2010 respectively. How many elements are there in a row? Rather how many columns are available in a? Assume language to be using row major order storage.

■ **Answer:** Assuming r and c as the rows and columns of a, we can write

Based address of a[3][2]=

Base address of a + (3*c+2)*2=2028

Base address of a[1][1]=a+(c+1)*2=2010

By solving the above equations, we get c as 4.

■ **Example** A computer that uses 2 bytes for integer is used to store a 2-D array a. Address of a[3][2] and a[1][1] is observed to be 2028 and 2010, respectively. How many elements are there in a column? Rather how many rows are available in a? It is reported that a program that is accessing this array sequentially is showing memory segment violation when 2040 address is accessed. Assume language to be using row major order storage. What will be the address of 4^{th} row 3^{rd} column element?

■ **Answer:** As usual, we calculate number of columns as above. That is, we get number of columns as 4. Now, we calculate base address of a from:
Base address of a[3][2]=a+(3*c+2)*2=2028
Base address of a[1][1]=a+(1*c+1)*2=2010.
By solving the above equations we get c as 4.
Therefore,
Base address of a= 2010-(4+1)*2=2000
As, we are getting segment violation at 2040, then address of last element=2038
Therefore, total number of elements in the array= (2040-2000)/2=20 (As integer takes 2 bytes)
Therefore, number of rows=20/4=5
Address of 4^{th} row 3^{rd} column element = 2000+ (4*4+3)*2 = 2038

■ **Example** We know addresses of two different elements of a 2-D array. Is it possible to find how many columns are there in it?

■ **Answer:** No. If they belong to same row, we cannot find number of columns otherwise, we can find.

■ **Example** How many elements addresses are needed to calculate geometry of a 3-D array that is physically organised in row major order?

■ **Answer :** We know that address of ith plane, jth row and kth column element can be given as:
Base address of the array+(i*N2*N3+j*N3+k)*number of bytes for each element.

Here, N1, N2 and N3 are dimensions of the 3-D array.

If you observe the above equation, we have three unknown's base address, N2 and N3. To find them, we need three equations, that is we need the addresses of three elements. Of course, we may see that base address will be common for all the three equations.

To find N3, we may need addresses of two elements that are in same plane. Because, base address, i*N2*N3 becomes common for both equations. Similarly, to find N2 we need addresses of two elements of two different planes. We can calculate N2 only after calculating N3. In order to calculate N1, we need address of either last element of the 3-D array or address of any element in last plane and last row.

Also, we can find base address only after calculating N2 and N3.

Another method of calculating offset of an element

Consider, we have a 5-D integer array A[N1][N2][N3][N4][N5]; We can view this five-dimension array as a single dimension array of arrays:

```
type
        OneD = array [N5] of int;
        TwoD = array [N4] of OneD;
        ThreeD = array [N3] of TwoD;
        FourD = array [N2] of ThreeD;
        A : array [N1] of FourD;
```

The size of `OneD` is N5. Since `TwoD` contains N4 `OneD` arrays, its size is N4N5 bytes. Likewise, `ThreeD` is N3 `TwoDs`, so it is N3N4N5 bytes long. Finally, `FourD` is N2 `ThreeDs`, so it is N2N3N4N5 bytes long. To compute the offset of "`A [b] [c] [d] [e] [f]`", we can use the following steps:

- Compute the offset of A [b] as " b * size". Here size is N2N3N4N5. Use this result as the Base in the next computation.

- Compute the offset of A [b] [c] by the formula "Base + c*size", where Base is the value obtained immediately above and size is N3N4N5. Use the result as the new Base in the next computation.
- Compute the offset of A [b] [c] [d] by "Base + d*size" with Base coming from the above computation and size being N4N5.
- Compute the offset of A [b] [c] [d] [e] with the formula "Base + e*size" with Base from above and size being N5. Use this value as the Base for the next computation.
- Finally, compute the offset of A [b] [c] [d] [e] [f] using the formula "Base + f" where Base comes from the above computation. Use this value as the Base for the next computation.
- To get the address of the desired element, base address of array + Base*number of bytes used for each element by the computer.

Suppose we have two arrays initialised as follows
A1 = {N2*N3*N4*N5, N3*N4*N5, N4*N5, N5, 1} and A2 = {b, c, d, e, f} then the following for loop can be used to find offset assuming base value is initially 0.

```
for(i=0;i<5;i++)
        base  += A1[i] * A2[i];
```

Note that this can be easily extend to any number of dimensions by simply initializing `A1` and `A2` appropriately and changing the ending values of the for loop.

■ **Example** Calculate address of the element a[3][2][1][2][3] in a 5-D array having each dimension as 4. Assume, computer uses 4 byte integers and base address of the array is 2000.

■ **Answer:** 2000 + (3*(4*4*4*4) + 2*(4*4*4) + 1*(4*4) + 2*4 + 3)* 4 = 5692

2.4.2 Searching

This problem can be stated as follows: An array of elements is given and we are required to find whether a given element x is available in the given array or not.

Sequential Search

Here, each element from the array is taken in one after another and compared with x and if it is same then it will be called as success. Even after traversing entire array if we can not find x then we call that element is not available in the array. The same can be written as:

```
int seqsearch( int a[], int n, int x){
     int I;
     for(I=0; I<n;I++) if( a[I]==x) return 1;
     return 0;
     }
```

By observing the above code we can understand that if first element of the array itself is x then we will be spending one comparison and if all the elements are not same as x then we will be spending n comparisons. Thus, best case complexity of this algorithm can be said as 1 and worst case complexity as n. Thus, average case complexity is n/2 comparisons. The same is represented in big-oh notation as O(n). Thus, this algorithm is also called as **linear search** or **serial search**.

Write a program in C language for linear search on a 2-D array

/* Assuming array is a of size mxn. X is the element we are looking for */

Solution 1 :

```
for(i=0;i<m;i++)
for(j=0;j<n;j++) if(a[i][j]==X) { printf("Found\n"); exit(-1); }
printf("Not found\n");
```

Solution 2 :

If we carry in using based address of the array.

```
for(i=0;i<(m*n);i++)
if( *(a+i) == X) {printf("Found\n"); exit(-1); }
printf("Not found\n");
```

Second one is preferable because of the following reasons:

Whenever we refer a[i][j], an implicit multiplication is used (a+i*columns+j) to locate the element.

Binary Search

Binary search is a simple and very fast way to access information in a **sorted array**. Binary search uses **divide-and-conquer** strategy and it is most naturally written as a recursive algorithm. Evidently, first we compare x with the middle element of the array. If it is same, then we have got the answer. Otherwise, we carry the search either in the left or right half of the array recursively. Here is code to search the array **a** for x using this approach.

```
int bsearch(int a[], int left, int right, int x){
/* Binary search function. Here, x is the value to be find in the array a,
left and right are the integer limits of the array to search and should always satisfy left ≤ right.
We assume that the array is having elements ascending order
*/
if (left == right)
if (a[left] == x ) val = left;
else val = '-1';
else
mid = (left + right)/2;//integer division
if (a[mid] < x)
val = bsearch(a, mid + 1, right,x);
else
val = bsearch(a,left, mid,x);
}
```

Binary search searches for an element in the array a which matches the "x" argument. If it finds such an element, it returns its index. If it cannot find the element, it returns **'–1'** indicating the given element is not seen in the array. Let us look at how it works.

First of all, **left** and **right** describe the limits of the subarray that we are searching in. If they are equal, then we are looking at a subarray that consists of a single element. Either the element **a[left]** is the element we are looking for, or there is no element matching the key.

In the general case where **left < right,** we are looking in an array with at least two elements. Binary search divides the array approximately in two by computing a midpoint, which is mid = (left + right)/2. Then it compares the element **a[mid]** with the key. Since the array is sorted, all the elements to the left of **a(mid)** are **<= a[mid]**, while all the elements to the right of **a[mid]** are **>= a[mid]**. So if x is larger than **a[mid]**, it must be to the right of **a[mid]**, if it is in the array at all. That is, it must be in the subarray from **a[mid+1]** to **a[right]**. The recursive call to **bsearch** looks for it there. If **x** is **<= a[mid]**, then it must be in the subarray from **a[left]** to **a[mid]**. The other recursive call looks for it there.

Iterative Implementation of Binary search

```
int bsearch(int a[], int left, int right, int x){
/* Binary search function
x is the value to find in the array a
left and right are the integer limits of the subarray to search and should always satisfy left ≤ right *
/int mid;
while (left<=right)
{
mid = (left + right)/2;
if(a[mid]==x) return mid;
else if (a[mid] < x) left=mid + 1;
else
right=mid-1;
}
```

```
return -1;
}
```

By observing the above code we can understand that if first time mid element of the array itself is x then we will be spending one comparison otherwise we will search in only either of the two halves. Thus, best case complexity of this algorithm can be said as 1, and worst case complexity as $\log_2(n)$. The same is represented in big-oh notation as $O(\log_2 n)$. The above equation illustrates how time complexity changes with number of elements. We find for large values of n also, we will be needing very small number of comparisions compared to serial search algorithm.

■ **Example** Which is costilier, linear or logarithmic?

■ **Answer:**

Look at the following table

n	Linear Search (n+1)/2	Binary Search: ½ $\log_2(n)$+1
8	4.5	2.5
16	8.5	3
32	16.5	3.5
1024	512.5	6
2048	1024.5	6.5
4096	2048.5	7
65536	32168.5	9.5

When we see the average costs of linear and binary search algorithms, we may find that the binary search takes very less number of operations for large values of n. Thus, it is efficient. Of course, a bog joke is: Is it meaningful to compare these two algorithms at all? In binary search, we assume that the data in order; where is in linear search, we donot assume any such thing. Thus, really it is a foolish thing to compare the both.

Fibnocci Search

The **Fibonacci search technique** is a method of searching a sorted array using a divide and conquer policy that narrows down possible locations with the aid of Fibnocci numbers. Compared to binary search, Fibonacci search has the property of examining locations whose addresses have lower dispersion. Therefore, when the elements being searched have non-uniform access memory storage (i.e., the time needed to access a storage location varies depending on the location previously accessed), the Fibonacci search has an advantage over binary search in slightly reducing the average time needed to access a storage location. The typical example of non-uniform access storage is that of a magentic tape, where the time to access a particular element is proportional to its distance from the element currently under the tape's head. Note, however, that large arrays not fitting in cache or even in RAM can also be considered as non-uniform access examples. Fibonacci search has a complexity of $O(\log(n))$.

Fibonacci search also assumed that the data is in sorted order; i.e., either in ascending order or in descending order.

Algorithm

Let *k* be defined as an element in *F*, the array of Fibonacci numbers. $n = Fm$ is the array size. If the array size is not a Fibonacci number, let *Fm* be the smallest number in *F* that is greater than *n*.

Let F_k represent the k-th Fibonacci number where $F_{k+2}=F_{k+1} + F_k$ for k>=0 and $F_0 = 0$, $F_1 = 1$. To test whether an item is in a list of $n = F_m$ ordered numbers, proceed as follows:

(a) Set k = m.

(b) If k = 0, finish – no match.

(c) Test item against entry in position F_{k-1}.

(d) If match, finish.

(e) If item is less than entry F_{k-1}, discard entries from positions $F_{k-1} + 1$ to n.
Set k = k – 1 and go to b).

(f) If item is greater than entry F_{k-1}, discard entries from positions 1 to F_{k-1}.
Renumber remaining entries from 1 to F_{k-2}, set k = k – 2 and go to b).

If F_m>n then the original array is augmented with F_{m-n} numbers larger than **val** and the above algorithm is applied.

```
int fibsearch(const int *arr, int n, int val){
 int j,mid,f1,f2,t;
/*  Precomputed Fibonacci numbers F0 up to F46. This implementation assumes that the size n
 *  of the input array fits in 4 bytes. Note that F46=1836311903 is the largest Fibonacci
 *  number that is less or equal to the 4-byte INT_MAX (=2147483647). This ensures correct operation
for n>F46.
 */
const static int Fib[47+1]={0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597,
2584, 4181, 6765,
               10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040, 1346269,
2178309, 3524578,
           5702887, 9227465, 14930352, 24157817, 39088169, 63245986, 102334155, 165580141, 267914296,
            433494437, 701408733, 1134903170, 1836311903};
for(j=1; Fib[j]<n; j++);
mid=n-Fib[j-2]+1;
f1=Fib[j-2];
f2=Fib[j-3];
while(val !=arr[mid])
{
          if(mid<0||val >arr[mid])
          {
                        if(f1==1) return  -1;
                        mid +=f2;
                        f1-=f2;
                        f2-=f1;
                        }
          else
          {
              if( f2==0) return -1;
              mid -=f2;
              t=f1-f2;
              f1=f2;
              f2=t;
          }
}
return(mid);
}
```

Interpolative Search

Consider our human actions while searching for a telephone number in a telephone directory. If the total number of pages is 500 and we are looking for the number of a person RANJIT. When we first opened the page 420, we found that it is having names of people with V. Then, next when we have opened page 350, we have found names starting with P. Thus, next we may open page number 387, assuming that the names are equally distributed. That is between P to V there exists 8 alphabets and names of this range are occupying 420-350=70 pages. Thus, names per alphabet is calculates as 9 (70/8). Thus, the names which starts with R is estimates as: 350+ 3 * 9= 387 (Approximately). Now, we may go to page 387. Depending

on what we found there, we may fine tune our search. This type of search is called interpolative search.

If we need to find items in a **sorted unbalanced array a**, we can use **interpolation search** using the formula:

$$\text{next} = \text{low} + \left\lfloor \frac{X - a[\text{low}]}{a[\text{high}] - a[\text{low}]} \times (\text{high} - \text{low} + 1) \right\rfloor$$

and using "next" instead of finding the mid-point each time. This is done on average with complexity **O(log log N)**

2.4.3 Sorting

Given an array with elements in any order, sorting is the process of moving the elements within the array to produce a sorted array (either in ascending or descending order) at the end. Sorting requires that the elements have an order. Ordering elements can make future search as easy.

Sorting algorithms are divided into two categories: **internal** and **external** sorts.

Internal Sort:

Any sort algorithm which uses main memory exclusively during the sort is referred as internal type. This assumes high-speed random access memory.

External Sort:

Any sort algorithm which uses external memory, such as tape or disk, during the sort, is referred as external type.

Note

Algorithms may read the initial values from magnetic tape or write sorted values to disk, but this is not using external memory during the sort. Note that even though virtual memory may mask the use of disk, sorting sets of data much larger than main memory may be much faster using an explicit external sort.

Sort Stable

A sort algorithm is said to be "stable" if multiple items which compare as equal will stay in the same order they were in after a sort.

Bubble Sort

The bubble sort is the oldest and simplest sort in use. The bubble sort compares each element with the element next to it, and swaps them if required. The algorithm repeats this process until it makes a pass all the way through the array without swapping any items (in other words, all items are in the correct order). This causes larger values to "bubble" to the end of the array while smaller values "sink" towards the beginning of the array. Under best-case conditions (the array is already sorted), the bubble sort can approach a constant $O(n)$ level of complexity. General-case complexity is an $O(n^2)$. While the insertion, selection, and shell sorts also have $O(n^2)$ complexities, they are significantly more efficient than the bubble sort. Realistically, there isn't a noticeable performance difference between the various sorts for 100 items or less, and the simplicity of the bubble sort makes it attractive. The bubble sort should not be used for repetitive sorts or sorts of more than a couple hundred items.

Pros: Simple and easy to implement.

Cons: Horribly inefficient.

Below is the basic bubble sort algorithm.

```
void bubbleSort(int numbers[], int array_size){
  int i, j, temp;
  for (i = (array_size - 1); i >= 0; i--)  {
    for (j = 1; j <= i; j++){
      if (numbers[j-1] > numbers[j]){
        temp = numbers[j-1];
        numbers[j-1] = numbers[j];
        numbers[j] = temp;
      }
    }
  }
}
```

Another version:

```
void bubblesort(int a[], int n){
  int  i, j, t;
for(i=0;i<n-1;i++){
 for(j=0; j<n-1-i; j++){
 if( a[j]<a[j+1]){
  t=a[j];  a[j]=a[j+1];  a[j+1]=t;
}
}
}
}
```

Analysis:

1'st time, inner loop runs for n-1 times thus n-1 comparisons are spent.

2'nd time, inner loop runs for n-2 time thus n-2 comparisons are spent.

……..

……..

Last time, inner loop runs for 1 time, thus 1 comparison is spent.

Thus, total number of comparisons are = 1+2+3+….+(n-2) + (n-1)= n(n-1)/2

Thus, time complexity of this algorithm = $O(n^2)$, i.e quadratic complexity.

Notice that if we make one complete pass through the inner loop of bubble sort without doing any swaps, then the array is already sorted.

Insertion Sort

The insertion sort inserts each item into its proper place in the final array. To save memory, most implementations use an in-place sort that works by moving the current item past the already sorted items and repeatedly swapping it with the preceding item until it is in place. Like the bubble sort, the insertion sort has a complexity of $O(n^2)$. Although it has the same complexity, the insertion sort is a little over twice as efficient as the bubble sort.

Pros: Relatively simple and easy to implement.

Cons: Inefficient for large lists.

The insertion sort is a good middle-of-the-road choice for sorting a few thousand items or less. The algorithm is significantly simpler than the shell sort, with only a small trade-off in efficiency. At the same time, the insertion sort is over twice as fast as the bubble sort and almost 40% faster than the selection sort. The insertion sort should not be used for sorting arrays larger than a couple thousand items or repetitive sorting of lists larger than a couple hundred items.

Below is the basic insertion sort algorithm.

```
void insertionSort(int numbers[], int array_size){
  int i, j, index;
  for (i=1; i < array_size; i++){
    index = numbers[i];
    j = i;
    while ((j > 0) && (numbers[j-1] > index))
    {
      numbers[j] = numbers[j-1];
      j = j - 1;
    }
    numbers[j] = index;
  }
}
```

Another Version of Insertion Sort

```
void insertionsort ( int a[], int n){
int i,j,k,m;
```

```
for(i=1; i<n; i++){
  m=a[i];
 for(j=0; j<i; j++){
   if(a[j] > m ){
      for(k=i-1; k>=j; k--) a[k+1]=a[k];
      a[j]=m;
    break;
  }
}
}
}
```

The above algorithm requires n–1 comparisons if the array contains elements in descending order. Whereas, if the array of elements are in ascending order the complexity becomes n(n–1)/2. Thus, the average case complexity can be also said as second order complexity, i.e $O(n^2)$.

Selection Sort

The selection sort works by selecting the smallest unsorted item (or largest) remaining in the array, and then swapping it with the item in the next position to be filled. The selection sort has a complexity of $O(n^2)$.

Pros: Simple and easy to implement.

Cons: Inefficient for large arrays, so similar to the more efficient insertion sort that the insertion sort should be used in its place.

The selection sort is the unwanted stepchild of the $O(n^2)$ sorts. It yields a 60% performance improvement over the bubble sort, but the insertion sort is over twice as fast as the bubble sort and is just as easy to implement as the selection sort. In short, there really isn't any reason to use the selection sort - use the insertion sort instead. If one really want to use the selection sort for some reason, it is wiser to avoid sorting arrays of more than a 1000 items with it or repetitively sorting arrays of more than a couple hundred items.

Below is the basic selection sort algorithm.

```
void selectionSort(int numbers[], int array_size){
  int i, j;  int min, temp;
  for (i = 0; i < array_size-1; i++){
    min = i;
    for (j = i+1; j < array_size; j++)
    {
      if (numbers[j] < numbers[min])   min = j;
    }
    temp = numbers[i];
    numbers[i] = numbers[min];
    numbers[min] = temp;
  }
}
```

If we observe the above function, we may find that inner loop runs for (n-1), (n-2)…..2,1 times. Thus, its complexity can be said as $O(n^2)$

■ **Example** Analyse the following recursive selection sort algorithm

```
void SelectionSort(int A[], int i, int n)        {
                int j, small, temp;
(1) if (i < n) {
(2)    small = i;
(3)    for (j = i + 1, j <= n, j++) {
```

```
(4)          if (A[j] < A[small])
                     small = j; }
(6)   temp = A[small];
(7)   A[small] = A[i];
(8)   A[i] = temp;
(9)   SelectionSort(A, i + 1, n);    }      }
```

In this function, n indicates the size of the array, i tells us how much of the array is left to sort, specifically A[i...n]. So, a call to **SelectionSort(A,1,n)** will sort the entire array through recursive calls. Do remember that we are assuming first element index is 1.

We now develop a recurrence relation. Note that the size of the array to be sorted on each recursive call is equal to n – i + 1. We will denote this value as m, the size of the array during any particular recursive call. There is one base case (m = 1). In this case, only line 1 is executed, taking some constant amount of time which we will call *a*. Note that m = 0 is not the base case because the recursion converges down to a list with one element; when i = n, m = 1.

The inductive case is for m > 1: this is when recursive calls are made. Lines 2, 6, 7, and 8 each take a constant amount of time. The **for** loop of lines 3, 4 and 5 will execute m times (where m = n – i + 1). So a recursive call to this function will be dominated by the time it takes to execute the **for** loop m times, which we shall designate O(m). The time for the recursive call of line 9 is T(m-1). So, the inductive definition of recursive SelectionSort is:

$$T(1) = a$$

$$T(m) = T(m\text{-}1) + O(m)$$

To solve this recurrence relation, we first get rid of the big-Oh expression by substituting the definition of big-Oh: (f(n) = O(g(n)) if f(n) <= C * g(n)), so we can substitute C*m for O(m):

$$T(1) = a$$

$$T(m) = T(m-1) + C*m$$

Now, we can either try repeated substitutions or just enumerate a few cases to look for a pattern. Let us try repeated substitution:

$$T(m) = T(m-1) + C*m$$

$= T(m-2) + 2Cm - C$ because $T(m-1) = T(m-2) + C(m-1)$

$= T(m-3) + 3Cm - 3C$ because $T(m-2) = T(m-3) + C(m-2)$

$= T(m-4) + 4Cm - 6C$ because $T(m-3) = T(m-4) + C(m-3)$

$= T(m-5) + 5Cm - 10C$ because $T(m-4) = T(m-5) + C(m-4)$

...

$= T(m-j) + jCm - (j(j-1)/2)C$

To get a closed form formula we let j = m – 1. We do this because our base case is T(1). If we were to continue the repeated substitution down to the last possible substitution, we want to stop at T(1) because T(0) is really not the base case that the recursion converges on. (Note that we have to do an inductive proof later to allow us to do this substitution, but for now):

$$T(m) = T(1) + (m-1)Cm - ((m-1)(m-2)/2)C$$

$$= a + m^2C - Cm - (m^2C - 3Cm + 2C)/2$$

$$= a + (2m^2C - 2Cm - m^2C + 3Cm - 2C)/2$$

$$= a + (m^2C + Cm - 2C)/2$$

So, finally we have a closed form formula $T(m) = a + (m^2C + Cm - 2C)/2$. The complexity of this formula is $O(m^2)$, which is the same complexity as iterative selection sort, so doing it recursively did not save us any time.

Shell Sort

Invented by Donald Shell in 1959, the shell sort is the most efficient of the $O(n^2)$ class of sorting algorithms. Of course, incidentally shell sort is also the most complex of the $O(n^2)$ algorithms. The shell sort is also referred to "diminishing increment sort", or "comb sort". The algorithm makes multiple passes through the array, and each time sorts a number of

equally sized sets using the insertion sort. The size of the set to be sorted gets larger with each pass through the list, until the set consists of the entire list. (Note that as the size of the set increases, the number of sets to be sorted decreases.) This sets the insertion sort up for an almost-best case run each iteration with a complexity that approaches $O(n)$.

The items contained in each set are not contiguous - rather, if there are i sets then a set is composed of every i-th element. For example, if there are 3 sets then the first set would contain the elements located at positions 1, 4, 7 and so on. The second set would contain the elements located at positions 2, 5, 8, and so on; while the third set would contain the items located at positions 3, 6, 9, and so on.

The size of the sets used for each iteration has a major impact on the efficiency of the sort. Several Heroes Of Computer Science, including Donald Knuth and Robert Sedgewick, have come up with more complicated versions of the shell sort that improve efficiency by carefully calculating the best sized sets to use for a given array.

Pros: Efficient for medium-size lists.

Cons: Somewhat complex algorithm, not nearly as efficient as the merge, heap, and quick sorts.

The shell sort is by far the fastest of the O(n^2) class of sorting algorithms. It's more than 5 times faster than the bubble sort and a little over twice as fast as the insertion sort, its closest competitor. The shell sort is still significantly slower than the merge, heap and quick sorts, but its relatively simple algorithm makes it a good choice for sorting lists of less than 5000 items unless speed is hyper-critical. It's also an excellent choice for repetitive sorting of smaller array.

Below is the basic shell sort algorithm.

```
void shellSort(int numbers[], int array_size){
  int i, j, increment, temp;
  increment = 3;
  while (increment > 0){
    for (i=0; i < array_size; i++)
    {
      j = i;
      temp = numbers[i];
while((j >=increment) && (numbers[j-increment] > temp))
      {
        numbers[j] = numbers[j - increment];
        j = j - increment;
      }
      numbers[j] = temp;
    }
    if (increment/2 != 0)
      increment = increment/2;
  else if (increment == 1)
      increment = 0;
    else
      increment = 1;
  }
}
```

Heap Sort

The heap sort is the slowest of the $O(n \log n)$ sorting algorithms, but unlike the merge and quick sorts it doesn't require massive recursion or multiple arrays to work. This makes it the most attractive option for *very* large data sets of millions of items.

The heap sort works as it name suggests – it begins by building a heap out of the data set, and then removing the largest item and placing it at the end of the sorted array. After removing the largest item, it reconstructs the heap and removes the

largest remaining item and places it in the next open position from the end of the sorted array. This is repeated until there are no items left in the heap and the sorted array is full. Elementary implementations require two arrays – one to hold the heap and the other to hold the sorted elements.

To do an in-place sort and save the space the second array would require, the algorithm below "cheats" by using the same array to store both the heap and the sorted array. Whenever an item is removed from the heap, it frees up a space at the end of the array that the removed item can be placed in.

Pros: In-place and non-recursive, making it a good choice for extremely large data sets.
Cons: Slower than the merge and quick sorts.

As mentioned above, the heap sort is slower than the merge and quick sorts but does not use multiple arrays or massive recursion like they do. This makes it a good choice for really large sets, but most modern computers have enough memory and processing power to handle the faster sorts unless over a million items are being sorted.

Below is the basic heap sort algorithm. The siftDown() function builds and reconstructs the heap.

```
void heapSort(int numbers[], int array_size){
  int i, temp;
  for (i = (array_size / 2)-1; i >= 0; i--)
    siftDown(numbers, i, array_size);
  for (i = array_size-1; i >= 1; i--){
    temp = numbers[0];
    numbers[0] = numbers[i];
    numbers[i] = temp;
    siftDown(numbers, 0, i-1);
  }
}
void siftDown(int numbers[], int root, int bottom){
  int done, maxChild, temp;
  done = 0;
 while ((root*2 <= bottom) && (!done)){
    if (root*2 == bottom)
      maxChild = root * 2;
    else if (numbers[root * 2] > numbers[root * 2 + 1])
      maxChild = root * 2;
    else
      maxChild = root * 2 + 1;
    if (numbers[root] < numbers[maxChild]) {
      temp = numbers[root];
      numbers[root] = numbers[maxChild];
      numbers[maxChild] = temp;
      root = maxChild;
    }
    else
      done = 1;
  }
}
```

Merge Sort

The merge sort splits the array to be sorted into two equal halves, and places them in separate arrays. Each array is recursively sorted, and then merged back together to form the final sorted list. Like most recursive sorts, the merge sort has an algorithmic complexity of $O(n \log n)$.

Elementary implementations of the merge sort make use of three arrays - one for each half of the data set and one to store the sorted list in. The below algorithm merges the arrays in-place, so only two arrays are required. There are non-recursive versions of the merge sort, but they does not yield any significant performance enhancement over the recursive algorithm on most machines.

Pros: Marginally faster than the heap sort for larger sets

Cons: At least twice the memory requirements of the other sorts; recursive.

The merge sort is slightly faster than the heap sort for larger sets, but it requires twice the memory of the heap sort because of the second array. This additional memory requirement makes it unattractive for most purposes - the quick sort is a better choice most of the time and the heap sort is a better choice for very large sets.

Like the quick sort, the merge sort is recursive which can make it a bad choice for applications that run on machines with limited memory.

Below is the basic merge sort algorithm.

```
void mergeSort(int numbers[], int temp[], int array_size){
  m_sort(numbers, temp, 0, array_size - 1);
}
void m_sort(int numbers[], int temp[], int left, int right){
  int mid;
  if (right > left)
  {
    mid = (right + left) / 2;
    m_sort(numbers, temp, left, mid);
    m_sort(numbers, temp, mid+1, right);
    merge(numbers, temp, left, mid+1, right);
  }
}
void merge(int numbers[], int temp[], int left, int mid, int right){
  int i, left_end, num_elements, tmp_pos;
  left_end = mid - 1;
  tmp_pos = left;
  num_elements = right - left + 1;
  while ((left <= left_end) && (mid <= right)){
    if (numbers[left] <= numbers[mid]){
      temp[tmp_pos] = numbers[left];
      tmp_pos = tmp_pos + 1;
      left = left +1;
    }
    else{
      temp[tmp_pos] = numbers[mid];
      tmp_pos = tmp_pos + 1;
      mid = mid + 1;
    }
  }
  while (left <= left_end){
    temp[tmp_pos] = numbers[left];
    left = left + 1;
    tmp_pos = tmp_pos + 1;
```

```
  }
  while (mid <= right){
    temp[tmp_pos] = numbers[mid];
    mid = mid + 1;
    tmp_pos = tmp_pos + 1;
  }
  for (i=0; i <= num_elements; i++){
    numbers[right] = temp[right];
    right = right - 1;
  }
}
```

Quick Sort

The quick sort is an in-place, divide-and-conquer, massively recursive sort. As a normal person would say, it's essentially a faster in-place version of the merge sort. The quick sort algorithm is simple in theory, but very difficult to put into code (computer scientists tied themselves into knots for years trying to write a practical implementation of the algorithm, and it still has that effect on university students). The recursive algorithm consists of four steps (which closely resemble the merge sort):

1. If there are one or less elements in the array to be sorted, return immediately.
2. Pick an element in the array to serve as a "pivot" point. (Usually the left-most element in the array is used.)
3. Split the array into two parts – one with elements larger than the pivot and the other with elements smaller than the pivot.
4. Recursively repeat the algorithm for both halves of the original array.

The efficiency of the algorithm is majorly impacted by which element is choosen as the pivot point. The worst-case efficiency of the quick sort, $O(n^2)$, occurs when the list is sorted and the left-most element is chosen. Randomly choosing a pivot point rather than using the left-most element is recommended if the data to be sorted isn't random. As long as the pivot point is chosen randomly, the quick sort has an algorithmic complexity of $O(n \log n)$.

Pros: Extremely fast.
Cons: Very complex algorithm, massively recursive.

The quick sort is by far the fastest of the common sorting algorithms. It is possible to write a special-purpose sorting algorithm that can beat the quick sort for some data sets, but for general-case sorting there isn't anything faster.

As soon as students figure this out, their immediate impulse is to use the quick sort for everything – after all, faster is better, right? It's important to resist this urge – the quick sort isn't always the best choice. As mentioned earlier, it's massively recursive (which means that for very large sorts, you can run the system out of stack space pretty easily). It's also a complex algorithm – a little too complex to make it practical for a one-time sort of 25 items, for example.

With that said, in most cases the quick sort is the best choice if speed is important (and it almost always is). Use it for repetitive sorting, sorting of medium to large lists, and as a default choice when you're not really sure which sorting algorithm to use. Ironically, the quick sort has horrible efficiency when operating on lists that are mostly sorted in either forward or reverse order – avoid it in those situations.

Below is the basic quick sort algorithm.

```
void quickSort(int numbers[], int array_size){
  q_sort(numbers, 0, array_size - 1);
}
void q_sort(int numbers[], int left, int right){
  int pivot, l_hold, r_hold;
  l_hold = left;
  r_hold = right;
  pivot = numbers[left];
  while (left < right){
```

```
    while ((numbers[right] >= pivot) && (left < right))
      right--;
    if (left != right){
      numbers[left] = numbers[right];
      left++;
    }
    while ((numbers[left] <= pivot) && (left < right))
      left++;
    if (left != right){
      numbers[right] = numbers[left];
      right--;
    }
  }
  numbers[left] = pivot;
  pivot = left;
  left = l_hold;
  right = r_hold;
  if (left < pivot)
    q_sort(numbers, left, pivot-1);
  if (right > pivot)
    q_sort(numbers, pivot+1, right);
}
```

When quick sort shows best case behaviour and when it shows worst case behaviour?

Each time pivot element becomes median then it shows best case behavior of O(nlogn) whereas it shows worst case behavior ($O(n^2)$) if pivot element is maximum of the partition.

Address calculation sort

This method is also known as sorting by address calculation or sorting by hashing. In this method a function f is applied to each element (key) which determines into which of the several subfiles the key to be placed. This function has to have order preserving property; i.e, if x, y are two keys where x<=y then f(x) <= f(y). Thus all the keys in a subfile will have their values less than or equal to the keys of next subfile. Items in a subfile can be sorted using any other sorting technique.

Let the data is 24, 44, 22, 34, 56, 99, 78 and 38 and function is value of first digit then the possible subfiles are

22, 24
34, 38
44
56
78
99

If the number of elements are n, subfiles are m and n/m is 1 then time complexity of this algorithm is O(n). Where as n/m is much larger and all the elements are uniformly distributed over all the subfiles then the time complexity of this algorithm still $O(n^2)$.

Radix Sorting

Radix sorting is another efficient, linear time sorting algrothm. It works by sorting data in pieces called digit, one digit at a time, starting from the least significant digit.

For example if the data are 15, 12, 49, 16, 36, 40 if we sort based on 1'st digit value then the list becomes 40, 12, 16, 36, 49. Then if we sort based on second digit the numbers becomes 12, 16, 36, 40, 49.

Here, a single loop governs the digit position on which we are currently sorting and apply shuffling of elements based on this digit position. This is repeated till shuffling is completed and till the most significant digit position. This works well for integers. However, the same can be applied on different data items also, by considering a group of bits of the data items at a time.

■ **Example** Given a set S of n elements and an index k ($1 \le k \le n$), we define the k-smallest element to be the k-th element when the elements are sorted from the smallest to the largest. Suggest an O(n) expected time algorithm for finding the k-smallest element in the mean case. For Example, if the given set of numbers: {6, 3, 2, 4, 1, 1, 2, 6,}. The 4-smallest element is 2 since in the 2 is the 4'th element in the sorted set {1, 1, 2, 2, 3, 4, 6, 6}.

■ **Answer:** The algorithm can be framed based on the Quick-Sort algorithm. We take a random number x and divide the array into three sets; those less than x (call set L), equal to x (call set E) and greater than x(call set G). If k is less than or equally to number of elements of L then we can search of kth largest in L recursively. Otherwise if k is less than or equal to number of elements of both the sets L and E, we return x as the kth largest. Otherwise, we carry search for kth largest in G recursively. This algorithm is given below.

```
Select(k, S)
   pick x in S
   partition S into:
      L < x
      E = x
      G > x
   if k ≤ length(L)
      return Select(k, L)
   else if k ≤ length(L) + length(E)
          return x
     else
      return Select(k - length(L) - length(E), G)
```

This algorithm shows worst case behavior if all the elements are larger than x (That is L is empty) and E contains only one x. That is, we have to search for kth largest in G which contains n-1 elements. If this happens for each of the recursive calls then this algorithm will be showing worst case behavior. In this situation it takes: n + (n–1) + (n–2) + ... + 1. Thus, worst case complexity can be said as: $O(n^2)$.

In the mean case, similar to quick-sort, half of the elements in L are good pivots, for which the size of L and G are each less than $\frac{3n}{4}$. Therefore, time complexity can be represented as given below; also from the Masters theorem, we can find the complexity in Big-oh notation as: O(n).

$$T(n) \le T\left(\frac{3n}{4}\right) + O(n) = O(n)$$

Example: Given an array of n numbers, suggest an O(n) expected time algorithm to determine whether there is a number in A that appears more than n/2 times.

Answer: If x is a number that appears more than $\frac{n}{2}$ times in A, than x is the $\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right)$-smallest in A.

Therefore, the algorithm is:

```
x ← Select ( ⌊n/2⌋ + 1 ), A)
NoAppearncess ← 0
for  i ← 1 to n do:
```

```
        if (A[i] = x) NoAppearances ++
    if  NoAppearances > n/2
          return TRUE
    else return FALSE
```

In the mean case, the above *Select* algorithm runs in O(n), computing NoA*ppearances* takes O(n) as well.
Total run time in the mean case: O(n)

■ **Example** Given an array A of M+N elements, where the first N elements in A are sorted and the last M elements in A are unsorted evaluate the run-time complexity in terms of M and N in the worst case, of fully sorting the array using insertion sort on A?

a. For each of the following cases, which sort method (or methods combination) would you use and what would be the run-time complexity in the worst case?

1. M = O(1)
2. M = O(logN)
3. M = O(N)

■ **Answer:** Assuming that the last M elements will be inserted to their right place using insertion sort; this requires N, N+1, N+2,...,N+M shifts. Thus, complexity can be said as O(M(M+N)). Complexity of $O(M^2 + N)$ is possible if we apply insertion sort to the last M elements and then merge them with first N elements.

1. Insertion-Sort in O(N)
2. Merge-Sort on the M elements in O(MlogM) and then Merge with the first N elements in O(N+M). Total: O(N)
3. Every sort method based on comparisons, when its running time is O(nlogn), such as Heap-Sort/Merge-Sort, is suitable for this case, and it's running time is O(NlogN). Quick-Sort is bad for this case, as its worst case analysis is $O(n^2)$.

■ **Example** Given the following algorithm to sort an array A of size n:

1. Sort recursively the first 2/3 of A (A[1..2n/3])
2. Sort recursively the last 2/3 of A (A[n/3+1..n])
3. Sort recursively the first 2/3 of A (A[1..2n/3])

Prove the above algorithm really sorts A and find a reccurence T(n), expressing it's running time.

■ **Answer** The basic assumption is that after the first 2 steps, the n/3 largest number are in their places, sorted in the last third of A. In the last stage the algorithm sorts the left 2 thirds of A.

$$T(n) = 3T(2n/3) = 3(3T(4n/9))$$

$$= 3(3(3T(8n/27))) = \ldots = 3^iT((2/3)^in)$$

after $i = \log_{3/2}n$ steps ... $T(n) = 3^{\log_{3/2}n}\cdot T(1)$

$$= 3^{(\log_3 n)/(\log_3(3/2))} = (3^{\log_3 n})^{1/(\log_3(3/2))}$$

$$= n^{1/(\log_3(3/2))} = n^{(\log_3 3)/(\log_3(3/2))} = n^{\log_{3/2}3}$$

$T(n) = O(n^{\log_{3/2}3})$, (using the Master-Theorem)

■ **Example** n records are stored in an array A of size n. Suggest an algorithm to sort the records in O(n) (time and space) in each of the following cases:

1. All the keys are 0 or 1
2. All the keys are in the range [0..k], k is constant

■ **Answer:**

1. Use the partition method (Quick-Sort) with pivot 0
2. First, partition method on A[1..n] with pivot 0, this way all the records with key 0 will be on the first x_0 indexes of the array. Second, partition method on A[x_0+1..n] with pivot 1 ... After k steps A is sorted

■ **Example** Assume that we are given as input n pairs of items, where the first item is a number and the second item is one of three colours (red, blue, or yellow). Further, assume that the items are sorted by number. Give an O(n) algorithm to sort the items by colour (all reds before all blues before all yellows) such that the numbers for identical colours stay sorted. For example: (1, blue), (3, red), (4, blue), (6, yellow), (9, red) should become (3, red), (9, red), (1, blue), (4, blue), (6, yellow).

■ **Answer:** First scan the pairs and write those pairs with red color then scan the pairs and write pairs with blue color and then with yellow. That is, we need three for loops to scan through all the n items. Thus, number of operations can be said as: 3n. Thus, complexity can be said as O(n).

■ **Example** Consider a sequence of $2n$ real numbers. Design an O($n$ln(n)) algorithm that partitions the numbers into n pairs, with the property that the partition minimises the maximum sum of a pair. For example, say we are given the numbers (1,3,5,9). The possible partitions are ((1,3),(5,9)), ((1,5),(3,9)), and ((1,9),(3,5)). The pair sums for these partitions are (4,14), (6,12), and (10,8). Thus the third partition has 10 as its maximum sum, which is the minimum over the three partitions.

■ **Answer:** Sort the numbers using any O($n$ln(n)) algorithm
Now, consider
first pair consisting of first and last numbers
second pair consisting of second and second last numbers and so on.
Then find out the minimum of them.

■ **Example** What is the maximum and minimum number of times that the largest element could be moved during the execution of Quicksort? Explain your answer with an example.

Max: n –1 An example input is ascending data like [5,4,3,2,1]
Min: 0 An example input is descending data like [1,2,3,4,5]

■ **Example** Consider a set S of $n \geq 2$ distinct numbers given in unsorted order. Give an algorithm to determine two distinct numbers x and y in the set S that satisfy a stated condition. In as few words as possible, describe your algorithm and justify its running time.

a. In O(n) time, determine x, y in S such that $|x - y| \geq |w - z|$ for all w, z in S.
b. In O($n$lg(n)) time, determine x, y in S such that $x \neq y$ and $|x - y| \leq |w - z|$ for all w, z in S such that $w \neq z$.

■ **Answer**:
(a) x, y are the two farthest apart. Find min, max, each in O(n) time.
(b) x, y are the two closest together. First sort with O($n$lg$n$) algo, then scan the sorted numbers for the two adjacent numbers with the minimum difference in O(n).

■ **Example** The mode of a set of numbers is the number that occurs most frequently in the set. The set (4, 6, 2, 4, 3, 1) has a mode of 4.

a. Give an efficient and correct algorithm to compute the mode of a set of n numbers.
b. Suppose we know that there is an (unknown) element that occurs $n/2+1$ times in the set. Give a worst-case linear-time algorithm to find the mode.

■ **Answer:**
(a) Sort the numbers using O(nlgn) algorithm.

Scan the sorted array and track the most frequent occurring number.

(b) Divide recursive the set by doing $\lfloor n/2 \rfloor$ pairwise comparisions and only keeping one representative of the pairs which are equal.

For exmple (1, 1, 1, 1, 2, 2) Divide the set in 3 pairs as follows:

1,1 equal take 1
1,1 equal take 1
2,2 equal take 2
Now divide (1,1,2) into the following pairs

1,1 take 1
2 discard
1 is the answer.
Consider another permutation of the above problem: (1,2,1,1,2,1)
1,2
1,1 take 1
2,1
1 is the answer.
Consider another permutation of the above problem: (1,1,2,2,1,1)
1,1 take 1
2,2 take 2
1,1 take 1
Now divide (1,2,1) into the following pairs
1,2 discard
1
1 is the answer.

■ **Example** Write a function to determine (in logarithmic time) the range of elements in a strictly increasing sequence $a_0, a_1, a_2, \ldots, a_{n-1}$ that have $a_i = i$.

```
void binRange(int a[], int n){
    // Binary search for some element with a[i]==i
    int low, high, mid, high1, high2, mid1, mid2, low1, low2;
    low=0;
    high=n-1;
    while (1)
    {
      if (high<low)
      {
        printf("Range is empty, bracketed by %d %d\n",high,low); return;
      }
      mid=(low+high)/2;
      if (a[mid]==mid)
        break;
      if (a[mid]<mid)
        low=mid+1;
      else
        high=mid-1;
    }
    // Find beginning of range
    low1=low;
    high1=mid;
    while (low1<=high1)
    {
      mid1=(low1+high1)/2;
      if (a[mid1]==mid1)
        high1=mid1-1;
      else
        low1=mid1+1;
```

```
}
printf("%d begins the range\n",low1);
// Find end of range
low2=mid;
high2=high;
while (low2<=high2)
{
  mid2=(low2+high2)/2;
  if (a[mid2]==mid2)
    low2=mid2+1;
  else
    high2=mid2-1;
}
printf("%d ends the range\n",high2);
}
```

■ **Example** Two players *A* and *B* are playing a guessing game (akin to Unix craps game) where *B* first thinks up an integer *X* (positive, negative or zero, and could be of arbitrarily large magnitude) and *A* tries to guess it. In response to *A*'s guess, *B* gives exactly one of the following three replies:

(a) Your guess is high
(b) Your guess is low
(c) Congrats. You won

Design an efficient algorithm that minimises A's guesses. Rather, defind A's strategy such that he takes minimal steps.

■ **Answer:** First find out whether *X* is positive or negative by guessing if *X* is ZERO. Say from the response you make out that it is positive. Then make the next guess "1". As long as you keep getting the response of "your guess is low", keep on doubling your guess until you finally get the response "your guess is high" for some guess *P*. Now you know that the number must lie between *P* and *P*/2, so do a binary search between them. The complexity of this approach will be $O(\log X)$.

■ **Example** In Internet protocols, MTU (Maximum Transfer Unit) defines the size of a packet that we can send to a target via a specific path without the packet getting fragmented into smaller packets. Let us assume that we do not know what is appropriate MTU size along a path. If our packet size is bigger than the MTU, we will get a message that our packet would be fragmented. We propose to find the acceptable MTU size (X) along a specific path by repeatedly sending packets of increasing size until we get the fragmentation message. Suppose we have to determine *X*, which is the size of MTU along some path, and X could be potentially huge. Give an efficient algorithm to determine X and give its complexity in terms of *X*. (Remember, we do not know the value of *X* beforehand).

■ **Answer:** First try a packet of size 1 bye. If we do not get any fragmentation message, then double our packet size and keep doing it till we get the message that packet will be fragmented (say for a packet size of *P*). Now, we know that X must lie between P and P/2 (the previous try for which we did not get any error message), so do a binary search. The overall complexity is O(log X).

■ **Example** An integer array a contains n students grades (distinct) in unsorted way. A student *X* has been told that his rank in the class is *R* (*R* is an integer and obviously, $1 <= R <= n$). We want to find out the *k* boys who are ranked closest to A ($k/2$ students below A, and $k/2$ students above A). Devise an efficient algorithm to identify the scores of these *k* boys.

■ **Answer:** Since X's rank is *R*, the *k* boys' rank must vary between R-k and R+k. First find the $(R–k)^{th}$ and $(R+k)^{th}$ ranked element using Selection (linear time). Then do Partition twice to discard all the elements that fall out of the range ((R-k), (R+k)). It is linear.

■ **Example** In a social gathering, there are *b* boys and *g* girls (b > g) of different ages. We have two unsorted arrays giving their ages (one for the boys, the other for the girls). Devise an efficient O(b log g) algorithm to find out the ages that are common between both the boys and girls.

e.g.,
If $Array_{boy}$ = {10, 20, 17, 30, 23, 21} and $Array_{girl}$ = {12, 30, 17, 20},
Then $Array_{common}$ = {17, 20,30}

■ **Answer:** Sort the smaller (girls') array, then for each element in the larger array (boys), do a binary search on the girls' array. If found in girls array store in $Array_{common}$. Print $Array_{common}$ at the end.

■ **Example** PARTS-SWAP: The input is an array of n elements and an integer k (where $1 \le k < n$).

The input array can be implicitly thought of having the following two parts:

- Part *A*: Elements with array index 0 through *k-1*, and
- Part *B*: Elements with array index *k* through *n-1*.

We need to swap these 2 parts, so that the array looks like Part *B* followed by Part *A*. Observe that the individual order of elements inside these two parts do not change after this swap.

For example (n = 10, k=3) will convert the following array

0	1	2	3	4	5	6	7	8	9
10	3	2	5	6	1	2	4	1	3

into

0	1	2	3	4	5	6	7	8	9
5	6	1	2	4	1	3	10	3	2

(The top row in both the tables is the header row, showing the array indices)
Design an efficient algorithm for doing the PART-SWAP *using at most O(1) extra space*. Justify its correctness and complexity. It is not required that your algorithm uses divide and conquer approach.

```
int main(){
    int a[10], n, i,j,k,l,t;
    scanf("%d", &n);
    for(i=0;i<n;i++) scanf("%d", &a[i]);

    printf("Enter k value"); scanf("%d", &k);
     if(k<=n/2){
      for(i=0,l=n-k;i<k;i++,l++){
              t=a[l];a[l]=a[i];a[i]=t;
    }

    l=n-2*k;
    while(l--){
    t=a[n-k-1];
              for(i=n-k-2;i>=0;i--)a[i+1]=a[i];
           a[0]=t;
    }
   }
   else{
    k=n-k;
    for(i=0,l=n-k;i<k;i++,l++){
              t=a[l];a[l]=a[i];a[i]=t;
```

```
        }
        l=k;
        while(l--){
                  t=a[n-1];
                  for(i=n-2;i>=k;i--)a[i+1]=a[i];
               a[k]=t;
        }
       }
        for(i=0;i<n;i++)printf("%d ", a[i]);

        system("PAUSE");
        return 0;
    }
```

■ **Example** Assume that we have two separate databases having n and m numerical values(or keys). Thus, there exists $n + m$ values in total and we may assume that no two values are same. Assume that both n and m are in the form $2i - 1$, where i is an integer. We would like to determine the median of this set of $n + m$ values, which we will define here to be the $(n+m)/2$ *th* smallest value. However, the only way we can access these values is through *queries* to the databases. In a single query, we can specify a value k to one of the two databases, and the chosen database will return the k^{th} smallest value that it contains. Give an efficient algorithm that finds the median assuming that each query to any of the databases requires constant time. Also, derive the asymptotic complexity of your algorithm.

■ **Answer:** Follow the following method repetitively. Compare the individual medians of the two databases by querying them. Whichever is greater, discard the elements greater than it from your consideration. Similarly, whichever is smaller, discard the elements smaller than it from your consideration. Thus, at every stage we are pruning half of the current input size. So the overall complexity is O(log (n+m)).

■ **Example** Consider a list of size n, which has atmost log n distinct numbers, the rest being repetitions of those numbers. Design an algorithm to sort this list in O(n log log n) time.

■ **Answer:** First partition the list into n log n lists of size n/log n and use a mergesort-like strategy.

■ **Example** Assume that you were given an array of n distinct integers that was originally sorted (either ascending or descending) but has been circularly shifted by an amount k (k is an integer and 0<= k< n). Like, some examples of the input array for n = 5 could be:
(1,2,3,4,5) k=0 (5,4,3,2,1) k=0 (4,5,1,2,3) k=2 (1,5,4,3,2) k=1.
Give an efficient algorithm to find the value of k in linear time.

■ **Answer:** Consider ascending data and its shifted versions.
1 2 3 4 5
2 3 4 5 1
3 4 5 1 2
4 5 1 2 3
If we try to traverse the given and compare ith element is less than i+1th element, this will be failing exactly with one element in the shifted data. At which element it is failing will be tracked down to find amount of shifting k.
Similarly, consider descending data
5 4 3 2 1
4 3 2 1 5
3 2 1 5 4
2 1 5 4 3
With this data also, if we try to traverse the array and compare ith element is greater than i+1th element, it will be failing exactly at one element whose index can be used to find the amount of shifting k. The following code fragment does the above tasks.

```
int main(){
    int x[10], n, F1=0, F2=0, I1, I2,i;
    scanf("%d", &n);
    for(i=0;i<n;i++) scanf("%d", &x[i]); // shifted data
    for(i=0;i<n-1;i++){
                    if(x[i]<x[i+1]) { F1++; I1=i;}
                      else
                      { F2++; I2=i; }
                     }
    if(F1==1)printf("k=%d\n", n-I1-1);
    else if(F2==1)
    printf("k=%d \n", n-I2-1);
    system("PAUSE");
    return 0;
}
```

■ **Example** An array A[1 . . n] is unimodal if it consists of an increasing sequence followed by a decreasing sequence, or more precisely, if there is an index m ∈ {1, 2, . . . , n} such that

- A[i] < A[i + 1] for all 1 ≤ i < m, and
- A[i] > A[i + 1] for all m ≤ i < n.

In particular, A[m] is the maximum element, and it is the unique "locally maximum" element surrounded by smaller elements (A[m − 1] and A[m + 1]). Give an algorithm to compute the maximum element of a unimodal input array A[1 . . n] in O(lg n) time. Prove the correctness of your algorithm, and prove the bound on its running time.

■ **Answer:** Notice that by the definition of unimodal arrays, for each 1 ≤ i < n either A[i] < A[i + 1] or A[i] > A[i + 1]. The main idea is to distinguish these two cases:

1. By the definition of unimodal arrays, if A[i] < A[i + 1], then the maximum element of A[1..n] occurs in A[i + 1..n].
2. In a similar way, if A[i] > A[i +1], then the maximum element of A[1..n] occurs in A[1..i].

This leads to the following divide and conquer solution (note its resemblance to binary search):

```
1 a, b ← 1, n
2   while a < b
3   do mid ← ⌊(a + b)/2⌋
4      if A[mid] < A[mid + 1]
5             then a ← mid + 1
6      if A[mid] > A[mid + 1]
7             then b ← mid
8 return A[a]
```

The precondition is that we are given a unimodal array A[1..n]. The postcondition is that A[a] is the maximum element of A[1..n]. For the loop we propose the invariant "The maximum element of A[1..n] is in A[a..b] and a ≤ b".

When the loop completes, a ≥ b (since the loop condition failed) and a ≤ b (by the loop invariant). Therefore, a = b, and by the first part of the loop invariant the maximum element of A[1..n] is equal to A[a].

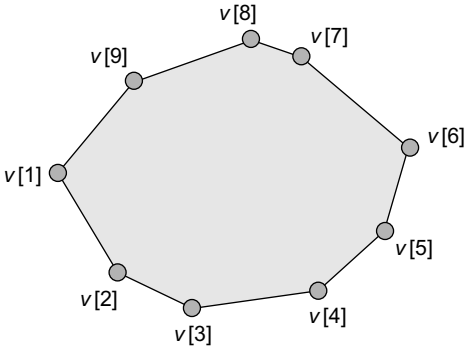
We use induction to prove the correctness of the invariant. Initially, a = 1 and b = n, so, the invariant trivially holds. Suppose that the invariant holds at the start of the loop. Then, we know that the maximum element of A[1..n] is in A[a..b]. Notice that A[a..b] is unimodal as well. If A[mid] < A[mid + 1], then the maximum element of A[a..b] occurs

in A[mid+1..b] by case 1. Hence, after a ← mid+1 and b remains unchanged in line 4, the maximum element is again in A[a..b]. The other case is symmetric.

To complete the proof, we need to show that the second part of the invariant a ≤ b is also true. At the start of the loop a < b. Therefore, a ≤ ⌊(a + b)/2⌋ < b. This means that a ≤ mid < b such that after line 4 or line 5 in which a and b get updated a ≤ b holds once more.

Thus, this divide and conquer approach leads to a running time of $T(n) = T(n/2) + \Theta(1) = \Theta(\lg n)$.

■ **Example** A polygon is ***convex*** if all of its internal angles are less than 180° (and none of the edges cross each other). Given figure shows an example. We represent a convex polygon as an array V [1 . . n] where each element of the array represents a vertex of the polygon in the form of a coordinate pair (x, y). We are told that V [1] is the vertex with the minimum x coordinate and that the vertices V [1 . . n] are ordered counterclockwise, as in the figure. You may also assume that the x coordinates of the vertices are all distinct, as are the y coordinates of the vertices.



An example of a convex polygon represented by the array V [1 . . 9]. V [1] is the vertex with the minimum x-coordinate, and V [1 . . 9] are ordered counterclockwise.

(a) Give an algorithm to find the vertex with the maximum x coordinate in O(lg n) time.

■ **Answer:** Notice that the x-coordinates of the vertices form a unimodal array and we can use the above algorithm for unimodal maximum to find the vertex with the maximum x-coordinate in O(lg n) time.

(b) Give an algorithm to find the vertex with the maximum y coordinate in O(lg n) time.

■ **Answer:** After finding the vertex V [max] with the maximum x-coordinate, notice that the y-coordinates in V [max], V [max + 1], . . . , V [n − 1], V [n], V [1] form a unimodal array and the maximum y-coordinate of V [1..n] lies in this array. Again unimodal maximum finding algorithm can be used to find the vertex with the maximum y-coordinate. The total running time is $\Theta(\lg n)$.

■ **Example** Let X[1..n] and Y [1..n] be two arrays, each containing n numbers already in sorted order. Give an O(lg n)-time algorithn to find the median of all 2n elements in arrays X and Y .

■ **Answer:** The median can be obtained recursively as follows. Pick the median of the sorted array A. This just takes O(1) time as median is the n/2th element in the sorted array. Now compare the median of A, call is a* with median of B, b*. We have two cases.

- a* < b* : In this case, the elements in B[n/2 · · · n] are also greater than a*. So the median cannot lie in either A[1 · · · n/2] or B[n/2 · · · n]. So we can just throw these away and recursively solve a subproblem with A[n/2 · · · n] and B[1 · · · n/2].
- a*> b* : In this case, we can still throw away B[1 · · · n/2] and also A[n/2 · · · n] and solve a smaller subproblem recursively.

In either case, our subproblem size reduces by a factor of half and we spend only constant time to compare the medians of A and B. So the recurrence relation would be T(n) = T(n/2) + O(1) which has a solution T(n) = O(log n).

■ **Example** An n element array contains unique numbers between 0 to n. Find out the missing element between 1 to n that is not seen in the array.

■ **Answer:** The missing integer in {0, 1, …, n} in an array of n elements in {0, 1, …, n} is the sum of all elements from1 to n (which is n(n+1)/2), minus the sum of the array elements.

We give the pseudocode for an algorithm that finds the missing integer. The algorithm runs in Θ(n) time.

```
MissingInteger(A, n)
   sum = 0; // compute the sum of the array elements
   For i = 1 to n do
          sum <- sum + A[i]
   return n(n + 1)/2 - sum
```

■ **Example** An n–1 element array contains unique numbers between 0 to n. Find out missing two elements (out of the numbers 0 to n) from the array.

■ **Answer:** We assume x and y are the two missing numbers.

We know the sum of n integers 0 to n is n(n+1)/2 and we calculate sum of all the array elements. Thus, x+y=n(n+1)/2-sum of the array elements.

Also, we know what is the sum of squares of n integers 0 to n is n(n+1)(2n+1)/6. Thus, $x^2+ y^2$=n(n+1)(2n+1)/6 - sum of squares of the array elements.

That is we can have two equations

x+y = a where a= n(n+1)/2-sum of the array elements

x^2+y^2 = b where b= n(n+1)(2n+1)/6 - sum of squares of the array elements

We can calculate x-y by carrying out simple algebraic manipulations as shown here.

$2b-a^2 = x^2 + y^2-2xy = (x–y)^2$.

Having x+y and x-y, we can calculate the missing numbers.

■ **Example** An n-2 element array contains unique numbers between 0 to n. Find out missing three elements (out of the numbers 0 to n) from the array.

■ **Answer:** We assume x,y,z are the two missing numbers.

We know the sum of n integers 0 to n is n(n+1)/2 and we calculate sum of all the array elements. Thus,

x+y+z=n(n+1)/2-sum of the array elements.

Also, we know what is the sum of squares of n integers 0 to n is n(n+1)(2n+1)/6. Thus,

$x^2+y^2+z^2$=n(n+1)(2n+1)/6 - sum of squares of the array elements.

In addition, we know what is the sum of cubes of n integers 0 to n is $(n(n+1) /2)^2$. Thus,

$x^3+y^3+z^3$=n(n + 1)(2n + 1)/6 – sum of squares of the array elements.

That is we can have three equations and three unknowns. We can solve for x, y, and z.

■ **Example** Calculate prefix averages of array elements of an array X. That is, given an X with n elements, we need to calculate for each of the element X[i] average of X[0], X[1], …X[i]. The following two solutions are proposed. Which is better? Explain.

Solution 1 :

for i ← 0 **to** n – 1 **do**
a ← 0
for j ← 0 **to** *i* **do**
a ← a + X[j]
A[i] ← a/(i + 1)
return array A

Solution 2:

s ← 0
for i ← 0 **to** n – 1 **do**
s ← s + X[i]
A[i] ← s/(i + 1)
return array A

■ **Answer:** Second one is preferable as its complexity is O(n) compared to $O(n^2)$ of first one.

■ **Example** The following is a recursive function that determines whether a given integer array is sorted or not. Is there anything missing in this function?. Does it works?

```
int isSorted(int data[], int n){
  if(n == 1)return 1;
  else{
        int temp = isSorted(data, n-1);
        return (temp) && (data[n-2] <= data[n-1]);
  }
}
```

■ **Answer:** Nothing is missing. It works without any problem.

■ **Example** Does the following is valid?

$1 + 2 + 3 + ... + \sqrt{n} = O(n)$

■ **Answer:** Sum becomes = $\sqrt{n}(\sqrt{n} + 1)/2$. As the resultant equation contains term with n, we can conclude that the given equality is valid.

■ **Example** Suppose an A is having following elements. 8 2 1 4 11 7 17

Also suppose that the function partition is called as follows: partition(A, 0, 6). What value does partition return? Draw the array A after the execution of partition.

```
int partition(int data[], int left, int right){
        int i, j;
        i = left;
        for(j = left; j <= right; j++){
              if(data[j] < data[i]){
                    swap(data, i, j);
                    i++;
                    swap(data, i, j);
              }
        }
        return i;
}
```

■ **Answer: 4**

2 1 4 7 8 11 17

■ **Example** Suppose we are given a sequence *S* of *n* elements, each of which is coloured red or blue. Assuming *S* is represented by an array, describe an in-place linear time algorithm for ordering *S* so that all the blue elements are listed before all the red elements.

■ **Answer:** To obtain a linear-time algorithm for this problem, we keep track of two indices into the array, *front* and *end* and traverse akin to quick sorting algorithm. Index *front* starts at the front of S and *end* start at the end of S. They move toward each other until they meet at which point the algorithm terminates. While they are traversing the sequence, every time *front* sees a red element and *end* sees a blue element, these two elements get swapped. This all the elements that *front* traverses, end up with blue elements, while all the elements that *end* traverses end up with red elements.

Note that, as required, this algorithm is in-place, i.e., it does not use any auxiliary data structures. Also since the *front* and *end* indices always move toward each other, each element of the sequence gets traversed by either one of them exactly once. Each time increment front or decrement end, we perform at most one swap, which takes constant time. Therefore the algorithm can run in $O(n)$.

While front < rear

Do
Increment front till it finds red element
Decrement end till it finds blue element
If front < end exchange red and blue
End do

■ **Example** Suppose we are given a sequence S of n elements, each of which is an integer in the range $[O,n^2 -1]$. Describe an algorithm for sorting S in $O(n)$ time.

■ **Answer:** Let k be an integer in the range $[O,n^2 -1]$. If we represent it in n-ary form, it would consist of two numbers (l,f) such that $k - l \times n + f$ where each l and f are in the range $[O, n -1]$. Conveniently, the lexicographical ordering of the n-ary representation corresponds to the ordering of the numbers. Therefore, Radix-sort can be applied directly to sort the sequence. Since radix-sort runs in $O(n)$, we only require $O(n)$ time to sort the sequence.

■ **Example** Two int arrays, A and B, contain m and n ints each, respectively. Elements of the arrays are in ascending order without duplicates (i.e. each table represents a set). The following function is to find the symmetric difference by producing a third array C (in ascending order) with the values that appear in exactly one of A and B and sets the variable p to the final number of elements copied to C. What is its complexity?

```
 int * symdifference(int A[], int m, int B[], int n){
int i=j=p=0;
while (i<m && j<n){
  if (A[i]<B[j])    C[p++]=A[i++];
  else if (A[i]>B[j])    C[p++]=B[j++];
  else{
    i++;
    j++;
  }
}
for ( ; i<m; i++) C[p++]=A[i];
for ( ; j<n; j++)  C[p++]=B[j];
return c;
}
```

■ **Answer:** O(m+n). It is more like merging in merge sort.

Linked Lists, Stacks and Queues

We assume the following structure for single linked list and double list problems. Nodes are assumed to be having integers.

```
struct lst{
int n;
struct lst *next;
}
struct dlst{
int n;
struct lst *next;
struct lst *prev;
}
```

■ **Example** Assuming H is the head of a single linked list, after the following while loop to which node H points?

```
while(H->next)H=H->next;
```

■ **Answer:** To last node.

■ **Example** Assuming H is the head of a single linked list, after the following while loop to which node H points?

```
while(H->next&&H->next->next)H=H->next;
```

■ **Answer:** To last but one node if the list contains more than one node, else it points to the first node itself.

■ **Example** Assuming H is the head of a single linked list, after the following while loop to which node H points? Critically comment if the list contains only one or two nodes?

```
while(H->next&&H->next->next)H=H->next;
```

■ **Answer:** In both the situations, H will be pointing to first node the while loop.

■ **Example** Write a function which takes head of single linked list and returns nth node address if available, otherwise returns 0.

```
struct lst * NthNode(struct lst *H, int n){
while(H&&n--)H=H->next;
return H;
}
```

Recursive Solution:

```
struct lst * NthNode(struct lst *H, int n){
if(H&&n--) return(NthNode(H->next, ));
else
return H;
}
```

■ **Example** Write both iterative and recursive versions to calculate sum of the values in the single linked list.

```
int sum(struct lst *H){
int s=0;
while(H){
s+=H->n;
H=H->next;
}
return s;
}
```

Recursive version:

```
int rsum(struct lst *H){
if(H)return( H->n +rsum(H->next));
else
return 0;
}
```

■ **Example** Write both iterative and recursive versions to calculate frequency of occurrence of a value in the single linked list.

```
int count(struct lst *H,int x){
int s=0;
while(H){
s+=(H->n==x);
H=H->next;
}
return s;
}
```

Recursive version:

```
int rcount(struct lst *H,int x){
if(H)return( (H->n==x) +rcount(H->next,x));
```

```
else
return 0;
}
```

■ **Example** Write a function to reverse a linked list. Not to reversely print the information of the linked list.

```
struct lst * rev(struct lst *H)
{
struct lst *B, *A;
if(H==0) return 0;
B=H->next;
H->next=0;
while(B){
A=B->next;
B->next=H;
H=B;
B=A;
}
return H;
}
```

■ **Example** Write a function to reverse a double linked list. Not to reversely print the information of the double linked list.

```
struct dlst * rev(struct dlst *H)
{
struct dlst *B, *A;
if(H==0) return 0;
B=H->next;
H->next=0;
while(B){
A=B->next;
B->next=H;
H->prev=B;
H=B;
B=A;
}
H->prev=0;
return H;
}
```

The following version also reverses the double linked list.

```
struct dlst * rev1(struct dlst *H)
{
struct dlst *B, *A, *C;
if(H==0) return 0;
while(H->next)H=H->next;
B=H;
while(B){
         C=B->prev;
```

```
        A=B->next;
        B->next=B->prev;
        B->prev=A;
B=C;
}
```

■ **Example** Write a function that returns number of nodes in a circular linked list by taking the address of a circular single linked list.

```
int count-nodes (struct lst *p) {
int count=0;
struct lst *start=p;
while (p->next != start)
{
count ++;
p = p->next;
}
return count;
}
```

■ **Example** Parentheses Matching Algorithm

An array X of n tokens, each of which is either a grouping symbol (brackets), a variable, an arithmetic operator, or a number. This function returns if all the grouping symbols in X match else returns false. What is the complexity of this algrorithm?

Algorithm ParenMatch(X,n):
Let S be an empty stack
for i=0 to n-1 do
if $X[i]$ is an opening grouping symbol then
S.push($X[i]$)
else if $X[i]$ is a closing grouping symbol then
if S.isEmpty() then
return false *{*nothing to match with*}*
if S.pop() does not match the type of $X[i]$ then
return false *{*wrong type*}*
if S.isEmpty() then
        return true *{*every symbol matched*}*
else
        return false *{*some symbols were never matched*}*

Answer: We know that we will be iterating each symbol of X. Thus complexity of this algorithm O(n) where n is the number of symbols in X.

What are the advantages and dis-advantages of using a linked list to implement a stack rather than an array?

One advantage of using a linked list to implement a stack is that a linked list does not have a fixed size. This means that our linked list can grow to the size of the stack. We do not need to worry about setting aside a certain block of memory for the stack.

One disadvantage of using a linked list is the increased complexity of using pointers. With pointers it is easier to make mistakes while implementing the logic of the stack operations.

While implementing a stack using an array, what is the first thing one need to do?

The first thing we must do is reserve a group of memory cells large enough to hold all the items which we want to put in the stack.

Which item is always the last item to leave a stack?

The last item to leave a stack is the item that is on the bottom of the stack or the first item that entered the stack.

What is the purpose of the stack pointer?

The stack pointer is used to keep track of the top of the stack. This pointer will always hold the memory location of the last item added to the stack.

■ **Example** How to find a cycle in a single linked list. Rather, how to check a given single linked list is circular or not. What is its complexity.

■ **Answer** We propose to use two pointers x and y. In the beginning of the algorithm, pointer x points to first element of the list and pointer y points to the second element of the list. Then, in each iteration of the a loop, pointer y moves two steps forward and pointer x moves one step forward. We continue the loop till x is not equal to y. If a loop exists in the list, then at some points, pointer x and pointer y will point to the same element.

The worst-case time complexity of this algorithm can be O(n).

■ **Example** Identify logical errors in the following function which is proposed to delete last node of a single linked list.

```
void delLast( Node *& head ) {
Node *p = head ;
while( p->next->next != 0 ) {
p = p->next ; }
p->next = 0 ;
}
```

■ **Answer:** There are at least two logic errors in the routine. List all errors you can spot:
The algorithm will dereference a null pointer in the case where the list is of length 1.
The algorithm does not delete the node from the heap. Thus it has a space leak.
To alleviate first error we can change while loop as while(p->next!=0)
To alleviate second problem, we can add an instruction or two to remove node.

2.4.4 Trees

In a nutshell, a tree is said to be having a set of nodes (G) and edges (E) which joins them. Of course, the same definition is used for graphs also. However, main difference is that the tree is also a graph, to be specifically acyclic graph. Evidently tree contains parent, child and grand child type hierarchical relationship.

A tree is a set of nodes which is either null or with one node designated as the root and the remaining nodes partitioned into smaller trees, called sub-trees.

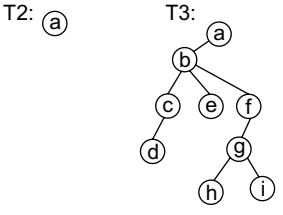


Figure 2.15 Sample Tree

■ **Example:**
T1={} (NULL Tree. Not shown in the figure 2.15)
T2={a} a is the root, the rest is T1
T3={a, {b,{c,{d}},{e},{f,{g,{h},{i}}}}

Another application of trees is in compiler construction where expression or parse trees (see Figure 2.16) are used for verifying the validity of expressions.

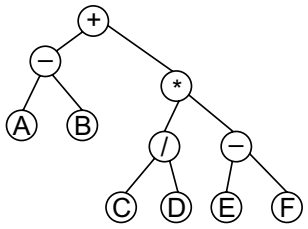


Figure 2.16 An example parse or expression tree

Notations

- **Root Node:** It is the top most node in the tree.
- **Degree of the Node:** The degree of a node is the number of partitions in the subtree which has that node as the root.
- **Leaf Node:** The one which does not have any more children. That is the nodes with degree=0 are called leaves.
- **Non-Leaf Node:** The node which contains child's. Non-Leaf nodes degree value will be other than zero.
- **Level:** The level of a node is the length of the path from the root to that node. Root node is said to be at 0^{th} level. Level numbers increases downwards.
- **Height (or depth) of the Tree:** Number of levels in which nodes are organised. The depth of a tree is the maximum level of any node of any node in the tree.
- **Left sub-tree or Sibling:** Left side portion of any non-leaf node is called as its left sub-tree or sibling.
- **Internal node:** Not a root or a leaf.
- **Parent:** Node with out-degree greater than 0.
- **Child:** Node with in-degree greater than 0.
- **Siblings:** Nodes with the same parent.
- **Path:** Sequence of adjacent nodes.
- **Right sub-tree or Sibling:** Right side portion of any non-leaf node is called as its right sub-tree or sibling.
- **Ancestor:** Node in the path from the root to the node.
- **Descendent:** Node in a path from the node to a leaf.
- **Sub-tree:** Connected structure below the root.
- **Degree:** The degree of a node is the number of partitions in the sub-tree which has that node as the root.
- **Balance of a node:** Balance of a node is the difference in heights of its left and right sub-trees or siblings. For leaf nodes, balance value will be 0.
- **Binary Trees and Multiway Trees:** Binary trees are the ones in which any node contains at most two children. That is maximum allowed degree of any node in a binary tree is 2. Violation of the above rule is seen in the multi-way trees. That is the nodes may have more than two children.
- A **binary tree** may contain up to 2^n nodes at level n.
- **Strictly binary tree** is the one whose non-leaf nodes contain exactly two children; i.e., its non-leaf nodes will be having degree value of 2.
- A **complete binary tree** of depth N has 2k nodes at levels k=0,…,N–1 and all leaf nodes at level N occupy leftmost positions. Complete binary tree is the one in which all leaf nodes will be at the lowest level and every non-leaf node will be having exactly two children. Thus, complete binary tree is evidently strictly binary tree.
- **Almost complete binary tree** is the in which all the leaf nodes will be at lowest level or one level above it. Some authors further divide these trees as **"almost complete strictly binary tree"** and **"almost complete binary tree"**, where strictness is not maintained.
- **Balanced Binary tree or AVL tree:** The tree whose nodes balance values are between –1 to 1.
- If level N has 2N nodes as well, then the complete binary tree is a full tree.
- If all nodes have degree = 1, the tree is a degenerate tree (or simply linked list).

- If a node in a binary search tree contains two children (leaf), then its successor has no left child and its predecessor has no right child.
- The deletion operation on binary search tree is commutative. That is, first if we delete x and followed by y then the resulting tree will be same as if we delete y first followed by x.

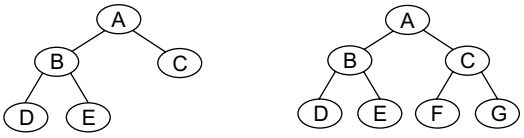


Figure 2.17 Allmost complete binary tree and complete binary tree

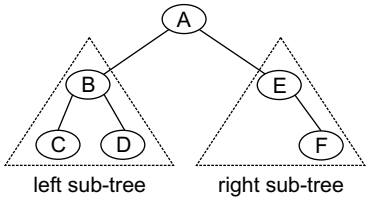


Figure 2.18 Shows left and right subtrees of a binary tree

A binary tree may contain up to 2n nodes at level n.

A complete binary tree of depth N has 2k nodes at levels k=0,…,N-1 and all leaf nodes at level N occupy leftmost positions.

If level N has 2N nodes as well, then the complete binary tree is a full tree.

- If all nodes have degree=1, the tree is a degenerate tree (or simply linked list). For example, a degenerate tree of depth 5 has 6 nodes.
- A full tree of depth 3 will have 1 + 2 + 4 + 8 = 15 nodes.
- A full tree of depth N has 2N+1–1 nodes.
- A complete tree of depth N has 2N – 1<=m <=2N+1–1 nodes.
- If N is the number of nodes then the depth of almost complete strictly binary tree will be less than or equal to $\log_2(N)$.
- For a given height, complete binary tree will be having more number of nodes than any binary tree organisations.
- For any strictly binary tree the equality "No of Leaf Nodes – No of Non-Leaf Nodes =1" will be satisfied.
- Given number of keys, height of the tree becomes minimum if they are organised in complete binary tree fashion compared to any other binary tree configurations.
- If lowest level in a complete binary tree is K, then number of leaf nodes with 2^K and number of non-leaf nodes will be 2^K–1. Total number of nodes will be 2^{K+1} –1.
- If H is the height of the complete binary tree, the number of leaf nodes are 2^{H-1}, number of non-leaf nodes are 2^{H-1}–1 and total number of nodes are 2^H–1.
- **Binary Search Trees** From the name itself we can know that Binary search trees are used for searching applications. Binary search is the one which some order exists among the information available in the nodes. For example, if the nodes contains numbers, then all the nodes which are left to a node will be having their node values smaller and right to it will be having their values larger as shown in Fig. 2.19.

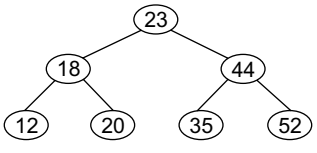


Figure 2.19 Binary search tree

Of course, one way we can think a linked list with ascending ordered values also as a binary search tree. To be critically, such a trees are called as skewed trees or degenerate trees.

- **Max Heap and Min Heap** Max heap is a special type of tree in which any node value will be more than its children. For example, see the Fig. 2.20.

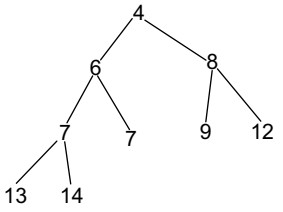


Figure 2.20 Min Heap

Min Heap is a complimentary to max heap. That is, any node value will be less than its children values. We have discussed about max heap in heap sorting. More over, heaps are nearly complete binary trees. That is all nodes at the lowest level will be in the left side as shown in Fig. 2.21.

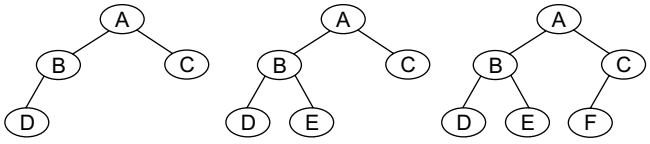


Figure 2.21 Acceptable heaps

Tree Traversals

In previous chapters, we have discussed about various structures like linked lists, double linked lists, circular lists, etc., we have always demonstrated first creation of them followed by traveling them. In the case of single linked list we have freedom to traverse from head node to tail node. While with double linked list, we can also traverse from tail to head node. In the case of circular lists we can traverse cyclically. In the same fashion, trees also can be traversed in the following ways.

1. In-Order Traversal
2. Pre-Order Traversal
3. Post-Order Traversal
4. Level-Order or Breadth first traversal

While traversing the information in the node will be processed. Here, processing can be simply printing/displaying or comparing with a search key, etc. Always, we assume traversal starts from the root node.

In-Order Traversal

At each node, A, we have to apply the following sequence of operations:

- Traverse Left sub-tree of A in In-Order Fashion
- Process A
- Traverse Right sub-tree of A in In-Order Fashion.

For example, if we traverse the tree in Fig. 2.22 we get the sequence as: CBDAEF

Pre-Order Traversal

At each node, A, we have to apply the following sequence of operations.

- Process A
- Traverse Left sub-tree of A in Pre-Order Fashion
- Traverse Right sub-tree of A in Pre-Order Fashion.

For example, if we traverse the tree in Fig 2.22 we get the sequence as:ABCDEF

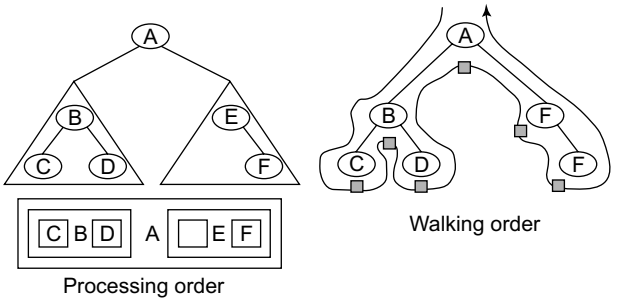


Figure 2.22 In-Order Traversal

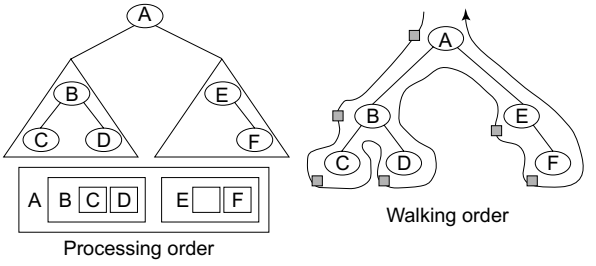


Figure 2.23 Pre-Order Traversal

Post-Order Traversal

At each node, A, we have to apply the following sequence of operations

- Traverse Left sub-tree of A in Post-Order Fashion
- Traverse Right sub-tree of A in Post-Order Fashion.
- Process A

For example, if we traverse the tree in Fig. 2.24 we get the sequence as: CDBFEA

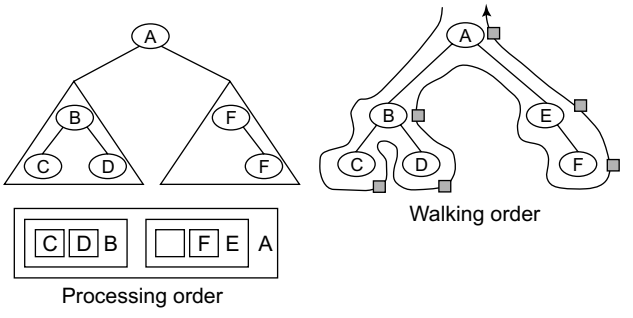


Figure 2.24 Post-Order Traversal

If we traverse an expression tree In-Order fashion, we will get its infix representation; pre-order fashion we will get its prefix representation; post-order traversal gives postfix representation.

Similarly, if we traverse a binary search such as the one in Fig. 2.19 in In-Order fashion, we get ascending ordered sequence of the node values. For example, the In-Order traversal of the BST in Figure 2.19 gives: 12, 18, 20, 23, 35, 44, 52.

Level-order Traversal

This is some what different from previous traversals. Here, all the nodes at each level will be processed from left-to-right starting from 0^{th} level. For example, level order traversal of the Fig. 2.25 gives sequence as: ABECDF

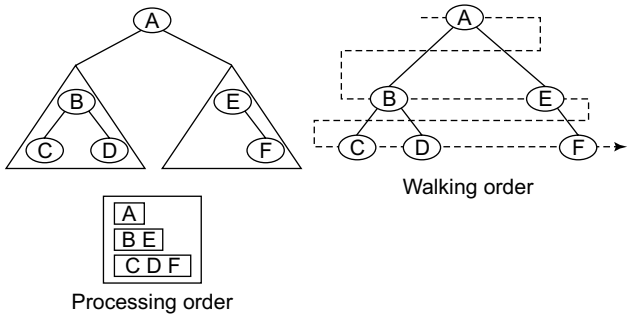


Figure 2.25 Level Order Traversal

Creating Binary Search Tree

Node structure of a binary search tree with integer information can be taken as:

```
struct Node
{
int n;
struct Node *left;
struct Node *right;
};
```

The above structure is assumed in the following questions:

1. **What will be the result of the following function on a binary search tree?**

```
strunt Node* xyz(struct Node *A)
{
if(A==0) return 0;
while(A->left)A=A->left;
return A;
}
```

Answer: It returns the address of the left most node.

2. **What will be returned value from the following function?**

```
struct Node * XYZ (struct Node *root) {
  if (! root -> left)
return (root)
else return XYZ (root -> left);
}
```

Answer: It returns the address of the left most node of a BST.

3. **How to return minimum value of a BST with integers as its node values? Do propose recursive solution also.**

Answer: We find out left most node and return the value available in it.

```
int minValue(struct Node *A){
if(A==0) return 0;
while(A->left)A=A->left;
return A;
}
int rminValue(struct Node *A){
if(A==0) return 0;
else if(A->left)return rminValue(A->left);
else return A->n;
}
```

4. **What will be the result of the following function on a binary search tree?**

```
struct Node* PQR(struct Node *A)
{
if(A==0) return 0;
while(A->right)A=A->right;
return A;
}
```

Answer: It returns the address of the right most node.

5. **How to return maximum value of a BST with integers as its node values? Do propose recursive solution also.**

```
int maxValue (struct Node *A)
{
if(A==0) return 0;
while(A->right)A=A->right;
return A->n;
}
int rmaxValue(struct Node *A){
if(A==0) return 0;
```

```
else if(A->right)return rmaxValue(A->right);
else return A->n;
}
```

6. What is the use of the following function?

```
void insertbst(struct Node **H, struct Node *A){
if(*H==0)*H=A;
else if((*H)->n <A->n) insertbst(&(*H)->right, A);
else insertbst( &(*H)->left, A);
}
```

Answer: It inserts a node in a binary tree in recursive manner.

7. Explain how to traverse a tree using a stack.

The following pseudo code gives us an idea of how traversal can be done using stack. Let p is the pointer to root node of the tree.

```
do
{
while(p !=NULL){
               push(p);
               p=p->left;
              }
if(stack if not empty)
{
 p=pop();
print information of node p;
p=p->right;
}
}while(stack is not empty or p is not NULL);
What does the following function do?
int H(struct Node *A){
if(A==0) return 0;
else if(A->left==0 && A->right==0) return 1;
else
return(1+H(A->left)+ H(A->right) );
}
```

Answer: It returns the total nodes in a tree whose root node is sent as argument to the function. It uses recursive traversal. When it encounters a leaf node it returns 1. If it encounters a non leaf node, it recursively finds left side nodes, right side nodes and returns their sum by adding one.

8. What does the following function does?

```
int NL(struct Node *A){
if(A==0) return 0;
else if(A->left==0 && A->right==0) return 1;
else
return(NL(A->left)+ NL(A->right) );
}
```

Answer: It returns number of leaf nodes in the tree whose root node address is given as argument.

9. What does the following function does?

```
int NONL(struct Node *A){
if(A==0) return 0;
else if(A->left==0 && A->right==0) return 0;
else
return(1+NONL(A->left)+ NONL(A->right) );
}
```

Answer: It returns number of non-leaf nodes in the tree whose root node address is given as argument.

10. What does the following function does?

```
int HEIGHT(struct Node *A){
int L,R;
if(A==0) return 0;
else if(A->left==0 && A->right==0) return 1;
else
{
 L=HEIGHT(A->left);
 R=HEIGHT(A->right);
return(1+ ( L<R? R:L) );
}
}
```

Answer: It returns the height of the tree.

11. What does the following function does?

```
int COMPLETE(struct Node *A){
int L,R;
if(A==0) return 0;
else if(A->left==0 && A->right==0) return 1;
else{
 L=COMPLETE(A->left);
 R=COMPLETE(A->right);
 if( (L==-1) || (R==-1) || (L!=R)) return -1;
return(1+L);
}
}
```

Answer: It takes address of the root node of a BST and returns 0 if it is not a complete binary tree otherwise it returns height of the tree.

12. What does the following function does?

```
int BAL(struct Node *A){
int L,R;
if(A==0) return 0;
else if(A->left==0 && A->right==0) return 1;
else{
L=BAL(A->left);
R=BAL(A->right);
if( (L==-1) || (R==-1) || (abs(L-R)>1 )) return -1;
```

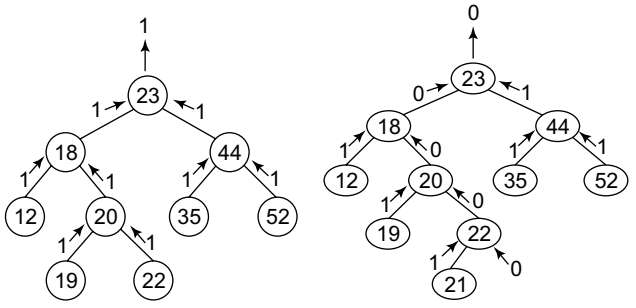
```
return(1+(L<R?R:L));
}
}
```

Answer: It takes address of the root node of a BST and returns - if it is not a balanced binary tree otherwise it returns height of the tree.

13. What does the following function does?

```
int STRICT(struct Node *A){
int L,R;
if(A==0) return 0;
else if(A->left==0 && A->right==0) return 1;
else{
 L=STRICT(A->left);
 R=STRICT(A->right);
return(L*R);
}
}
```

Answer: It takes address of the root node of a BST and returns 0 if it is not a strictly binary tree otherwise it returns 1. For example, see working of the above function.



14. What does the following function does?

```
int identicaltopology(struct Node* a, struct
Node* b) {
if (a==NULL && b==NULL){return(1);}
else if (a!=NULL && b!=NULL) {
return(
identicaltopology(a->left, b->left) &&
identicaltopology(a->right, b->right));
}
else return(0);
}
```

Answer: It takes addresses of the root nodes of two BST's and ruturns 1 if they are topologically same else returns 0.

15. What does the following function does?

```
int identical(struct Node* a, struct Node* b) {
if (a==NULL && b==NULL){return(true);}
else if (a!=NULL && b!=NULL) {
return(a->n == b->n &&
identical(a->left, b->left) &&
identical(a->right, b->right));
}
else return(0);
}
```

Answer: It takes addresses of the root nodes of two BST's and ruturns 1 if they are topologically same and also their nodes values are same, else returns 0.

16. What does the following function does?

```
struct Node *copy(struct Node *root){
struct Node *temp;
if(root==NULL)return(NULL);
temp = (struct Node *) malloc(sizeof(struct Node));
temp->n = root->n;
temp->left = copy(root->right);
temp->right = copy(root->left);
return(temp);
}
```

Answer: It creates copy of a BST.

17. What does the following function does?

```
void mirror(struct Node* node) {
struct Node *temp;
if (node==NULL) {
return;
}
else {
mirror(node->left);
mirror(node->right);
// Swap the pointers of this node
temp = node->left;
node->left = node->right;
node->right = temp;
}
}
```

Answer: It creates mirror of a tree.

18. What does the following function does?

```
struct Node * Find(struct Node *A, int x){
if(A==0) return 0;
while(A){
if(A->n==x)return A;
else if(A->n<x) A=A->right;
else A=A->left;
}
```

```
return A;
}
```

Answer: It takes address of the root node of a tree and a key. It returns the address of the node having the given key, else returns 0.

19. What does the following function does?

```
int mirror(struct Node* a, struct Node* b) {
if (a==NULL && b==NULL){return(1);}
else if (a!=NULL && b!=NULL) {
return(
mirror(a->left, b->right) &&
mirror(a->right, b->left));
}
else return(0);
}
```

Answer: It takes addresses of the root nodes

20. What does the following function does?

```
struct Node * RFind(struct Node *A, int x){
if(A==0) return 0;
else if(A->n ==x) return A;
else if( A->n<x) return(RFind(A->right,x));
else return (RFind(A->left,x));
}
```

Answer: This is the recursive version of Find(). If A is null, we return null. If A contains x then it returns A. Otherwise, we will search for x in its left and right sub-trees recursively with the same key value.

21. Explain how to return kth element of the ordered list.

Answer: Assume that our binary tree node contains one extra data member, linkcount, which indicates the number of nodes left of it.

```
struct Node
{
int n;
int linkcount;
struct Node *left;
struct Node *right;
};
```

Assuming, some how we have created a binary search tree with linkcount values also. The following function returns the address of kth element of the ordered list or not.

```
struct Node * kthelement(struct Node *H, int k)
{
if( H==0) return 0;
while( H &&  ( H->left || H->right ) )
{
if(H->linkcount >= k) H=H->left;
else
{
k -=H->linkcount;
H=H->right;
}
return H;
}
```

22. How to find out in-order successor of a node with a given value.

In-Order successor of Node, p, can be calculated by employing the following rules:

- If p is left child to its parent and p does not have any right sub-tree then its parent itself becomes in-order successor of p.
- If p is left child to its parent and p has right sub-tree, then left most child of p's right sub-tree becomes the p's in-order successor.
- If p is right child of its parent and p has right sub-tree, then left most child of p's right sub-tree becomes the p's in-order successor.
- If p is right child of its parent and p does not have any right sub-tree, then back track using father field till we will find a node q which is left to its parent and then return q's father as p's in-order successor.
- If p is right most node of a tree then its in-order successor is NULL.
- If p is root node then, left most child of its right sub-tree is the in-order successor.

In Fig. 2.26 for nodes marked as A,B, the in-order successors are their parents as A,B do not have right sub-trees.

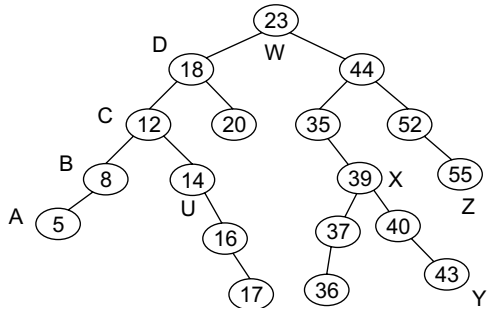


Figure 2.26 In-Order Successor

For nodes C,D, in-order successors are the left most nodes of its right sub-tress. In this, case they 14 and 20, respectively.

For nodes Y, Z (which are right children's for their parents and do not have any right sub-trees), in-order successors are 35 and NULL.
For nodes, U, X in-order successors are left most children of their right sub-trees. That is, 16 and 40, respectively.
For root node, W, in-order successor is 35.

Note

In-Order predecessors of a node can be easily calculated by replacing left with right and right with left in the above statements.

What are threaded binary trees?
Threaded Binary Trees are the ones which may contain nodes with their links pointing to their in-order successors or predecessors. That is, a binary tree is *threaded* by making all right child pointers that would normally be null, point to the inorder successor of the node. This type of trees is called as right threaded tree. If we make left child pointers to point to inorder predecessors, the resulting tree is called as left-threaded tree.

Give a binary search tree and a number, inserts a new node with the given number in the correct place in the tree. Returns the new root pointer which the caller should then use.

```
struct Node* insert(struct Node* node, int data) {
if (node == NULL) {
return(newNode(data));// create a new node with data
as its value
}
else {
if (data <= node->n) node->left = insert(node->left, data);
else node->right = insert(node->right, data);
return(node);
}
}
```

23. Write a function which takes root of a tree and an integer sum, return true if there is a path from the root down to a leaf, such that adding up all the values along the path equals the given sum; else it returns 0. Explain procedure also.

Answer: Strategy: subtract the node value from the sum when recurring down, and check to see if the sum is 0 when you run out of tree.

```
int hasPathSum(struct Node* node, int sum) {
// return true if we run out of tree and sum==0
if (node == NULL) return(sum == 0);
else {
// otherwise check both subtrees
int subSum = sum - node->data;
return(hasPathSum(node->left, subSum) ||
hasPathSum(node->right, subSum));
}
}
```

24. What does the following function does?

```
void doubleTree(struct Node* node) {
struct Node* oldLeft;
if (node==NULL) return;
// do the subtrees
doubleTree(node->left);
doubleTree(node->right);
// duplicate this node to its left
oldLeft = node->left;
node->left = newNode(node->data);// creates a
new node
node->left->left = oldLeft;
}
```

Answer: For each node in the given binary search tree, it creates a new duplicate node, and insert the duplicate as the left child of the original node. The resulting tree should still be a binary search tree.

25. For the key values 1...numKeys, how many structurally unique binary search trees are possible that store those keys. Explain the logic.

Answer: Consider that each value could be the root. Recursively find the size of the left and right subtrees.

```
int countTrees(int numKeys) {
if (numKeys <=1) {
return(1);
}
else {
// there will be one value at the root, with
whatever remains
// on the left and right each forming their own
subtrees.
// Iterate through all the values that could be
the root...
int sum = 0;
int left, right, root;
for (root=1; root<=numKeys; root++) {
left = countTrees(root - 1);
right = countTrees(numKeys - root);
// number of possible trees with this root ==
left*right
sum += left*right;
}
return(sum);
}
```

26. Write a function which takes root node of a BST and returns true if a binary tree is a binary search tree; else return false.

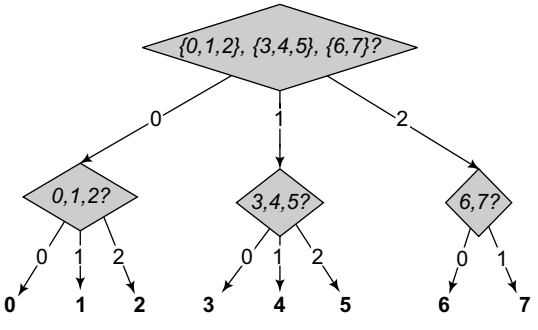
```
int isBST(struct Node* node) {
if (node==NULL) return(true);
// false if the min of the left is > than us
if (node->left!=NULL && minValue(node->left) >
node->n) return(false);
// false if the max of the right is <= than us
if (node->right!=NULL && maxValue(node->right)
<= node->n) return(false);
// false if, recursively, the left or right is not a BST
if (!isBST(node->left) || !isBST(node->right))
return(false);
// passing all that, it's a BST
return(true);
}
```

27. Write a function which takes root of a BST, an integer x and returns true (1) if atleast one path of the tree is having node values sum along that path as x, otherwise returns false(0).

```
int hasPathsum(struct Node *A, int x){
int subsum;
if(A==0) return (x==0);
else {
subsum=x - A->n;
return (hasPathsum(A->left, subsum) | has Path-
sum (A->right, subsum) );
}
```

28. Consider a quantity trit where one trit can distinguish between three equally likely values. For each trit, we can ask a ternary question (a question with three possible answers). How many trits are needed to distinguish among eight possible values?

Answer: Two trits are required since $\log_3 8 = 1.893$. Here is one such tree:



29. Devise a general formula for converting between bits and trits. How many trits are needed to describe b bits of information?

Answer:

To convert between logaithm bases, we use the formula

$$\log_b x = \frac{\log_a x}{\log_a b}.$$

So, for a n-bit value x,

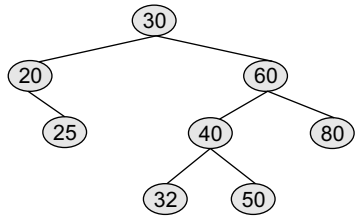
$$\log_b x = \frac{n}{\log_2 3} \approx 0.63093n$$

Thus, to represent an n-bit value required up to $\lfloor 0.63093n \rfloor$ trits.

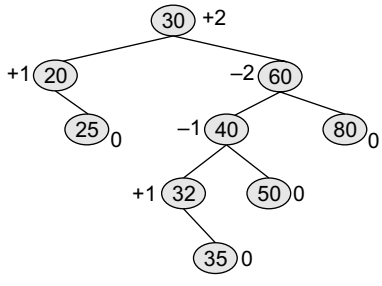
30. What's the Problem with Binary Search Trees?

The problem with binary search trees is that they can get thin and skewed, and to support fast insertions, searches and deletions they must be fat and bushy. All the operations we perform on a binary search tree take time proportional to the height of the tree. But the tree in the worst case can turn into a long thin straight line, so the time complexity becomes O(n) instead of O(log n). To alleviate this problem, we go for height balancing of the tree.

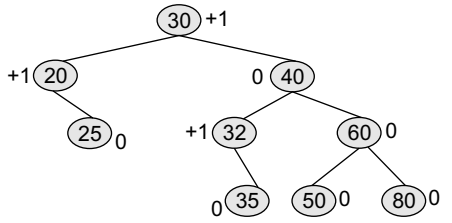
For the AVL tree shown here, perform the necessary rotations to balance the tree assuming a new node containing 35 is inserted into the tree. For each rotation that is required identify the type of rotation, i.e., right, left, or double. Show the balance factors at every node before and after the rotation. Draw the resulting AVL tree.



Insertion of new node 35 unbalances the tree at node 60 (the grandparent of the new node) as well as at the root (the great-grandparent of the new node).

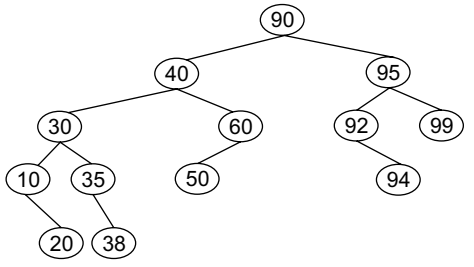


First rotation is 40 about 60 (the first unbalanced point working from the new node backwards along the path to the root). This is a right rotation that will produce the following tree.

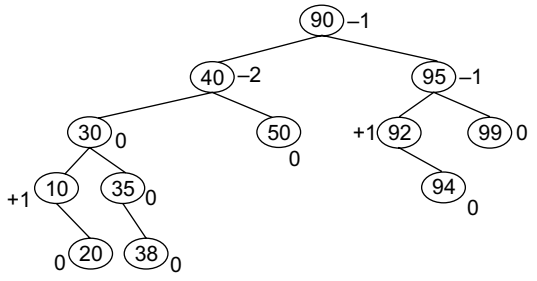


The single right rotation has restored balance in the tree and we are finished.

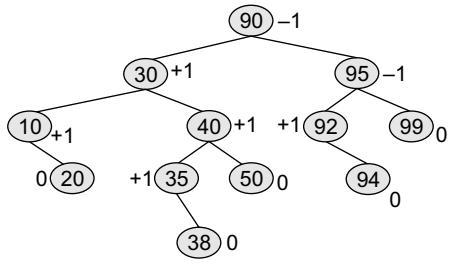
For the AVL tree shown here, perform the necessary rotations to balance the tree assuming the node containing 60 is deleted from the tree. For each rotation that is required identify the type of rotation, i.e., right, left, or double. Show the balance factors at every node before and after the rotation.



Deletion of node 60 will physically replace 60 with node containing 50. The resulting tree is shown below with the balance factor of each node shown adjacent to the node.



An imbalance has occurred at node 40 due to the deletion in its right subtree. A right rotation of 30 about 40 will restore balance. This is shown in the next tree. Tree is now balanced.



31. What is the difference between height balanced and weight balanced trees?

The height balanced trees keep the left and right heights from each node balanced, while the weight balanced trees keep the number of nodes in each right and left subtree balanced. They are similar in theory but height balanced won favour over weight-balanced trees.

32. What is the problem with maintaining things in an array?

Insertions take O(n) – because we have to shift over elements to make room for a new leaf, like insertion sort.

33. What is one reason that the physical representation of a tree is much different from the logical representation?

One reason that the logical representation of a tree is much different from the physical representation is the use of pointers to map a nonlinear data structure onto linear computer memory.

34. How to implement operations on binary trees without pointers?

The following code fragment illustrates this.

```
#include <stdio.h>
#include <strings.h>
#include <stdlib.h>
// Trees without pointers...
struct  {
char data[16];
int left;
int right;
} list[100];
int length=0; // assume root is 0
void print_tree(int root) {
  if(root !=-1)  {
    print_tree(list[root].left);
    puts(list[root].data);
    print_tree(list[root].right);
  }
}
int search(int i,char s[]) {
if(i==-1) return 0; // not found
if(strcmp(s,list[i].data)<0)   return   search
(list[i].left,s);
if(strcmp(s,list[i].data)>0)   return   search
(list[i].right,s);
```

```
return 1; // found
}
void newnode(int n, char s[], int side) {
       if(side==0) list[n].left=length;
       if(side==1) list[n].right=length;
       strcpy(list[length].data,s);
       list[length].right=-1;
       list[length].left=-1;
       length++;
}
void insert(int i,char name[]) {
  if (strcmp(name,list[i].data) < 0) {
        if(list[i].left==-1) newnode(i,name,0);
        else
        insert(list[i].left,name);
   }
  else if (strcmp(name,list[i].data) > 0){
       if(list[i].right==-1) newnode(i,name,1);
        else
        insert(list[i].right,name);
   }
}
void insert_node(char name[]) {
  if(length == 0) {
    newnode(0,name,-1);
  } else
  insert(0,name);
}
void main()
{
int t1;
  insert_node("Rama");
  insert_node("Sita");
  insert_node("Lakshman");
  insert_node("Hanuman");
  insert_node("Siva");
  insert_node("Christ");
  insert_node("Vinayaka");
  insert_node("Parvathi");
  insert_node("Venkateswara");
  print_tree(0);
  t1=search(0,"Sita");
  if (!t1)
    puts("Sita Not Found");
  t1=search(0,"Venkat");
  if (!t1)
    puts("Venkat Not Found");
}
```

Explain about Red-Black Trees

A Red-Black is a binary search tree where each node is colored Red or Black, every Red node has only Black children, every leaf node (nil) is Black, and all the paths from a fixed node to any leaf contain the same number of Black nodes. The operations on a Red-Black tree are efficient; height of a Red-Black tree with n nodes is at most 2 lg(n + 1), hence they are relatively bushy trees.

Insertions into a Red-Black Tree

Inserting a value into a binary search tree takes place at a leaf. What properties of the Red-Black tree might this violate. If we colour the new node Red, and make its nil children Black, then the number of Black nodes on any path has not changed, all nodes are still either Red or Black, all leaves are Black, and the children of the new Red node are Black. The only property to worry about is whether the parent of the new leaf is also colored Red, which would violate the rule about Red nodes having to have only Black children.

Fixing this property is not simple. We will need to recolour nodes, pushing up the Red-Red clash until finally get rid of it. In the worst case we need to recolor values all the way up the tree. In order to do this recoloring, we require 1 or 2 *rotations* at the end which involve a mutation of the search tree in addition to a recolouring.

Rotations

A rotation is a way to reshape a binary search tree that maintains the structure of the inorder traversal. It is an important tool for managing any height-balanced trees. Left and right rotations are inverses of one another and the basic movements are shown in Fig. 2.27.

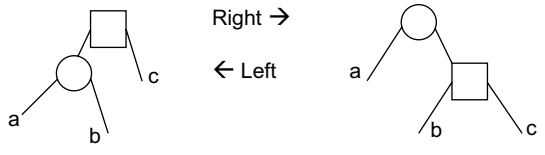


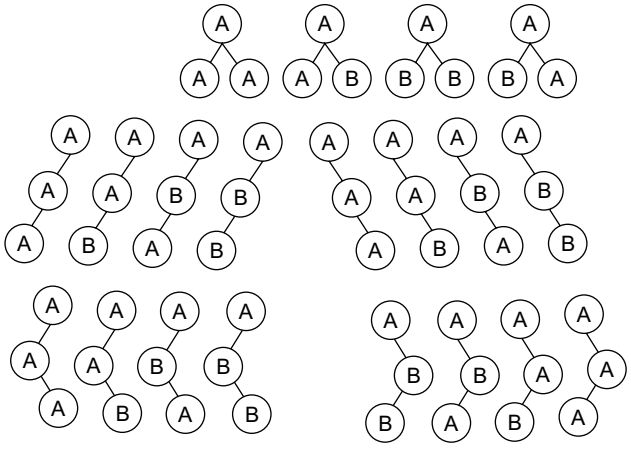
Figure 2.27

■ **Example** Consider the set S of binary trees having 3 nodes (including the root). Now consider the set S' of labeled binary trees formed from S as follows: for each tree in S, each node (except the root) can be labeled either A or B. Assume that the root always has the label A.

(a) How many labeled binary trees are in S′?

(b) If one tree s′ of S′ is chosen randomly, what is the expected number of nodes in s' that have the label A? Assume that, for any two trees s'_1 and s'_2 in S′, the probability of choosing s'_1 = the probability of choosing $s'_2 = 1/|S'|$

Solution: (a) 20



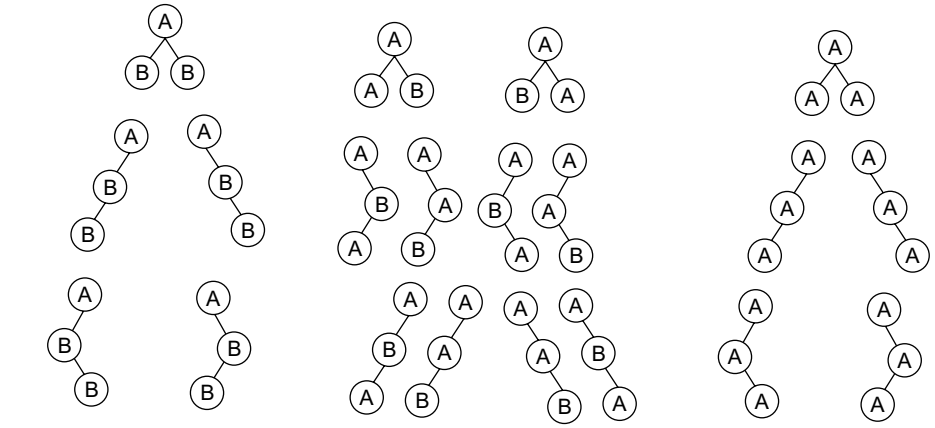
(b) The expected number of nodes labeled A = 2 =

$$\sum_{i=0}^{3} i \Pr(i\ A-labeled\ nodes)$$

$$= 0\Pr(0\ A-labeled\ nodes)+1\Pr(1\ A-labeled\ node)$$

$$+2\Pr(2\ A-labeled\ nodes)+3\Pr(3\ A-labeled\ nodes)$$

$$= 0+1\frac{5}{20}+2\frac{10}{20}+3\frac{5}{20}=\frac{5}{20}+\frac{20}{20}+\frac{15}{20}=\frac{40}{20}=2$$



There are 5 3-node, A/B labeled, binary trees containing 1 A-labeled node (assuming root is labeled A).

There are 10 3-node, A/B labeled, binary trees containing 2 A-labeled nodes (assuming root is labeled A).

There are 5 3-node, A/B labeled, binary trees containing 3 A-labeled nodes (assuming root is labeled A).

Important Points about the Trees

1. Tree is a non-linear structure which has child, parental type hierarchical relationship. Binary trees are the ones in which a node can have at most two children. Multi-way trees will be having more than two children.

2. Every binary tree except for empty binary tree is indeed a tree.

3. Strictly binary trees are the ones in which every non-leaf node will have exactly two children.

4. Complete binary tree is the one in which all the leaf nodes will be at lowest level and every non-leaf node will be having exactly two children. Thus, complete binary tree is evidently strictly binary tree.

5. Almost complete binary is the one in which, all the leaf nodes will be at lowest level or one level above it; and it is evidently strictly binary tree. In some text books these trees are called as "almost complete strictly binary tree", while defining another type as almost complete binary where strictness is not maintained.

6. If N is number of nodes then the depth of almost complete strict binary tree will be less than or equal to $\log_2 N$.

7. For any strictly binary tree the equality "No of Leaf Nodes = No of Non-leaf Nodes + 1" satisfies.

8. For a given height, complete binary tree will be having more number of nodes any other binary tree organisations.

9. Given number of keys, height of the tree becomes minimum if they are organised in complete tree fashion compared to any other binary tree configurations.

10. If lowest level in a complete binary tree is K then no of leaf nodes are 2^K, no of non-leaf nodes are 2^{K-1}, total no of nodes are $2^{K+1} - 1$. Similarly, if the height of the tree is H then total number of leaf nodes are 2^{H-1}, no of non-leaf nodes are $2^{H-1} - 1$, and total number of nodes are 2^{H-1}.

11. Binary search tree is the one in which some order exists among the information available in the nodes. For example, if the node contains numbers then all the nodes which are left to it will be having their node values smaller than it and the nodes right to it will be having their values larger than it. However, there is no restriction on number of child nodes for them.

12. If the keys are inserted in random order balanced trees result more often than not, so that on the average search time becomes log(n), where n is no of keys.

13. We can find out smallest element of the set of elements which are organised as binary tree by simply traversing the from using left link till the left most node and print the information in that node. Similarly, by traversing the tree from root using right link till we find right most and printing the info in that node we can get maximum.

14. The dynamic set operations SEARCH, MINIMUM, MAXIMUM, SUCCESSOR, PREDECESSOR can be made to run in O(log *n*) time on a binary search tree.

15. If a node in a binary search tree contains two children (leaf), then its successor has no left child and its predecessor has no right child.

16. The deletion operation on binary search trees is commutative. That is, first if we delete x and followed by y the resulting tree will be same as if we delete y first followed by x.

17. The number of trees which we can generate from n distinct keys is $\lfloor n(\text{factorial } n = n!) \rfloor$.

18. The following numbers are to be inserted into a empty binary search tree. Find out the odd one.

A 50, 70, 90, 65, 30, 20, 40

B. 50, 30, 20, 65, 90, 70, 40

C. 50, 70, 65, 90, 30, 40, 20

D. 50, 70, 30, 65, 40, 20, 90

Answer: D(This does not require any rotation)

19. Suppose that we have numbers between 1 and 1000 in a binary search tree and want to search for the number 363. Which of the following sequences could not be the sequence of nodes examined?

A. 2, 252, 401, 398, 330, 344, 397, 363

B. 924, 220, 911, 244, 898, 258, 363, 363

C. 925, 202, 911, 240, 912, 245, 363

D. 2, 399, 387, 219, 266, 382, 381, 278, 363

E. 935, 278, 347, 621, 299, 392, 358, 363

Answer: C and E

20. Internal path length is the sum of the levels of all the nodes in the tree. If we assume equal likelihood for accessing every node in the tree then the average no of comparisons needed equals (I+n)/n. where I is the internal path length.

21. External path length is the sum of the levels of all the external nodes of its extension. Then the average no of comparisons in unsuccessful search is equals to E/(n+1), where E is external path.

22. A binary search tree that minimizes the expected number of comparisons for a given set of keys and probabilities. Constructing optimal search trees is of complexity $O(n^2)$. However, this is very much depends on the keys distribution.

23. Split tree contains two keys, node key and split key. If during searching if match occurs with node key it will be stopped. Otherwise, using split key it is decided to move left or right. Median split tree is a variant of the split trees in which node key is set is most frequent among the keys in that subtree rooted at that node and the split key as median of the keys in that subtree. This can be constructed using O(nlog(n)). It requires always fewer comparisons than $\log^2 n$.

24. Max heap is the tree in which value of any node will be larger than its immediate children values. Where as the min heap is the tree in which value of any node will be smaller than its immediate children values.

25. Balanced or height balanced or AVL trees are the ones in which balance value (where balance is the difference of heights of left sub-tree and right sub-tree or vice versa) is either –1, 0, and 1. While binary trees are created through repeated insertions, deletions height balancing is done. This is to have uniform search cost when it is practically used in some applications.

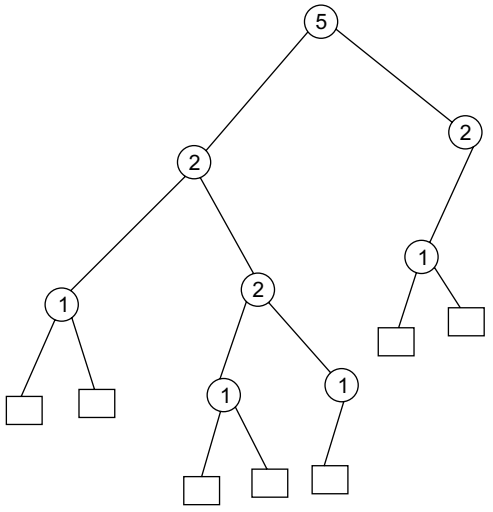
26. If we have a tree t, and if we have added a new leaf node at each NULL left and right pointers in the tree t, then the resulting tree can be called as extended binary tree which is strictly binary tree.

27. If a tree with n nodes is extended using the above approach and an extended tree is created then number of new nodes that will be added is n+1.

28. Braided binary search trees in which a linked list is threaded through the nodes in increasing order. The same tree can be called as braided search tree.

29. If we want to find k'th element in an ordered list we may need to visit k nodes. Thus, to reduce this complexity lists are represented as trees as shown below such that each leaf node is one node of the list. Non leaf node contains count known as leftcount which physically indicates the number of nodes of list left to it. The node structure has to be defined appropriately. Then by applying the following algorithm we can find the k'th node easily (assume H is the root node).

```
While( H is not leaf node)
  {
        if( H->linkcount >= k ) H=H->left;
        else
              {
                    k-=H->linkcount'
                    H=H->right
              }
  }
```



If there are N nodes and if they are organised as complete binary tree similar to above, then with $\log_2 N$ complexity we can find the list node of our interest.

30. Popularly, three traversals are used with trees. To name, In-Order (left-center-right), Pre-Order (center-left-right) and Post-Order (left-right-center). That is, while traversing the tree the above sequence is applied. Of course, traversing is meant for some operation such as searching the node information, etc. While doing this at every node this sequence is maintained. For example, In-Order means, at every node first we traverse left tree first, then the node and then its right tree. If we traverse a binary search tree in in-order fashion we will get all the node values ordered in ascending fashion.

31. Sequential representation of a tree means, storing tree nodes information an 1-D array fashion such that I'th node left, right children values are stored at 2*I+1, 2*I+2 elements of the array. Index of the root node is considered as 0. Sequential representation is employed to reduce time required to read/write tree information from disk to RAM in some applications. That is, if nodes are located sequentially in memory time required to write into a file or read from a file becomes less. Ofcourse, after reading into RAM by spending some more extra operations we can create tree (logical view) easily.

32. If there are N keys then if they are organised in complete binary tree fashion then it will be having minimum possible height.

33. If there are N keys then if they gets organised such that one node exactly contains one child (skewed tree) then we will have tree with worst possible height.

34. If there are N keys then if they gets organised such that one node exactly contains two children (i.e. As a strictly binary tree) then we will have tree with worst possible height as (N+1)/2 and the lowest level as N/2 (both are integer divisions). This situation arises if tree is simultaneously skewed and strictly binary. That is, at any level (other than lowest) only two nodes are available; for one of them two children are available and evidently the other one is leaf node. The number of non-leaf nodes for such a configuration is N/2 and leaf nodes will be N/2+1.

35. We can consider a 1-D array having elements in descending order as a max heap. Similarly, an a 1-D array having elements in ascending order as a min heap.

36. In-Order successor of a node (p) can be calculated by:

a. If p is left child to its parent and it does not have any right sub-tree then its parent itself becomes in-order successor for p.

b. If p is left child to its parent and it does have a right sub-tree then left most node of its right subtree becomes in-order successor for p.

c. If p is right child to its parent and it does not have any right sub-tree then back track using father field till we find a node (q) which is left to its parent or whose (q's) parent is null. Then return q's parent as in-order successor to p.

d. If p is right child to its parent and it does have a right sub-tree then left most node of its right subtree becomes in-order successor for p.

e. In-order successor of right most node of a binary tree is NULL.

37. Pre-Order successor of a node (p) can be calculated by:

a. If p is right child to its parent and it does not have any right sub-tree then back track using father field till we find a node (q) which is having right tree or whose (q's) parent is null. Then return q's right child as pre-order successor to p.

b. If p is left child to its parent and it does have a right sub-tree then its right child becomes pre-order successor for p.

c. If p is left child to its parent and it does not have any right sub-tree then back track using father field till we find a node (q) which is having right tree or whose (q's) parent is null. Then return q's right child as pre-order successor to p.

d. If p is right child to its parent and it does have a right sub-tree then its right child e becomes in-order successor for p.

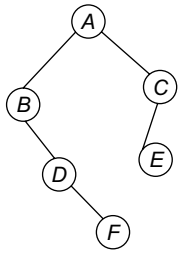
e. Pre-order successor of right most node of a binary tree is NULL.

38. Post-Order successor of a node (p) can be calculated by:

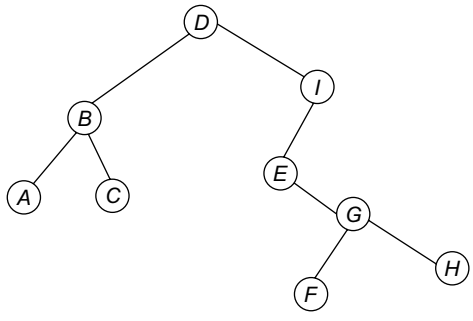
a. If p is left child to its parent and then return its parent as post order successor if it does not have any right sub tree. Otherwise, i.e if its parent has right sub tree then return find out left most node (q) of its right sub tree. If q is not having right sub tree then return q as post order successor to p else repeat the same operation on its right sub tree till we find a node which does not right child and return the same as post order successor to p.

b. If p is right child to its parent then return its parent as post-order successor for p.

39. The in-order, pre-order traversals of a tree are: "BDFAEC", "ABDFCE". Then the topology of the tree is:



40. The in-order, pre-order traversals of a tree are: "ABCDEFGHI", "DBACIEGFH". Then the topology of the tree is:



41. The height of the tree is four. (From Pre-order sequence, we know A is root node. By counting number of symbols, including A in in-order sequence we can calculate the left height and by counting symbols after A in in-order sequence; including A right height can be calculated. Maximum of them can be said as height of the tree).

42. If k is the smallest integer greater than or equal to $n+\log_2 n-2$, k comparisions are necessary and sufficient to find the largest and second largest elements of a set of n distinct elements.

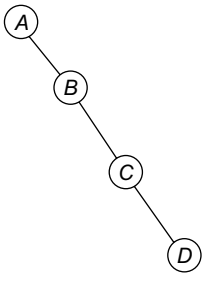
43. For any strictly binary the following relation holds.
No of Leaf Nodes = No of Non Leaf Nodes +1

44. Implicit array representation of the binary tree: Here, we assume root node information is stored in 0'th element of the array and any node p's left child information is stored at location 2p+1 and right child information at 2p+2 in the array. This is called as sequential representation of the tree. Here, if a element is at p then it can called as left child to its parent if p%2 is one else it is called as right child to its parent. Similarly, p/2 (integer division) gives index of its parent node. (Please note that this discussion is applicable to languages where array element indexes starts from 0, such as C language).

45. Leftmost node at level n in an almost complete strictly binary tree is assigned a number 2^n. Assuming root node level is 0 and its number is 1.

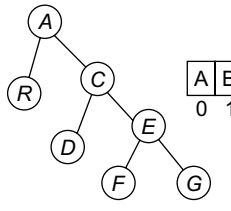
46. Number of array elements with null will be zero if all the N(where $N = 2^q-1$, for some integer value of q) elements of tree are represented in complete binary tree fashion.

47. If N (where $N = 2^q - 1$, for some integer value of q) keys are organised in a binary tree fashion such that there exists N-1 non leaf nodes and only one leaf node (as shown in the figure) then number of empty cells will be 2^N-1-N.



A		B				C							D
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

48. If N keys (where $N = 2^q - 1$ for some integer value of q) are organised in a binary tree fashion such that there exists N/2 non leaf nodes and N/2+1 leaf nodes (as shown in the figure) such that there exists two nodes at every level (one is leaf and the other is non leaf) and at the lowest level two leaf nodes are then, then number of empty cells will be $2^{N/2+1} - 1 - N$.



A	B	C	.	,	D	E							F	G
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

49. By employing stacks we can traverse trees without non recursively. The following C code gives how we can traverse tree using a stack in in-order.

```
p is the pointer to root of the tree.
        do{
/* travel down left branches as far as possible
and push nodes to stack */
   while (p!=NULL) { push(p);
                        p=p->left; }
         if(stack is not empty)
         {
                p=pop();
                print information of node p.
                p=p->right;
         }
} while( stack is not empty or p is not NULL);
```

50. To find whether two trees are topologically same or not we can use the above algorithm with little modification. That is, whenever we pop a node p, from stack we add L, R or N symbols to an output string depending on whether p is left or right child to its parent or whether p is root node respectively. If we apply this modified algorithm to both the trees and the resultant output strings are exactly same then we can say that both the trees are topologically same.

Similarly, if we want to find whether the second tree is flipped version of first tree then we can calculate output string of second tree and reverse the same while replacing L's with R's and R's with L's and compare with the output string of first tree traversal using the above modified algorithm. If they are matching then we can say they are mirrors to each other.

51. A tree is said to be right-in threaded tree if a node contains a pointer (right pointer) to its in-order successor. Having this link makes tree traversal easy.

52. A tree is said to be left-in threaded tree if a node contains a pointer (left pointer) to its in-order predecessor.

53. A tree is said to be in threaded tree is the one in which both right-in and left-in threaded.

54. A tree is said to be pre threaded tree in which NULL right and left pointers of nodes are replaced by their preorder successors and predecessors, respectively. This makes tree traversal easy using preorder.

55. A binary tree with n nodes uses 2n links, out of which n+1 links are NULL's, i.e unused. Threaded trees are

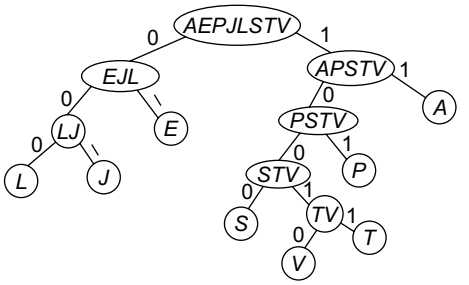
proposed to use these n+1 links to traverse the trees in a better manner.

56. Huffman Trees

These tree are used in compressing the data. For example, normally while storing characters in a text file every character takes 1 byte. In practice in a text file some characters occurs more frequently compared to other characters. Huffman coding technique is used to design codes for the symbols such that the code designed for characters which occurs more frequently will be having less code length compared to the infrequently occurring ones. Thus, the total number of bits required to store the file becomes less.

Essentially first all symbols are sorted in accordance with their frequency of occurrence. Then the last two probabilities are added and again the remaining probabilities and the new one are sorted. This repeated till we get single probability. That is all the characters are considered as one group. This picture is represented as a binary fashion in which each node will be having symbols which represents that probability. In order to find out the code of any symbol traverse from the root till we find a leaf node with that symbol. While traversing we output 1 if we move right, 0 if we move left. Thus, the resulting string becomes the designed code of that symbol.

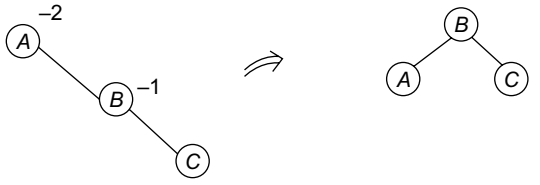
Symbol	Prob							
A	0.35	0.35	0.35	0.35	0.35	0.375	0.625	1
E	0.2	0.2	0.2	0.2	0.275	0.35	0.375	
P	0.15	0.15	0.15	0.175	0.2	0.275		
J	0.1	0.1	0.125	0.15	0.175			
L	0.075	0.075	0.1	0.125				
S	0.05	0.075	0.075					
T	0.05	0.05						
V	0.025							



In order to get log complexity in the worst case, the Binary Heap is organized as a **balanced tree** (virtually), while it is implemented (in reality) as an array.

57. Left Rotation about p, which is youngest ancestor which got unbalanced by inserting a new node. This is applied if new node is inserted as a right child to its parent and is also right to p.

```
q=p->right
hold=q->left
q->left=p
p->right=hold
```

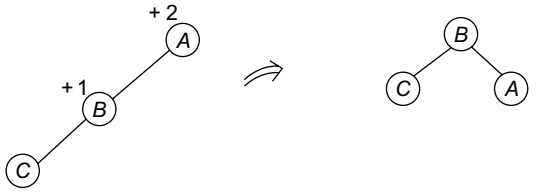


Note

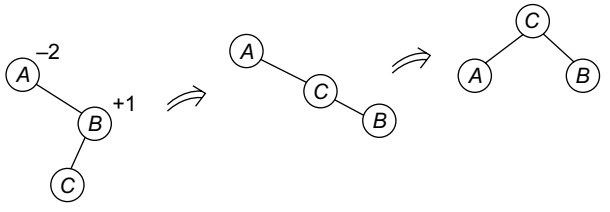
We assume that the node q will come in place of p of the original tree. This is assumed for all rotations.

58. Right Rotation about p which is youngest ancestor which got unbalanced by inserting a new node. This rotation is applied if new node is inserted as a left child to its parent and is also left to p.

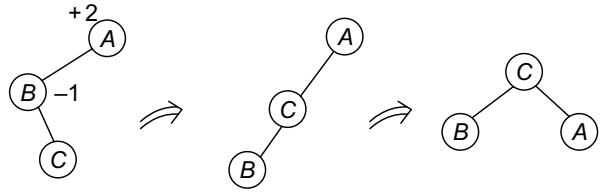
```
q=p->left
hold=q->rightt
q->right=p
p->left=hold
```



59. Left-Right Rotation about p, which is youngest ancestor which got unbalanced by inserting a new node. This is applied if new node is inserted as a right child to its parent and is also left to p. Here, first left rotation is applied about left child of p and then right rotation about p.

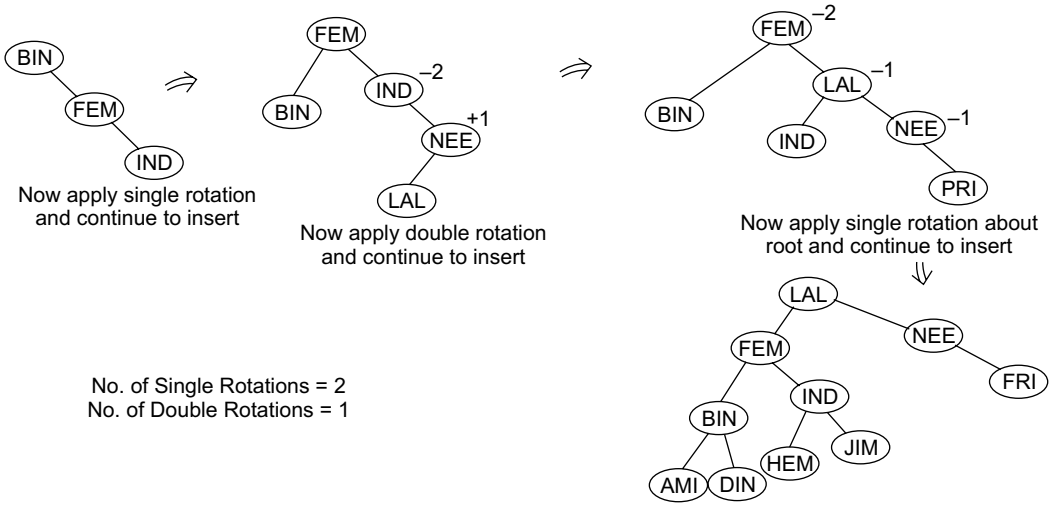


60. Right-Left Rotation about p, which is youngest ancestor which got unbalanced by inserting a new node. This is applied if new node is inserted as a left child to its parent and is also right to p. Here, first right rotation is applied about left child of p and then left rotation about p.

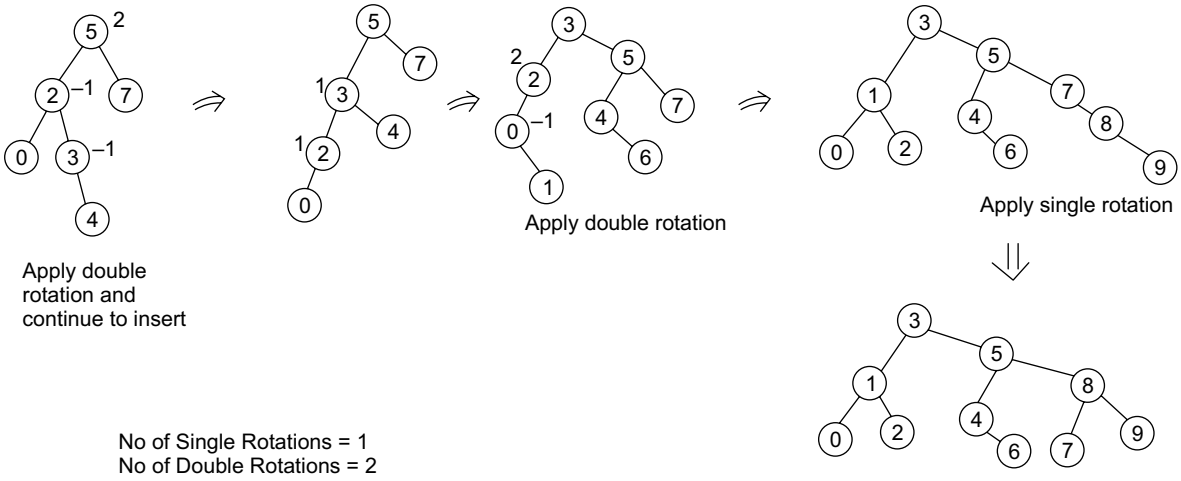


Examples of AVL tree creation (61– 63)

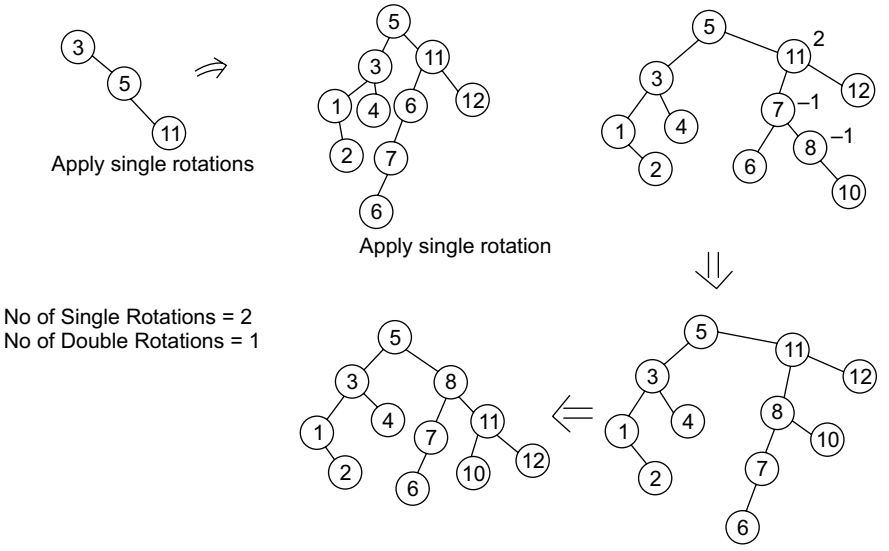
61. Create an AVL tree by inserting the following strings: BIN, FEM, IND, NEE, LAL, PRI, JIM, AMI, HEM, DIN. Assume tree is initially empty. Calculate number of single rotations, double rotations used.



62. Create an AVL tree by inserting the following numbers: 5, 2, 7, 0, 3, 4, 6, 1, 8, 9. Assume tree is initially empty. Calculate number of single rotations, double rotations used.



63. Create an AVL tree by inserting the following numbers: 3, 5, 11, 8, 4, 1, 12, 7, 2, 6, 10. Assume tree is initially empty. Calculate number of single rotations, double rotations used.



64. Complexity of an insertion operation in an AVL tree including balance adjustments is θ(log n). This is due to the fact that it takes θ(log n) time to add the new vertex, O(log n) time to retrace the path in order to find the youngest ancestor which gets unbalanced (which is also called as pivot) in addition to θ(1) time for rotation.

65. Re-balancing can be done in the case of single and double rotations with θ(1) time.

66. If keys 1, 2, 3,..2^k–1 are inserted in order into an empty AVL tree the resulting is perfectly balaced tree and needs about n/2 left rotations, where n is no of nodes.

67. AVL tree can be used to sort a sequence of n elements in O(n log n).

68. The first vertex encountered during the retracing step of the AVL tree insertion algorithm whose balance is ±1 is the only vertex that can possibly serve as the pivot.

69. Any arbitrary n-vertex binary search tree can be transformed into any other binary search tree on n vertices using O(n) basic rotation operations.

70. In the case of deletions more than one rotation. Worst case order is θ(log n).

a. If as a result of deleting a node the balance of the deleted nodes parent changes from 0 to ±1 then nothing has to be done to tree.

b. If as a result of deleting a node the balance of the deleted nodes parent changes from ±1 to 0, then re-balancing operation may or may not required to tree.

c. If as a result of deleting a node the balance of the deleted nodes parent changes from ±1 to ±2 then at least one rotation has to be done to tree.

71. The weight of a tree is defined as the number of external nodes in the tree (which is equal to number of null pointers in the tree). If the ratio of the weight of the left subtree of every node to the weight of the subtree rooted at the node is between some fraction a and 1–a, the tree is a weight-balanced tree of ratio a.

72. Red-Black Trees

A red-black tree is augmented binary search tree in which every vertex is coloured either red or black, and the arrangement of vertices obeys the following contraints.

a. (Black Rule) Every leaf is coloured black

b. (Red Rule) If a vertex is red, then both of its children are black.

c. (Path Rule) Every path from the root to a leaf contains the same number of black vertices.

73. In a red-black tree every path from a vertex to any descendent leaf must contain the same number of black vertices, and therefore every subtree in a red-black tree is itself a red-black tree.

74. Red rule implies that at least half of the vertices on any path from root to a leaf in red-black tree must be black. Therefore, if an n-vertex red-black tree has a height of h, its root must have a black-height of at least h/2.

75. A red-black tree with n internal nodes has a height of at most 2 log(n+1).

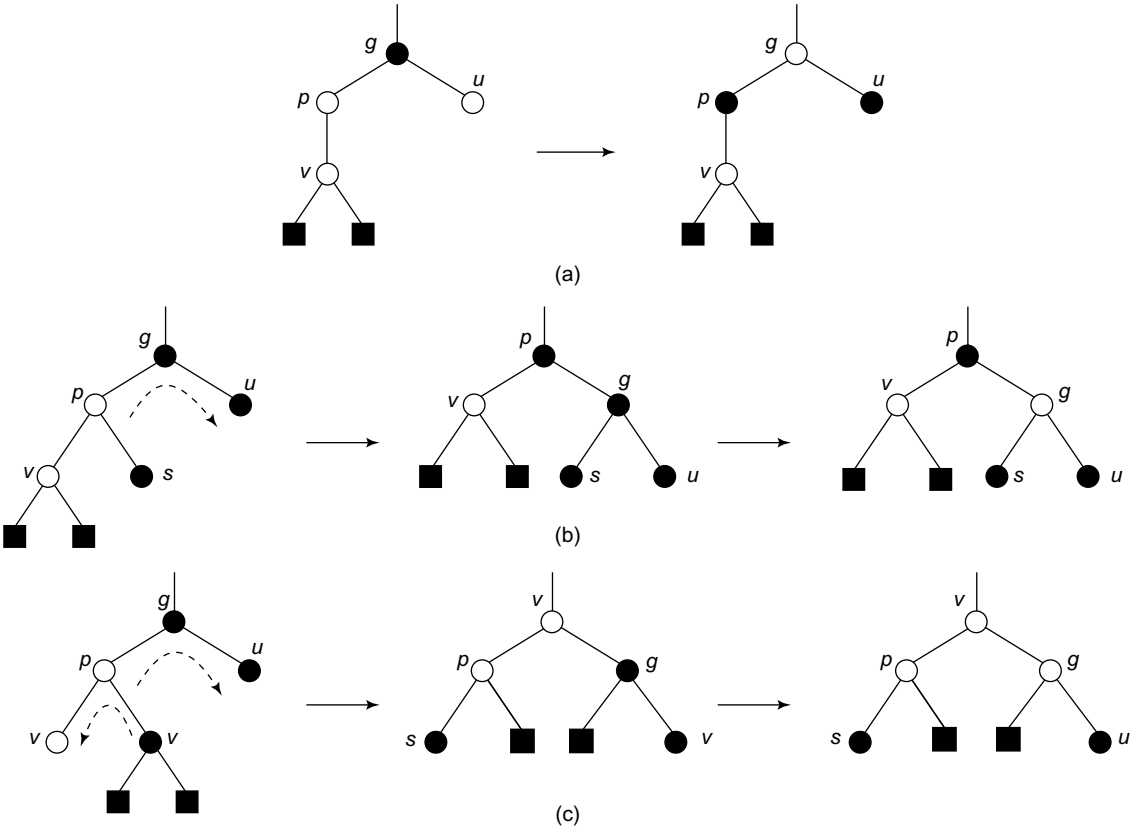
76. Node structure of a red-black tree is exactly same as normal binary tree with the exception that extra one bit is needed to encode color, i.e red or black.

77. Search time of red-black trees also Θ(log n).

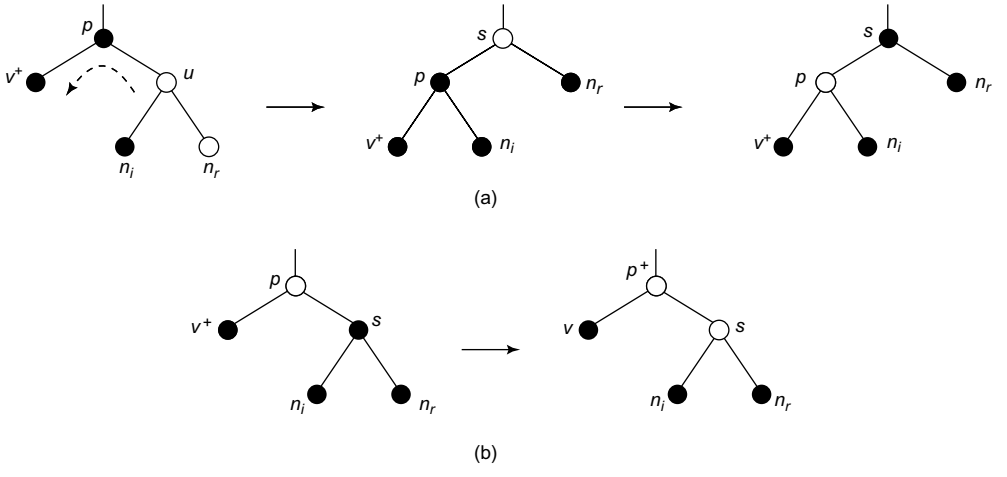
78. Normally insertion of a new key into red-black can be thought of as the replacement of an external black vertex with a subtree consisting of a red root and two black children. In fact, it may cause red rule to be violated while maintaining black rule. Thus, here also some rotations, re-colouring of nodes is needed. In some configurations, re-colouring may propagate till towards root.

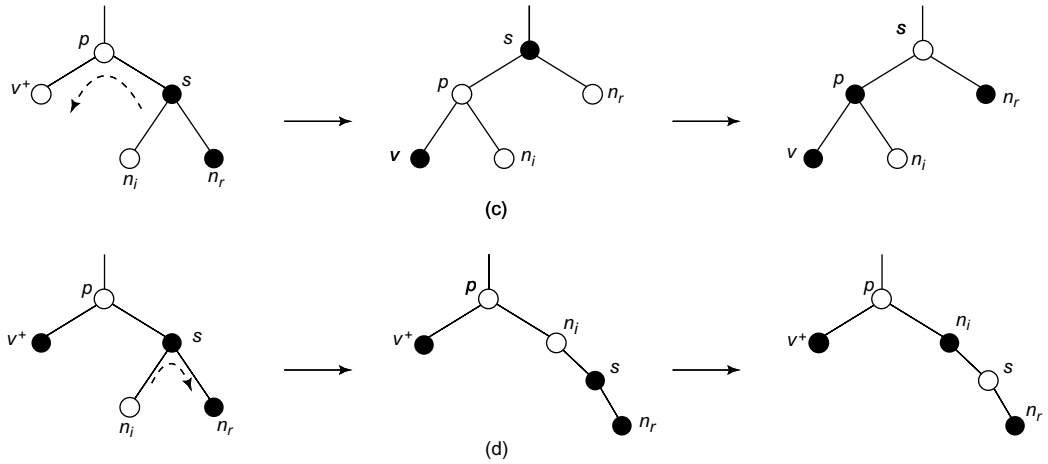
79. Every red-black tree of height h can be re-coloured so that there are exactly $\lfloor h / 2 \rfloor$ black vertices on every path from the root to a leaf, and that the root is black if and only if h is even.

80. Let us consider v is the inserted into red-black tree T, p is the parent of v, g is the grandparent of v and u is uncle of v, i.e p's sibling. Then the following situations arise while inserting the node and appropriate re-colouring, rotations are required to maintain the red-black tree structure.



81. Similarly deletions from red-black trees can be summarised as follows.





82. Worst case complexities of both deletion, insertions into red-black trees is Θ(log n).

■ **Example** Assume we have special tree say BEST in which root can have two children while every internal node is supposed to have exactly one child. Also, key values of the nodes are having relation as BST. Comment about such a tree with respect to BST.

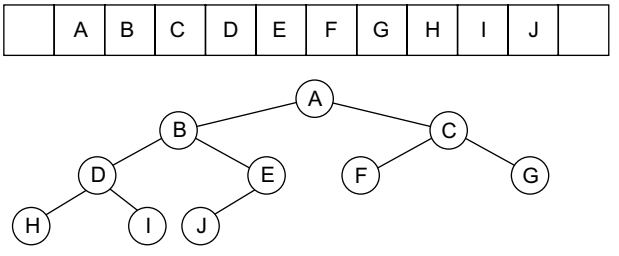
a. It can be realised by adjusting pointers of BST
b. It can be searched like BST.
c. Maximum possible height of BEST with N keys is N–1.
d. Best possible height is N/2 with N keys

2.4.5 Binary Heap

- A priority queue allows access to the min (or max) item
- The Binary Heap is a priority queue that allows insertion of new items and the deletion of the minimum item in logarithmic time.
- The Binary Heap is implemented by an array.
- To implement a priority queue we can use a linked-list with insertions on front done in constant time, and deletions in linear time, or vice versa. We implement a Binary Heap with a simple array, which supports insert and *deleteMin* in O(log N) in the worst case; it supports *insert* in O(c) in average case; it supports findMin in O(c) in worst case.

Virtual balanced tree structure of the Binary Heap

In order to get log complexity in the worst case, the Binary Heap is organised as a **balanced tree** (virtually), while it is implemented (in reality) as an array.



- The correspondence between an array and a balanced tree works if:
- the tree is a complete Binary Tree – all levels except the leaves must be filled there cannot be gaps in the nodes, including the leaves – J in the example must be a *left* child
- The height is at most [log N]

Implementation technique

- In a complete Binary Tree we do not need a left child and right child reference.
- This allows us to store each level in consecutive positions in an array.
- If we start the root node at position 1 rather than 0, then for every node in position *i* will have its left child in position *2i* and its right child in position *2i+1*.
- If either position *2i* or *2i+1* extends past the number of nodes, we know that that child does not exist.
- Inversely, given a node in position *i*, we know that its parent node will be in position *[i/2]*.
- The root node will have a virtual parent in position *[1/2]* = 0. Another reason to start the root at position *1*. In order to retrieve *min* or *max*, each parent node will be smaller than the children nodes.

Methods needed for Binary Heap

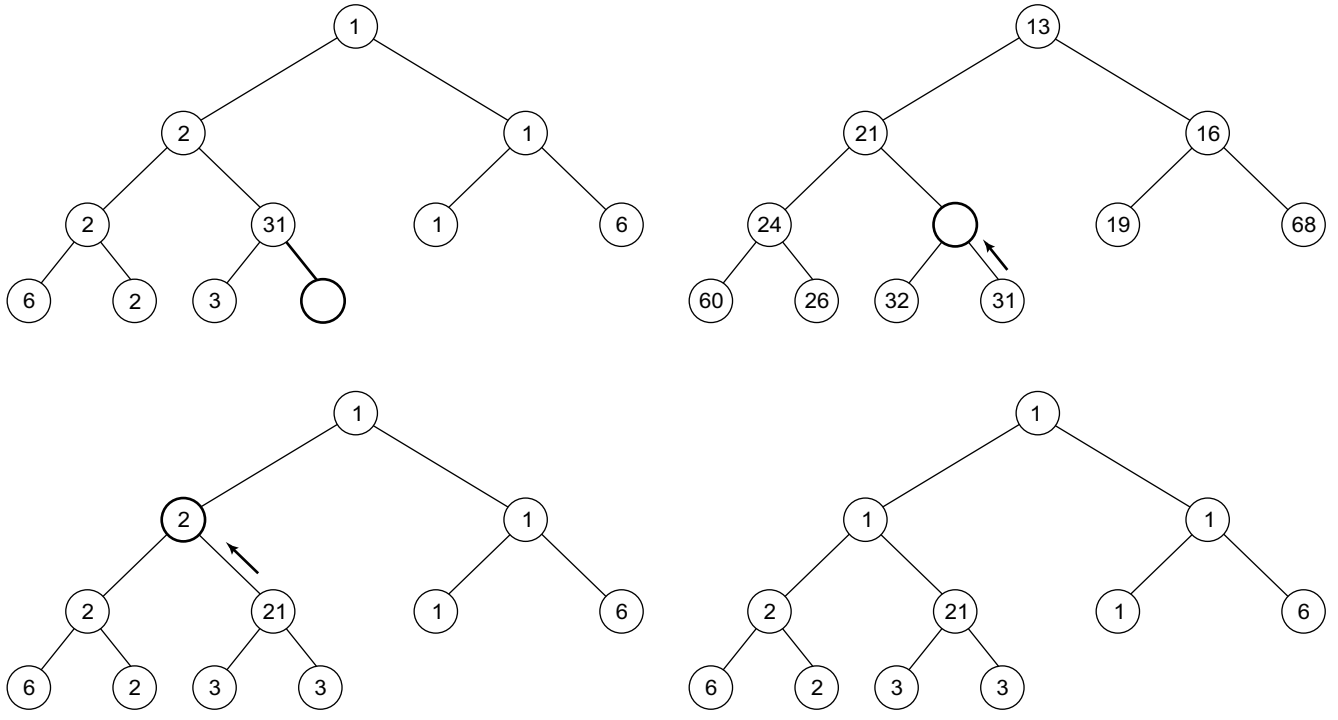
```
public class BinaryHeap implements PriorityQueue
{
          private int currentSize;
          private Comparable array [ ];
          private static final int DEFAULT_CAPACITY = … ;
  public BinaryHeap( Comparable negInf);
          public void insert( Comparable x);    // inserts in right order
          public void toss (Comparable x);      // inserts in any order
          public Comparable findMin( ) throws Underflow;
          public Comparable deleteMin( ) throws Underflow;
          public boolean isEmpty ( )
          {       return currentSize = = 0;     }
          public void makeEmpty ( )
          {       currentSize = 0;       }
          // if heap has (parent node > children nodes) after a toss
          private boolean orderOK;      // set to false when toss is performed
          private void getArray ( int newMaxSize );
          private void checkSize( );
// To insert a node in tree we find first hole & percolate up to keep order
          private void percolateDown ( int hole );
          private void fixHeap ( );             //reinstates heap order after toss
}
For many inserts it is cheaper to just toss an item and then reinstate order with fixHeap. This is done
in linear time.
To keep track of that, orderOK is set to false when a toss is performed.
public BinaryHeap ( Comparable negInf )
{
          currentSize = 0;
          orderOK = true;
          getArray (DEFAULT_CAPACITY);
```

```
            array[0] = negInf;
}
            public Comparable findMin ( ) throws Underflow
{
            if ( isEmpty ( ) )
                  throw new Underflow( "Empty binary heap" );
            if ( !orderOK)
            fixHeap ( );
            return array[1];
}
            private void getArray ( int newMaxSize)
{
            array new Comparable[ newMaxSize + 1];
}
```

Inserting

- Inserting an item into the heap implies adding a node to the tree.
- To preserve the completeness of the tree, we position ourselves at the next available location. If inserting there does not respect the parent-child order, we slide its parent to that position (hole).
- If parent is not in the right order, we slide ITS parent, and so on.
- Eventually all parents have slid into the right positions, leaving a hole.
- The node to be inserted takes that position, called the hole.
- This technique is called percolating up.
- This is done in logarithmic time.

View of inserting "14"



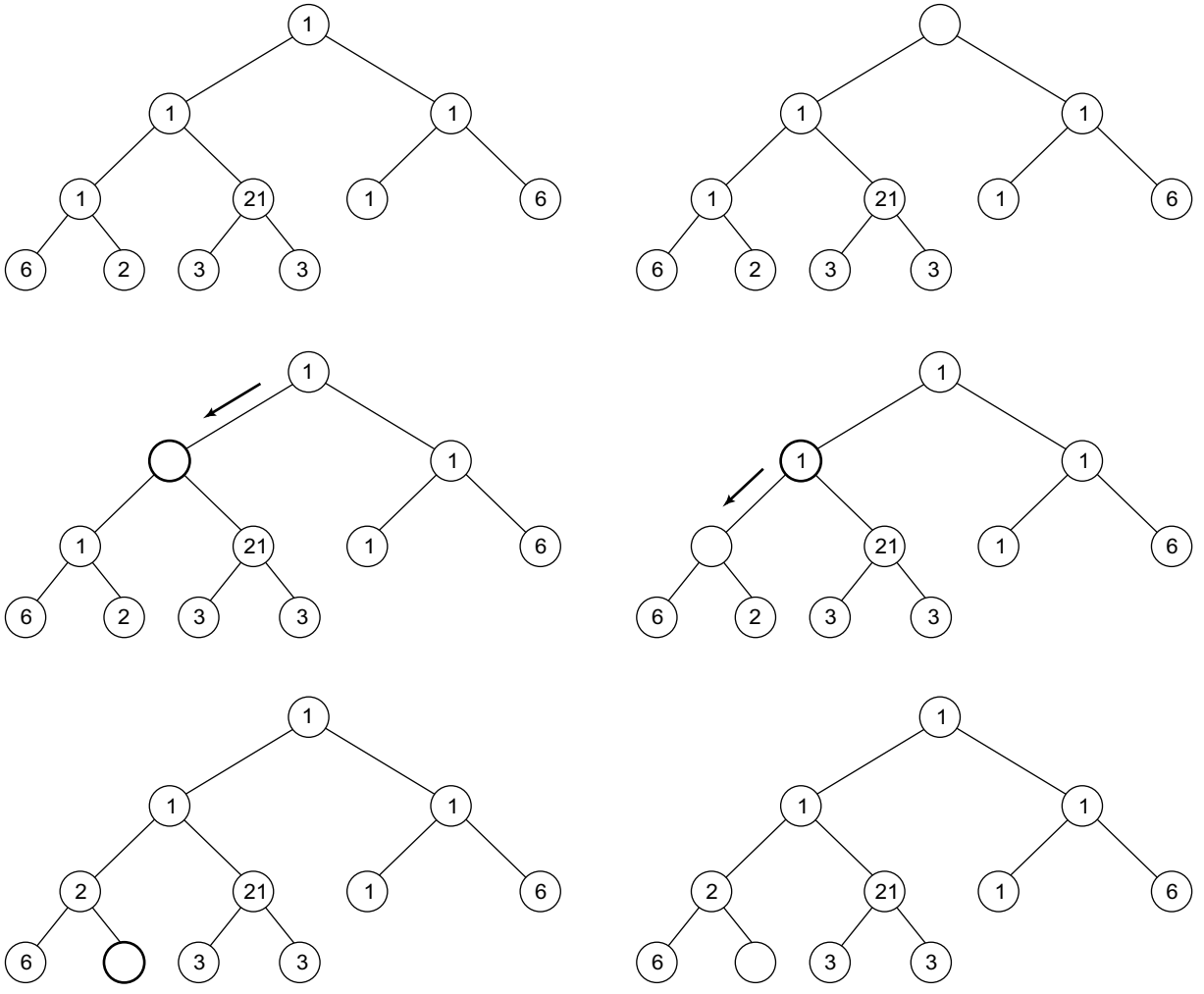
Code for insert

```
private void checkSize ( )
{
   if (currentSize = = array.size - 1)
   {
   Comparable oldarray[ ] = array;
   getArray( currentSize * 2);
   for (int i=0; i < oldArray.length; i++1)
          array[i] = oldArray[i];
   }
}
public void toss (Comparable x)
{
   checkSize( );
   array[++currentSize] = x;
   if (x.lessThan (array[currentSize/2]) )
          orderOK = false;
}
public void insert (Comparable x)
{
   if (!orderOK)
   {
          toss (x);
          return;
   }
   CheckSize ( );
// percolate up
   int hole = ++currentSize;
   for ( ; x.lessThan (array[hole/2]); hole /= 2 )
          array [hole] = array [hole/2];
   array [hole] = x;
}
```

Removing

- *deleteMin* creates the reverse problem from *insert*; it creates a hole and destroys the completeness of the tree. Since the min is the root of the tree, we have to *percolate down* the tree, moving the hole to the last leaf on the tree. This requires two methods, one to remove the smallest item at the root, the other to percolate the hole down to the last leaf.
- This is done in logarithmic time.

View of deleteMin



Code for remove

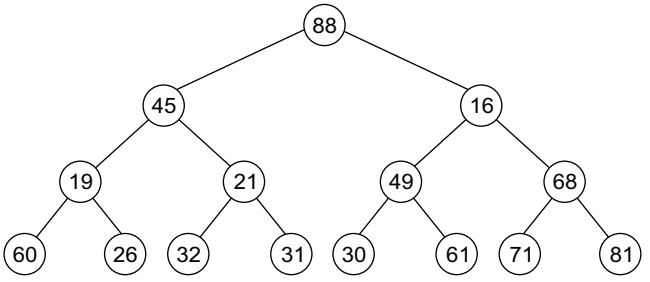
```
public Comparable deleteMin ( ) throws Underflow
{
   Comparable minItem = findMin ( );
   array [1] = array [currentSize --];
   percolateDown (1);
   return minItem;
}
private void percolateDown (int hole)
{
   int child;
   Comparable temp = array [hole];
   for ( ; hole * 2 <= currentSize; hole = child)
   {
          child = hole * 2;
          if (child != currentSize && array [child+1].lessThan (array [child] ) )
                 child++;
```

```
            if (array [child].lessThan (temp) )
                  array [hole] = array [child];
            else break;
      }
      array [hole] = temp;
   }
```

FixHeap

- An insertion can be done in O(log N)
- N insertions can be done in O(N logN)
- When N insertions must be done, it is better to *toss* the items into the heap and then fix the heap.
- We could exchange higher parents with smaller children, going recursively down left and right, using *percolate-Down.*
- It is cheaper to exchange from the lowest parents up. So we use reverse level order traversal.

View of fixHeap with reverse level traversal



- Check lowest parent on right, 68, and its children, 81 & 73. Order is OK.
- Check next lowest parent, 49, and its children, 61 and 30.
- Exchange nodes 49 and 30 and *percolateDown*.
- Check next lowest parent, 21, and its children, 32 and 31. Order is OK.
- Check next lowest parent, 19, and its children, 60 and 26. Order is OK.
- Check next lowest parent, 16, and its children, 30 and 68. Order is OK.
- Check next lowest parent, 45, and its children, 19 and 21. Exchange nodes 45 and 19 then *percolateDown*. Check 45 with 60 and 26. Exchange 45 and 26.
- Check next lowest parent, 88, and its children, 21 and 16. *percolateDown* and exchange nodes 88 and 21 then *percolateDown*.Exchange 88 and 21. *percolateDown* then exchange 88 and 45.

Code

```
private void fixHeap ( )
{
   for ( int i = currentSize/2;  i > 0 ;  i --)
   {       percolateDown (i);            }
   orderOK = true;
}
```

2.4.6 Graphs

A **graph** *G* consists of two types of elements, namely *vertices* and *edges*. Every edge has two *endpoints* in the set of vertices, and is said to **connect** or **join** the two endpoints. An edge can thus be defined as a set of two vertices (or an ordered pair, in the case of a **directed graph).**

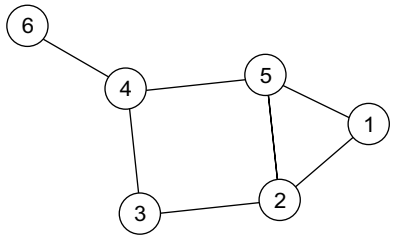


Figure 2.28 A sample graph

A **vertex** (basic element) is simply drawn as a *node* or a *dot*. The **vertex set** of G is usually denoted by $V(G)$, or V when there is no danger of confusion (see Fig. 2.28). The **order** of a graph is the number of its vertices, i.e. $|V(G)|$.

An **edge** (a set of two elements) is drawn as a *line* connecting two vertices, called **endvertices**, or *endpoints*. An edge with endvertices x and y is denoted by xy (without any symbol in between). The **edge set** of G is usually denoted by $E(G)$, or E when there is no danger of confusion.

The **size** of a graph is the number of its edges, i.e. $|E(G)|$.

A **loop** is an edge whose end vertices are the same vertex.

A **link** has two distinct end vertices. An edge is **multiple** if there is another edge with the same end vertices; otherwise it is **simple**. The **multiplicity of an edge** is the number of multiple edges sharing the same end vertices; the **multiplicity of a graph**, the maximum multiplicity of its edges. A graph is a **simple graph** if it has no multiple edges or loops, a **multigraph** if it has multiple edges, but no loops, and a **multigraph** or **pseudograph** if it contains both multiple edges and loops (the literature is highly inconsistent). When stated without any qualification, a graph is almost always assumed to be simple—one has to judge from the context.

The **complement** $\overline{G}$ of a graph G is a graph with the same vertex set as G but with an edge set such that xy is an edge in $\overline{G}$ if and only if xy is not an edge in G.

An **edgeless graph** or **empty graph** is a graph with zero or more vertices, but no edges.

Also, the **null graph** is the graph with no vertices and no edges. Or, it is a graph with no edges and any number n of vertices, in which case it may be called the **null graph on *n* vertices**.

A graph is **infinite** if it has infinitely many vertices or edges or both; otherwise the graph is **finite**. An infinite graph where every vertex has finite degree is called **locally finite**. When stated without any qualification, a graph is usually assumed to be finite.

A **walk** is an alternating sequence of vertices and edges, beginning and ending with a vertex, where each vertex is incident to both the edge that precedes it and the edge that follows it in the sequence, and where the vertices that precede and follow an edge are the end vertices of that edge. A walk is **closed** if its first and last vertices are the same, and **open** if they are different.

The **length** l of a walk is the number of edges that it uses. For an open walk, $l = n–1$, where n is the number of vertices visited (a vertex is counted each time it is visited). For a closed walk, $l = n$ (the start/end vertex is listed twice, but is not counted twice). In the example graph, (1, 2, 5, 1, 2, 3) is an open walk with length 5, and (4, 5, 2, 1, 5, 4) is a closed walk of length 5.

A **trail** is a walk in which all the edges are distinct. A closed trail has been called a **tour** or **circuit**, but these are not universal, and the latter is often reserved for a regular subgraph of degree two.

Traditionally, a **path** referred to what is now usually known as an *open walk*. Nowadays, when stated without any qualification, a path is usually understood to be **simple**, meaning that no vertices (and thus no edges) are repeated. (The term **chain** has also been used to refer to a walk in which all vertices and edges are distinct.) In the example graph, (5, 2, 1) is a path of length 2. The closed equivalent to this type of walk is called a **cycle**. Like *path*, this term traditionally referred to any closed walk, but now is usually understood to be simple by definition. In the example, graph, (1, 5, 2, 1) is a cycle of length 3. (A cycle, unlike a path, is not allowed to have length 0.) Paths and cycles of n vertices are often denoted by Pn and Cn, respectively.

A cycle that has odd length is an **odd cycle**; otherwise it is an **even cycle**. A graph is **acyclic** if it contains no cycles; **unicyclic** if it contains exactly one cycle; and **pancyclic** if it contains cycles of every possible length (from 3 to the order of the graph).

The **girth** of a graph is the length of a shortest (simple) cycle in the graph; and the **circumference**, the length of a longest (simple) cycle. The girth and circumference of an acyclic graph are defined to be *infinity* ∞.

A path or cycle is **Hamiltonian** (or *spanning*) if it uses all vertices exactly once. A graph that contains a Hamiltonian path is traceable; and one that contains a Hamiltonian path for any given pair of (distinct) end vertices is a *Hamiltonian connected graph*. A graph that contains a Hamiltonian cycle is a *Hamiltonian graph*.

A trail or circuit (or cycle) is **Eulerian** if it uses all edges precisely once. A graph that contains an Eulerian trail is **traversable**. A graph that contains an Eulerian circuit is an **Eulerian graph**.

Two paths are **internally disjoint** (some people call it *independent*) if they do not have any vertex in common, except the first and last ones.

The **degree**, or *valency*, $dG(v)$ of a vertex v in a graph G is the number of edges incident to v, with loops being counted twice. A vertex of degree 0 is an **isolated vertex**. A vertex of degree 1 is a leaf. In the example graph vertices 1 and 3 have a degree of 2, vertices 2,4 and 5 have a degree of 3 and vertex 6 has a degree of 1. If E is finite, then the total sum of vertex degrees is equal to twice the number of edges.

The **total degree** of a graph is equal to two times the number of edges, loops included. This means that for a graph with 3 vertices with each vertex having a degree of two (i.e. a triangle) the total degree would be six (e.g. $3 \times 2 = 6$). The general formula for this is **total degree = 2n** where **n = number of edges**.

A **degree sequence** is a list of degrees of a graph in non-increasing order (e.g. $d_1 \geq d_2 \geq \ldots \geq d^n$). A sequence of non-increasing integers is **realisable** if it is a degree sequence of some graph.

Two vertices u and v are called **adjacent** if an edge exists between them. We denote this by $u \sim v$ or $u \downarrow v$. In the above graph, vertices 1 and 2 are adjacent, but vertices 2 and 4 are not. The set of **neighbours** of v, that is, vertices adjacent to v not including v itself, forms an induced subgraph called the **(open) neighbourhood** of v and denoted $NG(v)$. When v is also included, it is called a **closed neighbourhood** and denoted by $NG[v]$. When stated without any qualification, a neighbourhood is assumed to be open. The subscript G is usually dropped when there is no danger of confusion; the same neighbourhood notation may also be used to refer to sets of adjacent vertices rather than the corresponding induced subgraphs. In the example graph, vertex 1 has two neighbours: vertices 2 and 5. For a simple graph, the number of neighbours that a vertex has coincides with its degree.

A **dominating set** of a graph is a vertex subset whose closed neighbourhood includes all vertices of the graph. A vertex v **dominates** another vertex u if there is an edge from v to u. A vertex subset V **dominates** another vertex subset U if every vertex in U is adjacent to some vertex in V. The minimum size of a dominating set is the **domination number** $\gamma(G)$.

The **distance** $dG(u, v)$ between two (not necessary distinct) vertices u and v in a graph G is the length of a shortest path between them. The subscript G is usually dropped when there is no danger of confusion. When u and v are identical, their distance is 0. When u and v are unreachable from each other, their distance is defined to be infinity ∞.

A graph is connected if there exists a path (of any length) from every node to every other node. The longest possible path between any two points in a connected graph is $n-1$, where n is the number of nodes in the graph.
A node is reachable from another node if there exists a path of any length from one to the other.

A *connected component* is a maximal subgraph in which all nodes are reachable from every other. Maximal means that it is the largest possible subgraph: you could not find another node anywhere in the graph such that it could be added to the subgraph and all the nodes in the subgraph would still be connected.

For directed graphs, there exist strong components and weak components. A strong component is a maximal subgraph in which there is a path from every point to every point following all the arcs in the direction they are pointing. A weak component is a maximal subgraph which would be connected if we ignored the direction of the arcs.

A cutpoint is a vertex whose removal from the graph increases the number of components. That is, it makes some points unreachable from some others. It disconnects the graph.

A cutset is a collection of points whose removal increases the number of components in a graph. A minimum weight cutset consists of the smallest set of points that must be removed to disconnect a graph. The number of points in a minimum weight cutset is called the point connectivity of a graph. If a graph has a cutpoint, the connectivity of the graph is 1. The minimum number of points separating two nonadjacent points, s and t is also the maximum number of point-disjoint paths between s and t.

A bridge is an edge whose removal from a graph increases the number of components (disconnects the graph). An edge cutset is a collection of edges whose removal disconnects a graph. A local bridge of degree k is an edge whose removal causes the distance between the endpoints of the edge to be at least k. The edge-connectivity of a graph is the minimum number of lines whose removal would disconnect the graph. The minimum number of edges separating two nonadjacent points, s and t is also the maximum number of edge-disjoint paths between s and t.

A graph is said to be weighted if its edges are having some numbers associated with them. The weights can be distances, time delays, or any thing. For example, see Fig. 2.29.

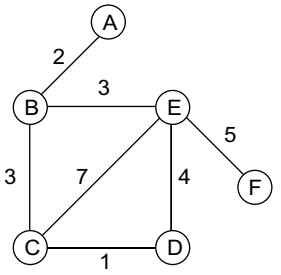


Figure 2.29 A sample weighted graph

List structures

- **Incidence list** : The edges are represented by an array containing pairs (ordered if directed) of vertices (that the edge connects) and eventually weight and other data.
- **Adjacency list :** Much like the incidence list, each vertex has a list of which vertices it is adjacent to. This causes redundancy in an undirected graph: for example, if vertices A and B are adjacent, A's adjacency list contains B, while B's list contains A. Adjacency queries are faster, at the cost of extra storage space.

Matrix structures

- **Incidence matrix :** The graph is represented by a matrix of E (edges) by V (vertices), where [edge, vertex] contains the edge's data (simplest case: 1 - connected, 0 – not connected).
- **Adjacency matrix :** there is an N by N matrix, where N is the number of vertices in the graph. If there is an edge from some vertex x to some vertex y, then the element M[x][y] is 1, otherwise it is 0. This makes it easier to find subgraphs, and to reverse graphs if needed.
- **Laplacian matrix** or **Kirchhoff matrix** or Admittance matrix : is defined as degree matrix minus adjacency matri*x* and thus contains adjacency information and degree information about the vertices
- **Distance matrix** : A symmetric N by N matrix an element M[x][y] of which is the length of shortest path between x and y; if there is no such path M[x][y] = infinity. It can be derived from powers of the **Adjacency matrix.**
- **Path matrix or Accessibility or Reachability Matrix:** An NxN binary matrix which indicates the reachability of (existence of route) from one node to another node. It does not contains any information about number of jumps (or hops) or distance between the stations. It only indicates the whether there is a route or not between two nodes.

Graph Representations

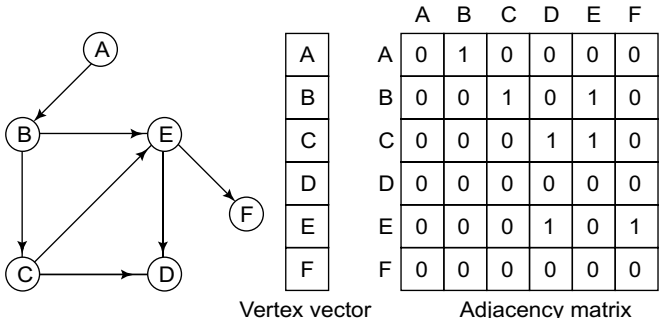
Graphs are represented graphically by drawing a dot for every vertex (or a un-filled circle), and drawing an arc between two vertices if they are connected by an edge. If the graph is directed, the direction is indicated by drawing an arrow.
In order to analyse graph information, we need to represent the same in a manner suitable for manipulation and storage. There are different ways to store graphs in a computer system. Evidently, two approaches, adjacency matrix approach and adjacency list approach, are in wide use. Theoretically one can distinguish between list and matrix structures but in concrete applications the best structure is often a combination of both. List structures are often preferred for sparse graphs as they have smaller memory requirements. Matrix structures on the other hand provide faster access but can consume huge amounts of memory if the graph is very large.

Adjacency Matrix Representation of the Graph

Here, graph topology information is stored in an matrix known as adjacency matrix which is an NxN matrix with its elements representing the number of edges between one station to another station (or between the nodes). None of its elements will be negative. If there is no route or edge between two stations, then the respective entry in this matrix will be zero. Also, if there exists more than n edges between two stations then the respective entry of this matrix is n. This matrix retains all the topological information of the graph. We can draw the graph easily if this matrix is known to us.

Figure 2.30 contains a sample graph and its adjacency matrix. Observe the principal diagonal elements values. All are zeros. This indicates that the stations are not having self cycles. For example, if we consider the graph shown is a road net-

work, the nodes are places, and edges are roads connecting the places then by chance a place (such as big city Hyderabad) is having a ring road then the respective diagonal element will become 1.

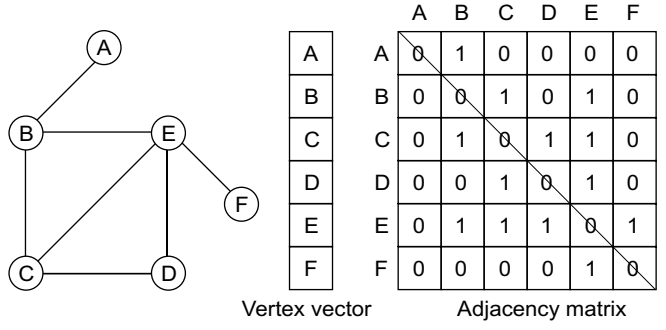


	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	1	0	1	0
C	0	0	0	1	1	0
D	0	0	0	0	0	0
E	0	0	0	1	0	1
F	0	0	0	0	0	0

Figure 2.30 A sample graph with its adjacency matrix

If we observe the last row of the adjacency matrix, we may find all 0's indicating that station is sink. Which means, there are no edges emanating from that station.

If the above graph is undirected graph, the adjacency matrix will be created by considering each undirected edge as two directed edges, one forward and one backward. The resultant adjacency matrix will be given in Fig. 2.31. However, for a weighted graph, the adjacency matrix will be having weight values of the edges.



	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	1	0	1	0
C	0	1	0	1	1	0
D	0	0	1	0	1	0
E	0	1	1	1	0	1
F	0	0	0	0	1	0

Figure 2.31 Adjacency matrix for an un-directed graph

If we observe the adjacency matrix of undirected graph, we may find it as symmetric matrix. Of course, one has to remember that a graph is called as directed graph (or di-graph) even if a single edge is directed. On the contrary, to call as an un-directed graph, all the edges should be un-directed. Of course, there are some algorithms which works only on directed graphs. In order to apply those algorithms on undirected graphs, we can replace each un-directed edge with two directed edges, one forward and one backward as shown in Fig. 2.32.

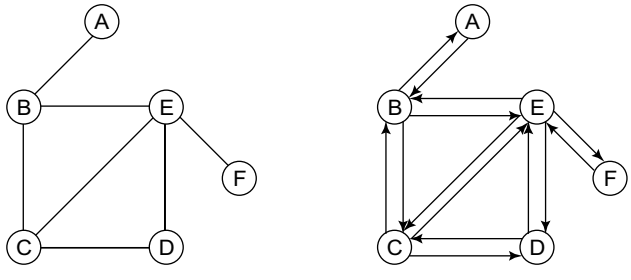
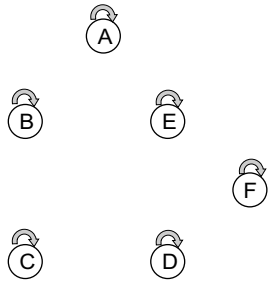


Figure 2.32 Converting an un-directed graph to directed graph

As mentioned in the above paragraph that the adjacency matrix of an un-directed graph is symmetric about its principal (or main) diagonal. In un-directed graphs, self loops are not allowed. Thus, the adjacency matrix contains its main diagonal elements as zeros.

Sum of the elements of an adjacency matrix of an un-directed graph will be even. That is, the sum of the degrees of all the vertices of an un-directed graph G is equal to twice the number of edges in G.

If an adjacency matrix of a graph (with N vertices) contains only 1's in main diagonal and all remaining elements are 0s, then we can say that the graph contains N connected components are N isolated points with self cycles.

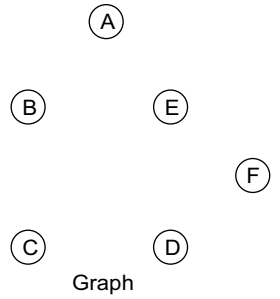


Graph

	A	B	C	D	E	F
A	1	0	0	0	0	0
B	0	1	0	0	0	0
C	0	0	1	0	0	0
D	0	0	0	1	0	0
E	0	0	0	0	1	0
F	0	0	0	0	0	1

Adjacency matrix

If an adjacency matrix of a graph (with N vertices) contains all 0s, then we can say that the graph contains N connected components are N isolated points as shown in Fig. 2.33.



Graph

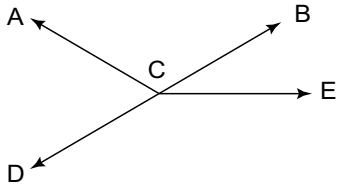
	A	B	C	D	E	F
A	0	0	0	0	0	0
B	0	0	0	0	0	0
C	0	0	0	0	0	0
D	0	0	0	0	0	0
E	0	0	0	0	0	0
F	0	0	0	0	0	0

Adjacency matrix

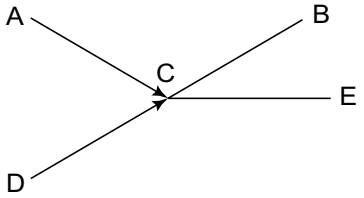
Figure 2.33 Graph and Adjacency matrix

An Adjacency matrix of a graph with N nodes contains all 1's in ith row (except ith element) and all 0's in the ith column, then ith station can be said as source. Similarly, An Adjacency matrix of a graph with N nodes contains all 0's in ith row (except ith element) and all 1's in the ith column, then ith station can be said as sink.

An Adjacency matrix of a graph with N nodes contains all 1's in ith row (except ith element) and all 0's in the remaining portion of the matrix, then the graph can be said as star shaped with ith station as center and directed edges for all other stations as shown here.



An Adjacency matrix of a graph with N nodes contains all 1's in ith row (except ith element), all 1's in ith column (except ith element), and all 0's in the remaining portion of the matrix, then the graph can be said as star shaped with ith station as center and un-directed edges for all other stations as shown here.



If all the stations are connected with exactly one directed edge and all are in a cycle as shown in Fig. 2.34 then the related adjacency matrix looks as shown in Fig. 2.34.

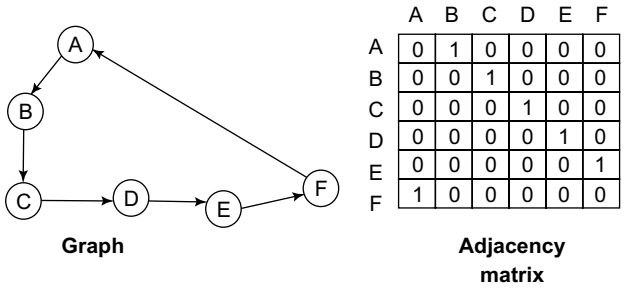


Figure 2.34 Adjacency matrix of circular graph

If the directed edges of the above graph are replaced with un-directed edges, then the adjacency matrix looks like (Fig. 2.35)

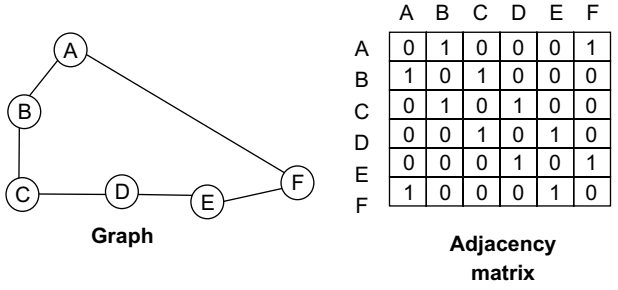


Figure 2.35 Adjacency matrix of a graph with single cycle and un-directed edges

What will be the nature of stations if the adjacency matrix looks like a band matrix of width 3 as shown is figure. All the stations will be having self loops and all are connected through un-directed edges except last two stations (see Fig. 2.36)

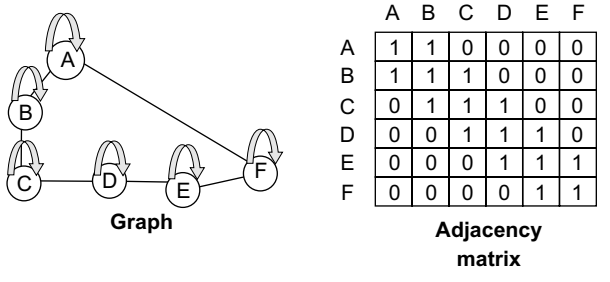


Figure 2.36 Adjacency matrix of a graph in which all the nodes are in a circular chain with self cycles

Adjacency matrix of a weighted graph is shown in Figure Fig. 2.37.

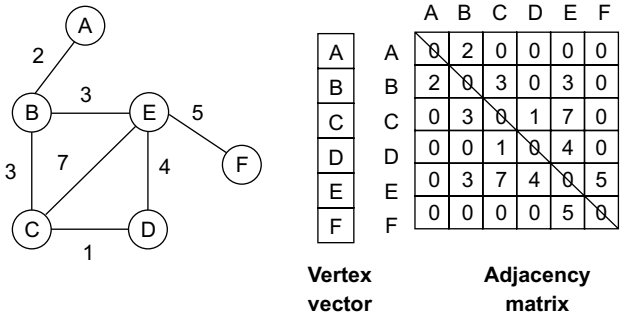


Figure 2.37 Adjacency matrix of a weighted graph

Adjacency List Representation

Graph information is maintained in another format known as adjacency list representation. Here, for each node, its neighbour's information is maintained in a linked list fashion as shown in Fig. 2.38.

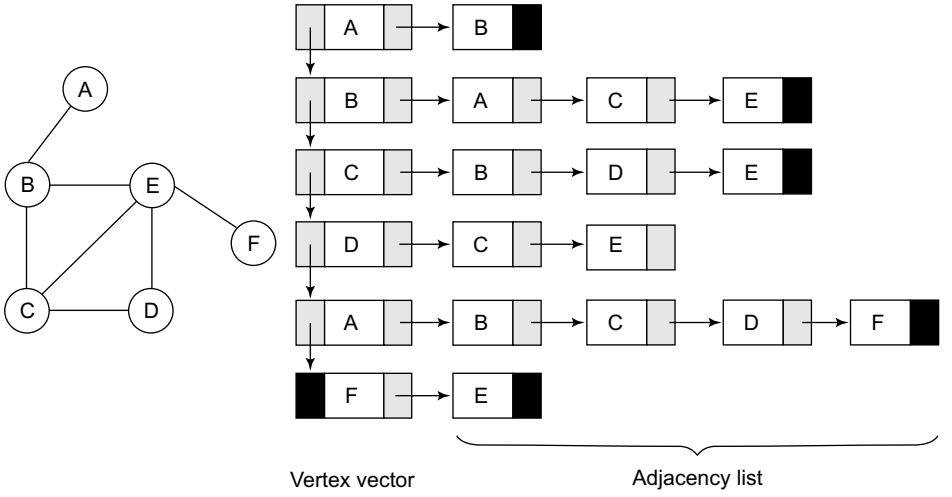
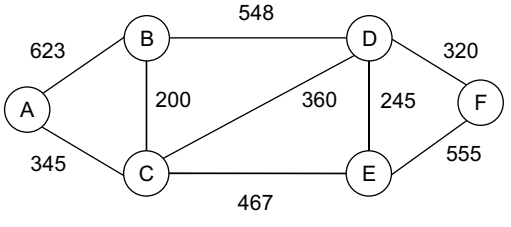


Figure 2.38 Adjacency list representation of the graph

Obviously, people may be getting doubt about which representation is better. The answer is, in practical applications combination of both is used. Thus, we really do not want to enter in to the debate of which is better. However, for the sake of comparison, we shall discuss about their relative merits and demerits in the following table.

Theme	**Adjacency list**	**Adjacency Matrix**
Memory requirement	O(|V|+|E|), where V, E are number of vertices and edges.	$O(V^2)$, V is number of vertices.
Insertions/deletions	Easy	Difficult
Is memory requirement constant assuming vertices are fixed?	No. Varies if extra edges are added.	Yes.
When it is preferred?	If the graph is sparse.	If the graph is dense.
Programmability?	Difficult.	Ease as matrix operations can be used.

Figure 2.38 displays the adjacency matrix and adjacency list of a sample weighted graph.



	A	B	C	D	E	F
A	0	623	345	0	0	0
B	623	0	200	548	0	0
C	345	200	0	360	467	0
D	0	548	360	0	245	320
E	0	0	467	245	0	555
F	0	0	0	320	555	0

Adjacency matrix

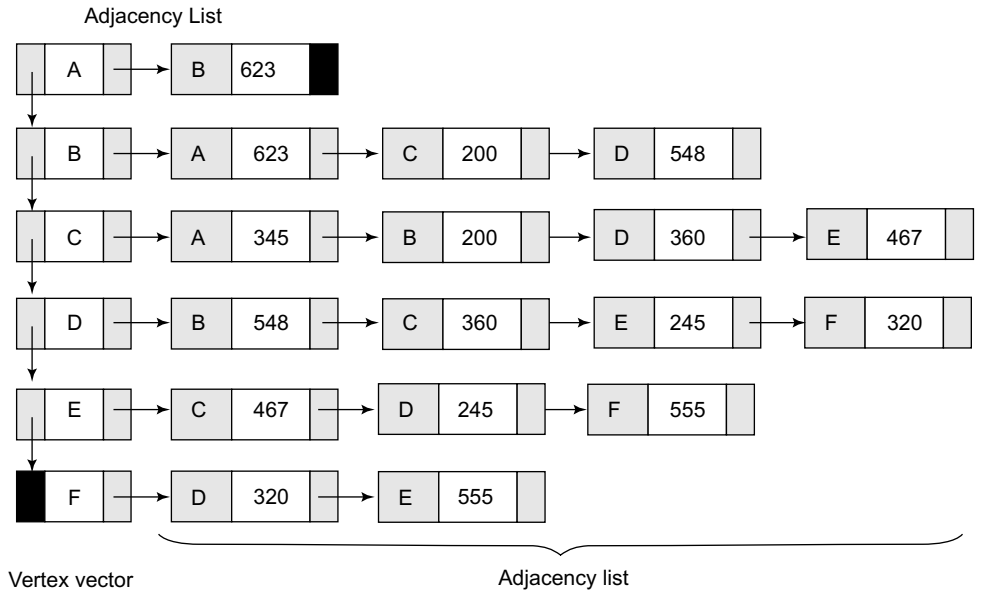
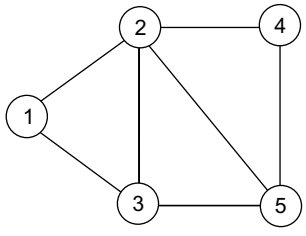


Figure 2.39 Adjacency Matrix and Adjacency List of a weighted matrix

Transitive Closure and Path Matrix

We have already understood that the adjacency matrix of the graph G is defined as A[J, K] = 1 if and only if J is adjacent to K; otherwise 0. The transitive closure of the graph G is defined as T[J, K] = 1 if and only if there is a nontrivial directed path from J to K; otherwise 0. That is, we are not interested in the number of edges in the path, but only in the existence of the path. The transitive closure of a graph is built on the set of vertices with the binary relation being adjacency.

Consider the following sample undirected graph.



The adjacency matrix A is:

0	1	1	0	0
1	0	1	1	1
1	1	0	0	1
0	1	0	0	1
0	1	1	1	0

Now, A^2 can be given as:

2	1	1	1	2
1	4	2	1	2
1	2	3	2	1
1	1	2	2	1
2	2	1	1	3

What is the physical significance of the above matrix? For example, station 1-5, the value is given as 2. This indicates the existence of two routes between 1 to 5 with 2 hops or jumps. We can verify the same as: 1-2-5 and 1-3-5. Similarly, we have 2-2 as 4. The possible routes are: 2-1-2, 2-3-2, 2-4-2, and 2-5-2 with two jumps. Like this, this matrix indicates the

number of paths with two jumps. Similarly, A^3 will indicate the possible routes from one station to another station with 3 jumps or hops. If we add A, A^2 and A^3 then the resultant matrix indicates the existence of routes between stations with at most 3 jumps.

We know that any path length can not be more than N edges in a graph with N nodes. Thus, we can calculate the path matrix or transitive closure by binarising the summation matrix of $A + A^2 + A^3 + ... + A^N$.

That is,

$X = A+A^2+A^3+..+A^N$

$$P[i][j] = \begin{cases} 1 & \text{if } X[i][j] > 0 \\ 0 & \text{otherwise} \end{cases}$$

This approach of calculating, $A, A^2, A^3, .., A^N$ is computationally demanding. We know that matrix multiplication algorithm has time complexity of $O(N^3)$. Thus, this method of finding transitive closure will be having order of complexity as $O(N^4)$.

■ **Exercise** Assuming A as adjacency matrix of a star shaped directed graph with N vertices (with one center vertex), what will be the matrices $A^2, A^3, .., A^N$?.

■ **Exercise** Assuming A as adjacency matrix of a star shaped un-directed graph with N vertices (with one center vertex), what will be the path matrix?.

Warshall's Algorithm

The basis of this algorithm is a very simple theme. Let, we have three stations x,y and z. There exists a path between x to y and y to z. Which means, we have the path for x to z also. Thus, in this algorithm, we try to explore the existence path between stations by explore existing paths of the stations. If we do not have path currently from station I to J. Then, we search whether there is a path between I to K and K to J, If exists, we consider that there is path between I to J. We will vary K such that all the stations are verified by assuming them as intermediate stations between I and J. However, in order to alleviate unnecessary computations, if already there exist a path between I and J, we don't explore further with intermediate stations.

Pseudo-Code of Warshalls Algorithm

Input: A – the N-by-N adjacency matrix of the graph.

Output: R – the N-by-N transitive closure of the graph.

```
For I = 1 to N Do
    For J = 1 to N Do
       R[I, J] = A[I, J]
    End Do
End Do
For K = 1 to N Do
    For I = 1 to N Do
       For J = 1 to N Do
         If (R[I,J]==0)  Then R[I,J] = R[I,K] ∧  R[K,J]
       End Do
    End Do
End Do
Return R
```

This is algorithm is computationally less intensive. Its order of complexity is $O(N^3)$. Moreover, the operations involved here are simple AND operators.

Graph Traversals

Like tree traversals discussed in the previous chapters, we do have some methods to traverse the graphs. They are:

- Depth First Traversal
- Breadth First Traversal

The basic object of the traversal is to visit each node at least once. However, the algorithm will actually give us much more than that.

The DFT algorithm is quite famous and can be used to solve a variety of graph problems. We employ stacks to traverse the graph in DF manner. As the algorithm proceeds, nodes status will be changed. Initially, we assume all the nodes will be in un-processed state. When they are in stack, we assume they are in the ready state. When they leave the stack, we consider them that they are processed.

Phase	Condition of Node	Status
1	Undiscovered	Un-Processed
2	Discovered and being processed	Ready
3	Finished	Processed

DFT Algorithm

1. Select any node of the graph and push into stack while making its status as Ready.
2. Repeat step 3 till stack becomes empty.
3. Pop a node (A) from the stack and make its status as processed. Push all the un-processed nodes of A into stack while making their status values as ready.

■ **Example** We apply the above algorithm for the graph shown in figure given below and show the stack changes.

1. We assume here that we will be traversing from node A. We will push the same into stack.
2. When we pop, we will get A as output. The same will be displayed in the output string. Its unprocessed neighbour, X, is pushed into stack.

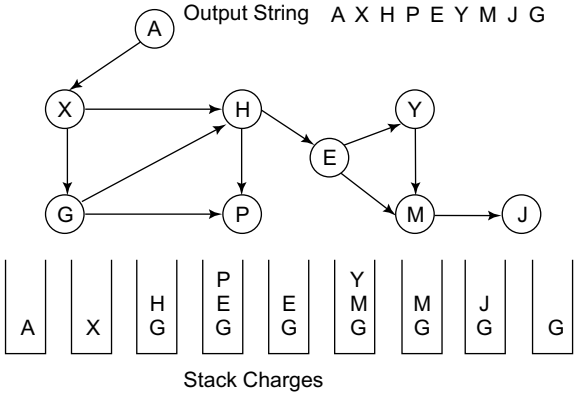


Figure 2.40 Depth First Traversal snap shot on a selected graph

3. Now, when we pop, we will get X. The same will be printed and its unprocessed neighbours, G and H are pushed into stack.
4. Now, when we pop, we will get H. The same will be printed and its unprocessed neighbour, E,P are pushed into stack.
5. When we pop, we will get P as output. The same will be printed. As it does not have any unprocessed neighbours, nothing is pushed into stack.
6. This time, when we pop, we will get E. The same will be printed and its unprocessed neighbours, M,Y are pushed into stack.
7. Now, when we pop, we will get Y. The same will be printed. Though it has a neighbour M, as M is already in the stack (i.e. its status is ready), nothing will be pushed into stack.
8. Next when we pop, we will get M as output. The same will be printed. Its unprocessed neighbour is pushed into stack.
9. Next, when we pop, we will get J as output. The same will be printed. As it does not have any neighbours, nothing will be pushed into stack.

10. Next when we pop, we will get G as output and the same will be printed. As G does not have unprocessed neighbours, nothing will be pushed into stack. Remember, though G as neighbours, all of them are in processed state (i.e. they went to stack and came out. That is, they are processed).
11. Now, stack empty. Thus, it terminates.

    One useful aspect of the DFS algorithm is that it traverses connected components one at a time, and thus it can be used to identify the connected components in a given graph.

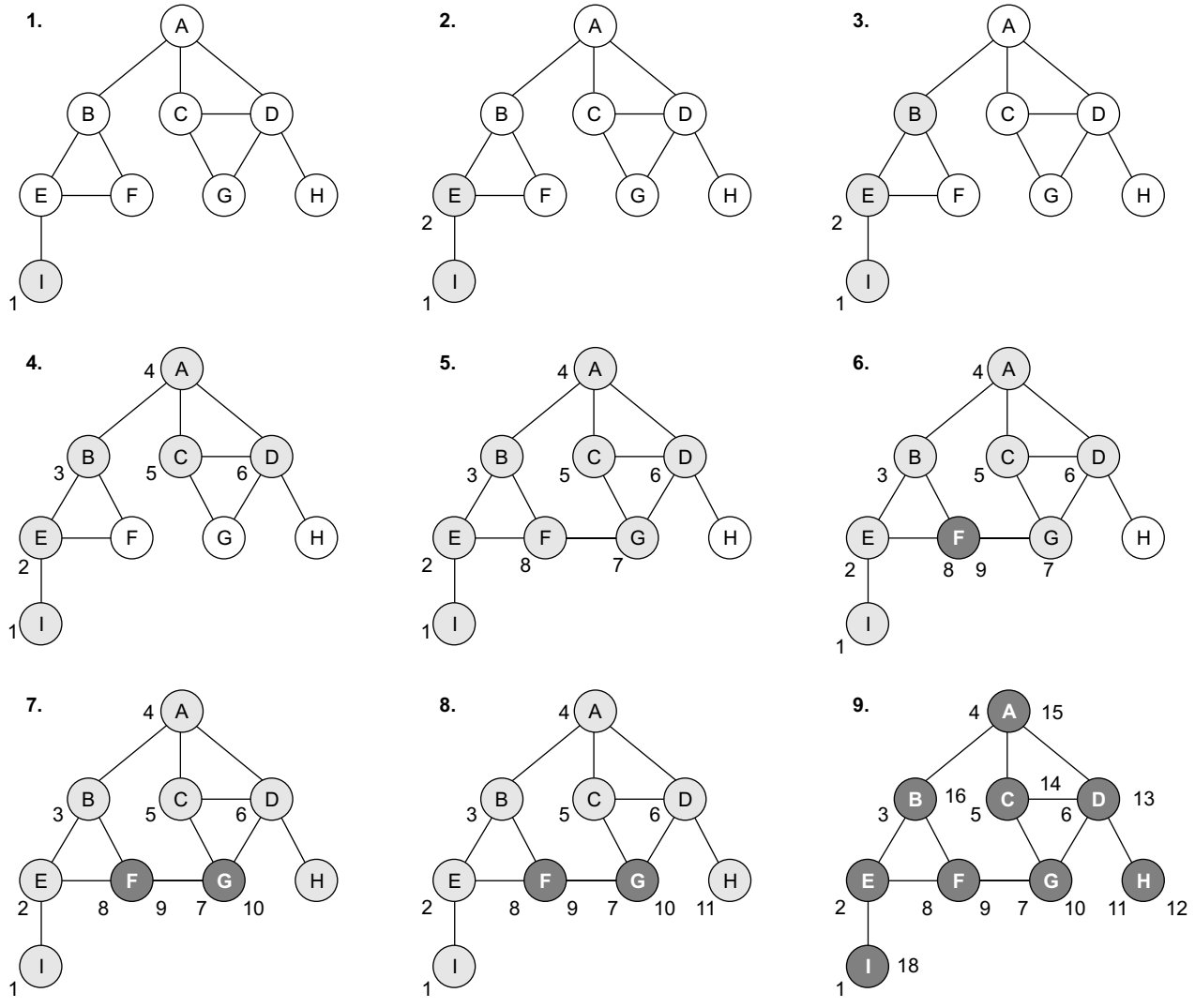


Figure 2.41 Depth First Traversal on a selected graph

■ **Example** Consider the above figure 2.41. Here, white nodes are unprocessed, gray nodes are ready and black nodes are processed. Algorithm started from node 1.

BFT Algorithm

In the case of BFT algorithm, we employ the queue. Other notations are same as above.

1. Select any node of the graph and insert into queue while making its status as Ready.
2. Repeat step 3 till queue becomes empty.
3. Remove a node (A) from the queue and make its status as processed. Insert all un-processed nodes of A into queue while making their status values as as ready.

■ **Example** Explain the BST traversal algorithm on the graph given in following figure 2.42.

1. We assume that traversal starts from node A. Thus, we insert the same into queue while making its status as ready.
2. We will remove a node from queue. Obviously, it is A itself. We will mark the same as processed and insert its unprocessed neighbours, into queue while making their status as ready.

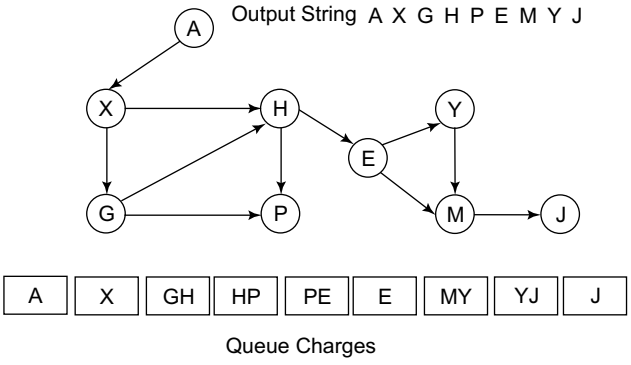


Figure 2.42 BFT traversal of a graph

3. Next, when we remove a node, we will get X. The same will be printed. Its neighbours G, H are inserted into queue.
4. Next, when we remove a node, we will get G. The same will be printed. Its unprocessed neighbour is inserted into queue while making its status as ready (Of course, though H is also G's neighbour it is not inserted as it is already in the queue).
5. Next, when we remove a node from queue, we will get H. The same will be printed. Its unprocessed neighbour, E, is inserted into queue.
6. When we remove a node from queue, we will get P. It does not have any neighbours to be inserted into queue.
7. Next, we remove a node, we will get E. The same will be printed. Its unprocessed neighbours, M.Y are inserted into queue while making their status as ready.
8. Next, when we remove, we will get M as output. Its unprocessed neighbour is inserted into queue.
9. Next, when we remove, we will get Y as output. As it does not have any unprocessed neighbours. Nothing will be inserted into queue.
10. Next, when we remove a node from queue, we will get J. The same will be printed. As it does not have any neighbours, nothing will be inserted into queue.
11. Now, queue is empty. Thus, algorithm terminates.

Minimum Distance Problems

There are plethora of problems in which we need to find out minimum distance between two nodes of a graph. For example, while planning our holiday trip, we need to find minimum distances between the places which we have identified. Similarly, in computer networks (i.e., in Internet), we need to find our best route to rout the packet. Here, we may consider many parameters such as minimum delay or minimum cost, minimum jotter, etc. In some other applications, we may have to find minimum cost routes.

In all the above problems, we will be given a weighted graph and we need to find out minimum distance route between any given two stations. In the literature, we may find many solutions to solve this problem. However, Dijkstra's algorithm is popular out of all.

Dijkstra's Algorithm

Here also, we assume same notations as in graph traversals algorithm. Initially, all the stations are at infinite distance from the source station. That is their state is considered as unprocessed. When we find a route (may not be best route) for a station from source, those stations are said to be in ready state. If we have found best possible route to a station, we mark its status as processed. The algorithm is given as follows:

1. Select the source node and mark it as processed.
2. Select all the neighbours of source station and mark them as ready and update their minimum distances as the distances from source.

3. Repeat step 4 till destination station status becomes processed.
4. Select the station (B) which is having minimum distance out of the stations which are in the ready status. Make status of B as processed and update all of its unprocessed, ready state nodes minimum distance (by adding distance from B to this station to distance of B from source) while making their status as ready.

The Minimal Spanning Tree Problem

Minimal spanning tree of a graph is the tree (no cycles) which joins all the nodes and the sum of the weights of the edges which joins all the nodes is minimum. Consider Fig. 2.43.

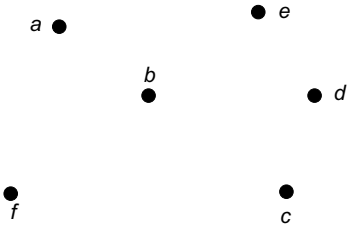


Figure 2.43 A set of planar points

In Fig. 2.44, we show three spanning trees of the set of points in Fig. 2.43.

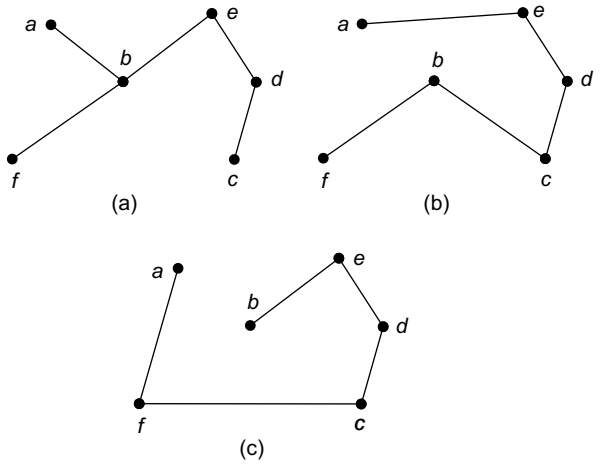


Figure 2.44 Three spanning trees of the set of points in Fig. 2.43

Among the three spanning trees, the tree in Fig. 2.44(a) is the shortestand this is what we are interested. Thus the *minimal spanning tree problem* is defined as follows: We are given a set of points and we are asked to find a spanning tree with the shortest total length.

How can we find a minimal spanning tree? A very straightforward algorithm is to enumerate all possible spanning trees and one of them must be what we are looking for. Fig. 2.45 shows all of the possible spanning trees for three points. As can be seen, there are only three of them.

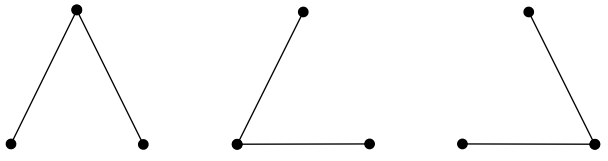


Figure 2.45 All possible spanning trees for three points

For four points, as shown in Fig. 2.46, there are sixteen 16 possible spanning trees. In general, it can be shown that given n points, there are n^{n-2} possible spanning trees for them. Thus if we have 5 points, there are already $5^3 = 125$ possible spanning trees. If $n = 100$, there will be 100^{98} possible spanning trees. Even if we have an algorithm to generate all spanning trees, time will not allow us to do so. No computer can finish this enumeration within any reasonable time.

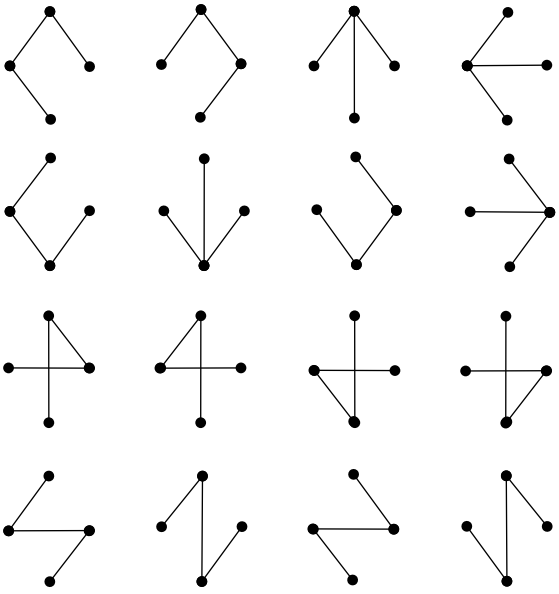


Figure 2.46 All possible spanning trees for four points

Prim's Algorithm

Consider the points in Fig. 2.43 again. Suppose we start with any point, say point *b*. he nearest neighbour of point *b* is point *a*. We now connect point *a* with point *b* , as shown in Fig. 2.47(a). Let us denote the set {*a*, *b*} as *X* and the set of the rest of points as *Y*. We now find the shortest distance between points in *X* and *Y* which is that between between *b* and *e*, We add *e* to the minimal spanning tree by connecting *b* with *e*, as shown in Fig. 2.47(b). Now *X* = {*a*, *b*, *e*} and *Y* = {*c*, *d*, *f*}. During the whole process, we continuously create two sets, namely *X* and *Y*. *X* consists of all of the points in the partially created minimal spanning tree and Y consists of all of the remaining points. In each step of Prim's algorithm, we find a shortest distance between *X* and *Y* and add a new point to the tree until *Y* is empty. For the points in Fig. 2.43, the process of constructing a minimal spanning tree through this method is shown in Fig. 2.47.

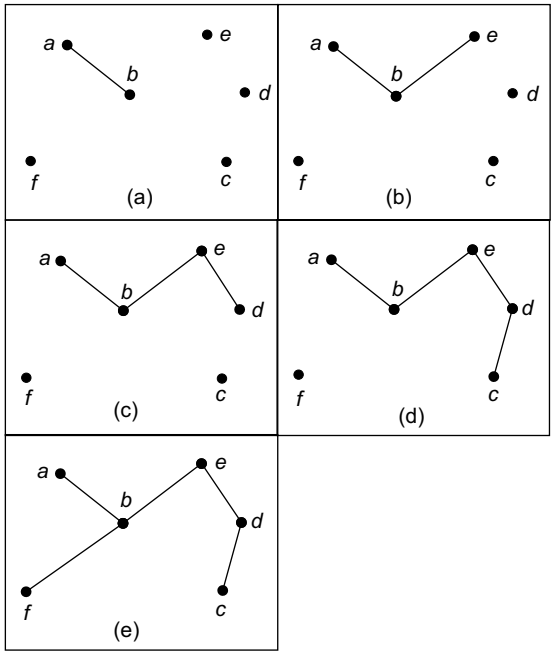


Figure 2.47 The process of constructing a minimal spanning three based upon prim's algorithm

In the above figure, we assumed that the input is a set of planar points. We can generalise it so that the input is a connected graph where each edge is associated with a positive weight. It can be easily seen that a set of planar points corresponds to a graph where there is an edge between every two vertices and the weight associated with each edge is simply the Euclidean distance between the two points. If there is an edge between every pair of vertices, we shall call this kind of graph a *complete graph*. Thus a set of planar points correspond to a complete graph. Note that in a general graph, it is possible that there is no edge between two vertices. A typical graph is now shown in Fig. 2.48.

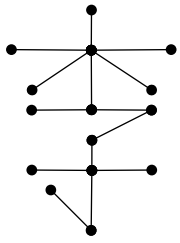


Figure 2.48 A General Graph

We now present Prim's algorithm as follows:

Algorithm 2.1 Prim's Algorithm to Construct a Minimal Spanning Tree

Input: A weighted, connected and undirected graph $G = (V,E)$.

Output: A minimal spanning tree of G.

Step 1: Let x be any vertex in V. Let $X = \{x\}$ and $Y = V \setminus \{x\}$

Step 2: Select an edge (u,v) from E such that $u \in X$, $v \in Y$ and (u,v) has the smallest weight among edges between X and Y.

Step 3: Connect u to v. Let $X = X \cup \{v\}$ and $Y = Y \setminus \{v\}$.

Step 4: If Y is empty, terminate and the resulting tree is a minimal spanning tree. Otherwise, go to Step 2.

Let us consider the graph in Fig. 2.49. the process of applying Prim's algorithm to this graph is now illustrated in Fig. 2.50.

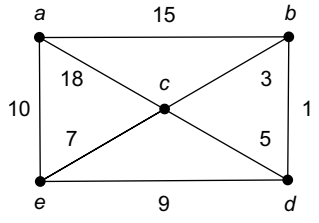


Figure 2.49 A General Graph

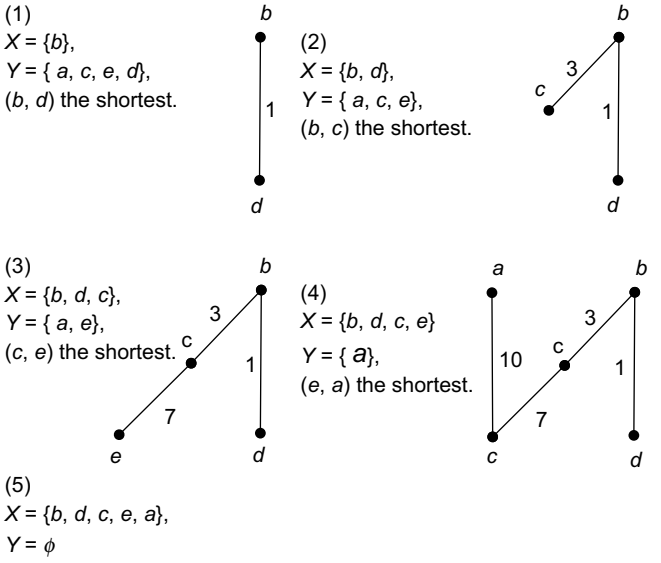


Figure 2.50 The process of applying Prim's Algorithm to the graph in figure 2.49.

Kruskal's Algorithm

Kruskal's algorithm to construct a minimal spanning tree is quite similar to Prim's algorithm. It would first sort all of the edges in the graph into an ascending sequence. Then edges are added into a partially constructed minimal spanning tree one by one. Each time an edge is added, we check whether a cycle is formed. If a cycle is formed, we discard this edge. The algorithm is terminated if the tree contains n-1 edges.

Algorithm 2.2 Kruskal's Algorithm to Construct a Minimal Spanning Tree
Input: A weighted, connected and undirected graph $G = (V, E)$.
Output: A minimal spanning tree of G.
Step 1: $T = \phi$.
Step 2: while T contains less than n–1 edges **do**
Choose an edge (v,w) from E of the smallest weight.
Delete (v,w) from E.
If the adding of (v,w)does not create cycle in T **then**
Add (v,w) to T.
Else Discard (v,w).
end while
Let us consider the graph in Fig. 2.50. The process of applying Kruskal's algorithm to this graph is illustrated in Fig.2.51. Both algorithms presented above are efficient.

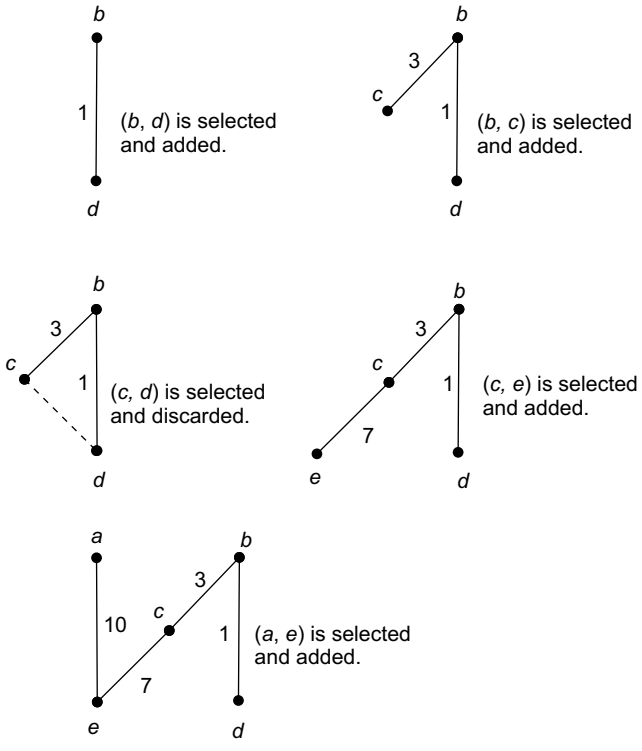


Figure 2.51 The process of applying Kruskal's Algorithm to the graph in Fig. 2.49

2.4.6.1 Graph Theory Important Points

In-degree of a node is number of edges coming into the node.
Out-degree of a node is number of edges going out of the node.
Degree of a node is sum of in and out degrees.
Self Loops are the ones in which there exists an edge for a node from itself. If u is a node and (u,u) is element of edge set E. A **self-loop** is a **cycle** of **length 1.**
If **(u,v)** is an edge then it is said to be **incident from** or **leaves** vertex **u** and is **incident to** or **enters** vertex **v**.
A path is **simple** if it contains distinct nodes.

Two paths $(v_0, v_1, v_2, \ldots, v_{k-1}, v_o)$, and $(v_0^1, v_1^1, \ldots, v_{k-1}^1, v_0^1)$ form the same cycle if there exists an integer j such that $v_i^1 = v_{i+j \bmod k}$ for **k = 0, 1,…k–1.**

An **undirected** graph is **connected** if every pair of vertices is connected by a path.
Connected components of a graph are vertices under "is reachable from" relation.

A directed graph is **strongly connected** if every two vertices are reachable from others.

A directed graph is **strongly connected** then it will have only one strongly connected component.

Two graphs are isomorphic if there exists bijection f: $V \rightarrow V^1$ such that a edge in G also in G^1 only labels may change.

If two graphs are **isomorphic** then their degrees are same.

■ **Example** How do you make an undirected graph as a directed graph using (1) adjacency list (2) adjacency matrix.

■ **Answer:**

Adjacency Matrix: If A is adjacency matrix traverse A and if A(i, j) is 1 set A(j, i) as 1.
Adjacency List: Traverse adjacency list node by node if a node V_j is in adjacency list of V_i and then keep V_i in adjacency list of V_j.
A **complete graph** is an undirected graph in which every pair of vertices is adjacent.
In a **complete graph** max path (possible) length is 1.
Forest is acyclic, undirected graph.
Tree is an acyclic, connected, undirected graph.
Multigraph is an undirected graph with multiple edges between vertices and with self-loops.
Hypergraph is undirected with hyperedges connecting subset of vertices.

■ **Problem :**

In a party every member gave shake hand to every other. Find out in total how many shake hands took place.

Assuming each member as a node and shaking with other one as an edge, then number of shake hands which took place given as

Sum of degree values of all nodes. That is 2 times of |E| where E is edge set.

Proove that in an undirected graph the length of a cycle must be at least 3.

In a directed graph if there is a cycle then it contains simple cycle.

In any connected undirected graph G(V, E) |E| > =|V|–1 is valid.

In a complete graph with N nodes there will be N (N–1)/2 edges.

A graph is weekly connected if we suppress direction and resulting undirected graph is connected.

A graph is regular if every vertex has valence (order) that it is adjacent to same no of other vertices.

The diameter of a graph is largest of all shortest path distances in the tree.

■ **Exercise :**

1. The minimum number of edges in a connected acyclic graph on n vertices is _____
2. The number of edges in a regular graph with degree d and n vertices is _____
3. Proove that the number of vertices of odd degree is always even.
4. Prove that if a connected undirected graph with n vertices has a min-cut of cardinality k, then G has at least nk/2 edges.
5. Prove that if D is a digraph then sum of in degrees and sum of out degrees are equal.
6. Prove that if T is a tree with n vertices then it will have n–1 edges.
7. Prove that only if all vertices of a Graph are even degree Euler tour is possible.

How to Find a Cycle in an Undirected Graph

We use a stack in our DFS algorithm to keep track of the vertices in the order that they are visited. One has to be careful to distinguish between a back edge, and a tree edge in the reverse direction. The algorithm below can be used to find and print a cycle in an undirected graph. Works very much like DFS, except that when it encounters a node it has already searched (which only happens when a cycle is present), it prints out the cycle.

FindCycle(G)

1. for each vertex u in V[G]
2. do color[u] < – white

3. parent[u] < – nil
3. depth <– 0
4. for each vertex u in V[G]
5. do if color[u] = white
6. then FC-Visit(u)

FC-Visit(u)

1. color[u] = gray
2. for each v in Adj[u]
3. do if color[v] = gray and v != parent[u]
4. then parent[v] = u
5. FC-PrintOut(v)
6. if color[v] = white
7. then parent[v] = u
8. FC-Visit(v)
9. color[u] = black

FC-PrintOut(v)

1. w <- v
2. do w <- parent[w]
3. print w
3. while w != v
4. end

How graphs are different from trees?
Graphs are similar to trees in that both data structures are represented with nodes and edges. Graphs are different from trees in that graphs do not have restrictions on the relationships between nodes. For example, tree nodes can only have one parent node. Graphs do not have the concept of parent and child nodes.

Consider the following pseudocode

```
for (k=0; k<n; k++) {
   for (each vertex w adjacent to vertex vk) {
          process the edge (vk, w);
   }
}
```

processes every edge in a directed graph with n vertices.

a. Suppose that the graph is represented as an array of adjacency lists. In "big-Oh" terms involving n and e, the number of edges in the graph, give the best estimate for the running time of the algorithm just given.
b. Suppose that the graph is represented as an adjacency matrix. Give the best estimate for the running time of the algorithm.

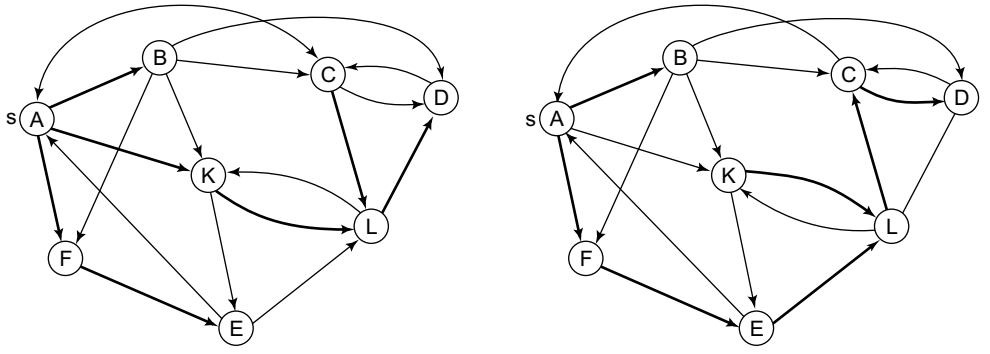
■ **Answer:** Using adjacency lists, the body of the inner loop gets executed exactly e times, and the outer loop is executed n times. Thus the total execution time is O(n+e). In particular, $O(n^2)$ is incorrect for a graph with a relatively small number of edges.

With an adjacency matrix, determining what vertices are adjacent to a given vertex takes O(n) steps, and this is done for each vertex. Thus the total running time is $O(n^2)$.

■ **Example** Below you are given two spanning trees (marked with thick edges) of a directed graph. Determine which of those trees are DFS trees. (More specifically, whether there are some orderings of adjacency lists for which DFS will produce these trees.) If so, show the DFS numbering (order in which the vertices are visited) corresponding to this DFS traversal. If not, explain why?

■ **Answer:** First figure does not represent a DFS tree. If the first edge of A is (A;B) then K should be descendant of B (not of A). If (A;K) is the first edge of A then E should be descendant of K (not of F). Finally, if (A; F) is the first edge of A then L should be a descendant of F (not of K). However, second figure represents a DFS tree. One possible DFS numbering is:

A	B	C	D	E	F	K	L
1	8	6	7	3	2	5	4



■ **Example** An undirected graph without cycles is called a forest. In other words, a forest is a collection of disjoint trees. Suppose that G is a forest with n vertices and m edges, that consists of k trees. Prove that m = n–k.

■ **Answer:** We will prove this using induction on the number of edges, with n fixed.

Base Case: m = 0. Since G has no edges, the number of trees is equal to the number of vertices, that is n = k, and the formula is true.

Inductive Step: m >=1. For the inductive assumption, assume that the formula holds for forests with m' = m-1 edges. Now pick any forest G with n vertices, m edges, and k trees, and remove one of its edges e = (u, v), to produce a new forest G' with m – 1 edges. G' has k + 1 trees, because the tree that contained e is now split into two trees. By the inductive assumption, m – 1 = n – (k + 1), which implies m = n – k.

■ **Example** We have a weighted graph G. Assuming W is the largest edge weight in G, we have created another graph G' by subtracting each of the edge weights of G from W. Explain what happens if we apply Kruskals algorithm on G'.

■ **Answer:** We get maximum spanning tree.

2.4.7 Hashing

Hashing is a technique used for storing (or organising) items so that they can be found efficiently when needed. Searching methods such as BST, etc., discussed so far vary in the efficiency with which they can find the appropriate key. Obviously, the ideal one is to have searching with constant time, O(1). Hashing is a method which has time complexity of O(1) rather than being dependent on the number of data elements. Hash tables are a way of finding a record using keys which are a part of the record and which also can be found by using specific algorithms which operate on the key to determine the location of the information. The information (records for example) can be stored in the hash table and the process of determining the location of the information in the hash table uses a hash function. The process of building or finding information in the hash table is called hashing.

In essence, hashing involves

1. Items (or records) are stored in a hash table (We can conceive hash table is a linear array. In the literature, hash table is also referred to collection of buckets)
2. A number called a hash code (or hash value) is calculated from the target value (key value) using a hash method (or hash function).
3. The hash code is then scaled to the size of the hash table, often by using the remainder (%) operator.
4. The result, the hash index, designates the position of the target value (item or record) in the hash table.

That is, while storing records we find hash index of a record from its key value and store the record in the location pointed by hash index. While searching, we repeat the same; that is given key value we access the record by calculating hash index.

Hashing is the antithesis to sorting. We all know that sorting arranges the records in some pattern or in some order like ascending or descending. However, hashing scatters the records throughout the hash table in a completely random fashion. Therefore, hashing is appropriate for implementing a specified relationship among elements but it does not lend itself to operations which attempt to make use of any other relationships among the data. In implementing the hash table, other relationships may be destroyed. For example, an ordered list can be put into a hash table but since hashing is the antithesis of sorting, the sorted nature of the data is lost. It is therefore difficult to determine the nth item in the ordered sequence. The price we pay for efficient searching is the loss of relationships which was previously existed in the data.

Hash Functions

We know that the hash array must be at least equal in size to the number of pieces of data (records or items) that has to be maintained in the hash table. Actually, we will choose an array which has a prime number of elements. Hash function is used to find the index of the hash table where the given record with a given key value has to be maintained (or searched). A good hash function should be:

- Uniform (all indexes are equally likely for a given key)
- Random (not predictable)

For example, suppose we wanted to hash a number of six digit telephone numbers of people living in a building (<1000 residents in all). Using the first three digits of the phone number is probably a bad idea (phone numbers in the same general area often have the same numbers, so all numbers may fall into same bucket or location in the hash table which is called as clustering). Instead, using the last three digits of the phone number would be a better idea as it distributes the telephone numbers uniformly. This approach is called as truncation. That is, here some parts of the key are ignored and the remaining portion is used to find the index. We do have another method known as folding to avoid clustering. Folding process breaks the key into several parts and recombines the parts to form an index. The parts may be recombined by addition, subtraction, multiplication, etc and may be truncated as well. Such a process is usually better than truncation by itself since it produces a better distribution of records (or items) because all of the numbers in the key contribute to the hash function.

Modular Arithmetic is widely used in hashing, i.e., index calculation. This ensures that the index produced falls within a specified range of the hash table. Here, the key is converted to an integer which is divided by the range of the index with the resulting function being the value of the remainder. In its simplest form the key is an integer and the modulus of the key is found directly. If the value of the modulus is a prime number, the distribution of indices obtained is quite uniform, in fact, hash tables almost always have a size which is a prime number. A table whose size is some number which has many prime factors provides the possibilty of many indices which are the same. One should not choose a hashsize which is a multiple of 2, for example. Therefore, if a hash table is to be constructed for about 1000 records, the best bet would be to have a table which can hold at least 1009 records.

Sometimes, the hash function is suggested by the structure of the key. In other cases, the best hash function can only be determined by experimentation. Perfect minimal hash functions do exist. These are hash functions which will place every item uniquely in the smallest possible hashsize (ideally equal to the number of items). Generally these hash functions are produced when all of the keys are known in advance. It is extremely rare to find a perfect hash for an arbitrary number of keys.

Collision

It is obvious that no matter what function is used, the possibility exists that the use of the function will produce an index which is a duplicate of an index which already exists. This is termed a collision. Therefore, choosing a hash function which is efficiently calculated and which yields a good distribution is only solving part of the problems in hashing. The resolution of collisions must also be addressed.

Load Factor

When looking at the efficiency of a hash table, the number of items already stored in the hash table will have an effect on the performance of the hash table. The measurement of fullness is called the *Load Factor*. In textbooks it is often represented by the symbol λ (pronounced lamda). λ = number of items in list/size of list. Put another way,. λ is simply the percentage of occupied spots in the table. Therefore: if $\lambda = 0.5$ it would mean that 50% of the table is full. $\lambda = 0.1$ means it is 10% full.

Note

Depending on collision resolution method (described in next section) it is possible that $\lambda > 1$.

Collision Resolution

The process of using a hash function to find the location in a hash table is often referred to as **open hashing**. If the position obtained is already occupied, then another open position must be found since there is a collision. Collisions are resolved by a process of rehashing (also called **closed hashing**). As an example, we will use the following list of keys in order to show what happens. We will also use

hash code or index = key mod hashsize

to determine the position in the hash table of size 11.

Key	23	18	29	28	39	13	16	42	17
Position	1	7	7	6	6	2	5	9	6

There are two major ways of resolving the collisions.

1. **Open Addressing** By taking the next open space or location as determined by rehashing the key according to some algorithm. Since the hash table has enough space to hold all of the records some space must be always open; thus the term 'open addressing'. Some open addressing procedures are:

Linear Probing Here, we start from the position at which the collision occurred and does a sequential search for the next open position. That is,

new position = (current position + 1) MOD hashsize

The results of using linear probing are shown below for the above data.

	23	18	29	28	39	13	16	42	17	
	1	7	7	6	6	2	5	9	6	
17	23	13			16	28	18	29	39	42
0	1	2	3	4	5	6	7	8	9	10

Major drawback with this approach is that once the table becomes allmost full, the length of the search for an open space increases. Also, we may find **Clustering to** occur in this method; that is, the used spaces tend to appear in groups which tend to grow and thus increase the search time to reach an open space. Of course, it eventually finds an open space.

This method only works well if the size of the array is larger than the maximum number of expected values. In other words you would expect lots of open spaces. In general you will need to keep no less than around 30% to 35% of the table empty.

The problem is that you must estimate the maximum size of the table which can be hard. Another problem is that whenever you have a collision, the item you are trying to add will have to be placed into the table in another spot. This of course could mean that it is taking up a spot that was meant for anther item that would have hashed directly into it. That item would in turn have to be placed at another place. The problem that this creates is that the items in hash tables tend to create clusters. Any attempt to hash a value into the cluster would cause the item to be placed in an alternate spot and increase the size of the cluster.

Average number of probes in an unsuccessful search using linear probing is around:

$$\frac{1+\frac{1}{(1-\lambda)^2}}{2}$$

Average number of probes for a successful search using linear probing is around:

$$\frac{1+\frac{1}{(1-\lambda)}}{2}$$

Note that the actual cost of finding an item depends on l at time that item was inserted. For example, if table was empty

when item was inserted, then any future searches for the items would always find the item where the hash function said it should be.

Incremental Functions

In order to try to avoid clustering, a method which does not look for the first open space must be used. Two common methods are used.

Quadratic Probing

If there is a collision at hash address 'h', then the probing begins by adding 1 to the address, then 4, then 9, etc. In general form this can be expressed as:

new position= (first collision position + j^2) MOD hashsize

where j is the number of the probe and hashsize is the size of the hash table. The results of using a quadratic probe are shown below.

	23	18	29	28	39	13	16	42	17
	1	7	7	6	6	2	5	9	6
	23	13		17	16	28	18	29	42	39
0	1	2	3	4	5	6	7	8	9	10

Once again, the best results are obtained when the size of the hash table is a prime number since numbers which have several divisors would yield a fair number of duplicate values when the mod function is used. When a prime is used for hashsize, the number of distinct probes which will be made is (hashsize + 1) DIV 2. Therefore, not all positions are probed but the results are generally satisfactory. If the collision cannot be resolved, then **overflow** is said to be occured.

Key-dependent Increments

An obvious difficulty in the quadratic probe is that overflow may occur when there is still space in the hash table. While this can be partly rectified by making a table slightly larger than would be required when all elements are in the table, this is wasteful, especially for large hash tables. To counteract this, key-dependent increments are used. These increments vary according to the key used for the hash function. If the original hash function results in a good distribution, then key-dependent functions work quite well for rehashing and all locations in the table will eventually be probed for a place to put the element.

Key-dependent increments are determined by using the key to calculate a new value and then using this as an increment to determine successive probes. Some checks should also be done before the increment obtained is used to search for new positions. For example, since the original hash function was key MOD 11, we might choose a function of key MOD 7 to find the increment. Thus the closed hash function becomes

newposition= (currentposition + (key DIV 11)) MOD 11

The results of using this key dependent increment method is shown below.

	23	18	29	28	39	13	16	42	17
	1	7	7	6	6	2	5	9	6
	2	1	2	2	3	1	1	3	1
	23	13		39	16	28	18	17	29	42
0	1	2	3	4	5	6	7	8	9	10

For instance, hash code for 23 is 23%11, i.e., 1. Thus, 23 is stored in location 1. Similarly, 18%11 is 7. Thus, 18 is stored at location 7. In the same way, 29%11 is 7. Now, collision is occurred. We calculate increment value as 29/11, i.e, 2(integer division). Thus, new location becomes (7+2)%11, i.e., 9. Thus, 29 is stored at location 9. Similarly, 28%11 is 6. As no element is stored at location 6, we store 28 there. Now, take 39 for hashing. We calculate 39%11, i.e., 6. We know allready 28 stored there; thus collision is occurred. Thus, we calculate increment value, which become 39/11=3. Now, new location is (6+3)mod 11, where we have 29. Thus, we try (6+2*3)%11=1. We know 23 is available at this location. Thus, we try (6+3*3)%11=4. We don't have any element at location 4. Thus, we store 39 at location 4.

This closed hash function works for the data in the example because there are no keys with a value of less than 11. **In all of the closed hash functions it is important to ensure that an increment of 0 does not arise. If the increment is equal to hashsize the same position will be probed each time so this value cannot be used.**

The increment is now dependent on the key and different increments exist. Generally, a different number is used to find the modulus and while the open hash function could be used to find an increment for the closed hash function, this should be avoided if possible. Another problem exists if the size of the hash table is not a prime. For a number of the increment values only some of the positions will be probed regardless of the number of increments.

If we ensure that the hashsize is prime and the divisors for the open and closed hash are prime, the rehash function does not produce a 0 increment, then this method will usually access all positions as does the linear probe. Using a key-dependent method usually result reduces clustering and therefore searches for an empty position should not be as long as for the linear method.

Collisions increase the search length to find any key since the search follows the same procedure as doing the original insertion. To reduce this effect, coalesced hash table is used. It adds some steps to the process of building the table and it also requires a second vector rather than a single one-dimensional array. The first column/row in the table is the hash table of keys. The second row or column is an indicator of which element to look in if a collision occurs. Table 2.5 was built using the simplest collision resolution technique, that of a linear probe.

Table 2.5

	23	18	29	28	39	13	16	42	17
	1	7	7	6	6	2	5	9	6

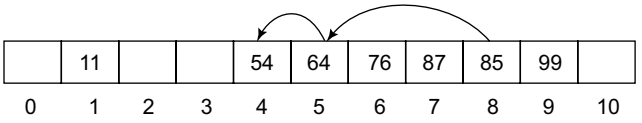
17	23	13			16	28	18	29	39	42
		3				9	8		10	0
	1	2	3	4	5	6	7	8	9	10

For instance, 23 and 18 will be hashed into empty positions 1 and 7, respectively. When we try to insert 29, it collides with 18. In the reference vector, an 8 indicates that a collision at position 7 was initially resolved by going to position 8. Thus, 29 will be will be placed at location 8. Then, 28 hashes to position 6 and is placed there. When we try to insert 39, it also hashes to position 6, but there is no previous collision there so we just use the rehash and move one more to position 7. At this point, we see that there was already a collision there and it was necessary to go to position 8, so we move to 8. There have been no previous collisions there so we use the rehash to get to position 9 where 39 can be placed. It is then necessary to go back to position 6 and indicate that the resolution is in location 9. The number of probes that we went through to place 39 is now reduced to only 2. If we put a 9 in the reference vector in position 6, then a search for 39 would go to position 6, and then to position 9, eliminating some of the probes. The next collision occurs when an attempt is made to place 42 which should go in position 9. There is no link to another place so we use the rehash to put 42 in position 10 and set the indicator in 9 to 10. Similarly, 17 eventually ends up in position 0 by starting at 6 and going to 9 and 10, and finally 0. The indicator in 10 is set to 0. Resolving collisions by coalescing the table can reduce the average probes to find a key. It is also slightly easier to just follow the indicator to a new array element than to rehash and then go to the element.

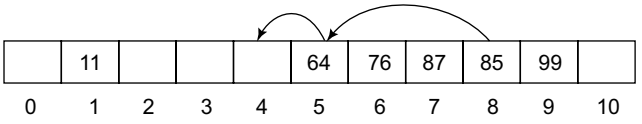
Effect of Deletions

Deletions from a hash table pose considerable difficulties. Memory space which becomes free because of a deletion of a key has to be re-used. If there were no (previous) collisions at that location, then the deletion would pose no problem; otherwise it poses difficulties. That is, if after inserting the first element into the location, subsequent hashing has resulted in the same location and rehashing has been used to place the element in another position, taking out the first element causes problems. When searching for an element in a hash table, it is important to note that the searching algorithm is really the same as the insertion algorithm. Therefore, if there is a collision some mechanism is available to resolve the problem and place the element. The probe continues until an open space is found. Removing any element breaks the search because an open element indicates that an element can be placed, or conversely, indicates that no more elements hash to that location given the initial key. In order to cope up with this, any deletion must set a key in the hash table (often called a tombstone) to indicate that the space is now open for the insertion of a new element if insertion is taking place but that if the hash table is being searched, there are other elements which follow and the collision resolution procedures should be used to continue searching the table. We can also move the elements as explained below.

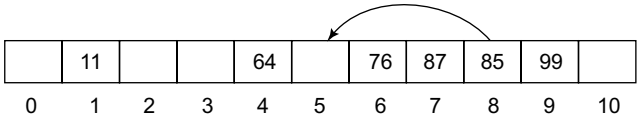
Let us suppose that after doing a number of hashes we have the following array. Arrow indicates their proper location in the array (no arrow means item is already in proper place):



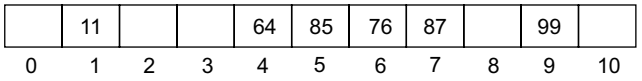
Using linear probing we must be very careful about leaving empty spaces. This is because when we search we look for empty spaces as indication that the search is done. We must not leave an empty space between where an item should be and where an item actually is. Suppose now we delete 54:



We can not just leave the array like this because if we try to look for 64 now, we will not find it since the space where it should have been is empty. We will need to move 64 over



Problem now of course is that 85 will not be found. We will need to move 85 over too.

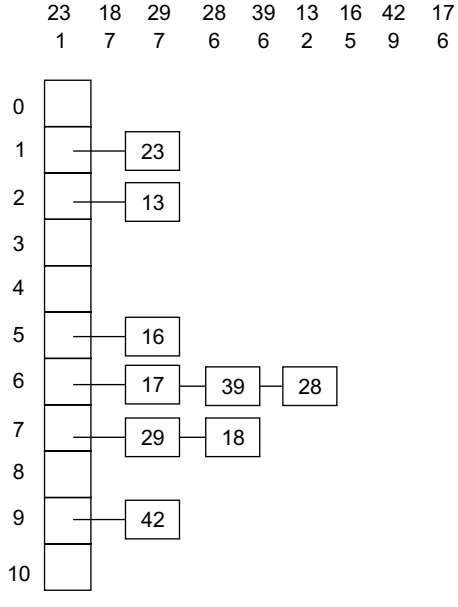


2. Collision Resolution by Chaining

External Chaining

Array as hash table is reasonable as it extends random access to any element. Thus access can be reduced to O(1), or if collisions occur some multiple depending on the number of rehashes required before the collision is resolved. However, arrays can use substantial amounts of memory and in order to make collision resolution effective it is often necessary to employ arrays which are larger than number of elements (to avoid overflow condition). To alleviate this problem, we make the hash table as an array of pointers to linked lists. Thus, when a key is hashed, the position in the table is accessed which refers to a linked list. Resolution of collisions is simple ; it requires only that a node be added to a linked list. This insertion is usually as the first item in the list which is the easiest to accomplish since it does not require a traversal of the list. Deletions also ose no special problems. The oringinal data is shown below in Table 2.6 where collisions are resolved by chaining.

Table 2.6



We know that creation of linked lists involves use of dynamic memory. If the hash function produces a reasonable distribution of keys, none of the linked lists will become very long. However, there may be elements in the hash table which remain unused, the savings in using dynamic storage for the information should make up for the few un-used elements.

Obviously, chaining consistently requires fewer probes than open addressing. However, traversal of the linked list is slow and if the records are small it may be just as well to use open addressing. Chaining is best under two conditions; when the number of unsuccessful searches is large or when the records are large. Open addressing would likely be a reasonable choice when most searches are likely to be successful, the load factor is moderate, and the records are relatively small.

For chaining, the runtimes depends on l. The average length of each chain is λ. λ is the number of expected probes needed for either an insertion or an unsuccessful search. For a successful search it is $1 + \lambda/2$ probes.

In comparison to other methods of search it is important to note that the number of probes is dependent only on the load factor on the hash table and not on the absolute number of items in the table.

Double Hashing

Double Hashing is works on a similar idea to linear and quadratic probing. Use a big table and hash into it. Whenever a collision occurs, choose another spot in table to put the value. The difference here is that instead of choosing next opening, a second hash function is used to determine the location of the next spot. For example, given hash function H1 and H2 and key, do the following:

- Check location H1(key). If it is empty, put record in it.
- If it is not empty calculate H2(key).
- check if H1(key)+H2(key) is open, if it is, put it in
- repeat with H1(key)+2*H2(key), H1(key)+3* H2 (key) and so on, until an opening is found.

Like quadratic probing, one must take care in choosing H2. H2 CANNOT return 0. H2 must be done so that all cells will be probed eventually.

■ **Example** Suppose we want to store names, or words, consisting of lower case letters only. We will consider three hash functions, all hashing to a hashtable with 26 entries:

```
unsigned int hash1( char *s ) {
return s[0] - 'a' ;
}
unsigned int hash2( char *s ) {
int h = 0, i ;
for( i=0 ; s[i] != '\0' ; i++ ) {
h = h + s[i] ;
}
return h % 26 ;
}
unsigned int hash2( char *s ) {
int h = 0, i ;
for( i=0 ; s[i] != '\0' ; i++ ) {
h = h * 3 + s[i] ;
}
return h % 26 ;
}
```

Suppose we wish to store the following words: *break , brake, dear, dare, bristol, bristle, not and ton*

(a) How many collision will we have when using hash1?

■ **Answer:** 4 (brake, bristol and bristle collide with break, and dear with dare)

(b) How many collision will we have when using hash2?

■ **Answer:** 3 (brake with break, hear with hare, and ton with not)

(c) How many collision will we have when using hash3 (guess, don't compute the answer)?

■ **Answer:** 0, there is a good chance that there is a collision already, birthday paradox.

(d) What is the strength of hash1?

■ **Answer:** simple and fast!

(e) What is the strength of hash3?

■ **Answer:** few collisions

(f) assume that we use hash function 1 and we use open addressing to resolve this. Sketch the contents of the hash table.

■ **Answer:**

slot 0 is empty;
slot 1 contains break;
slot 2 contains brake;
slot 3 contains dear;
slot 4 contains dare;
slot 5 contains bristol;
slot 6 contains bristle;
slot 7-12 are empty;
slot 13 contains not;
slot 14-18 are empty;
slot 19 contains ton;
slot 20-25 are empty;

(g) We will need a string comparison to compare the contents of a hash cell with a word when we are looking a word up. how many string comparisons are required to check Whether the following words are in the hashtable: aardvark, chicken, bristol, peanuts, gorilla?

■ **Answer:** aardvark: 0 (slot 0 is empty), chicken: 5 (comparewith all values from slots 2-6 inclusive), bristol: 5 (compare with all values from slots 1-5 inclusive), peanuts: 0 (slot 16 is empty), gorilla: 0 (slot 7 is empty)

(h) Repeat the previous questions, assuming that we use direct chaining.

■ **Answer:** slot 0 is empty;

slot 1 contains bristle, bristol, brake, break.
slot 2 is empty
slot 3 contains dare, dear.
slot 4-12 are empty;
slot 13 contains not;
slot 14-18 are empty;
slot 19 contains ton;
slot 20-25 are empty;
0, 0, 2, 0, 0

■ **Example** Assume we have an array storing the values 3, 9, 12, 13, 24, 25, 26, 200, 202, 208, 215, 300, 314, 31415, 80000. The lowest value is stored at index 0, the highest at index 14.

(a) Suppose we want to check whether 14 is in the array or not. Which values does a binary chop ened to compare with?

■ **Answer:** 200, 13, 25, 24

(b) How many comparisons would you need if you did a linear search?

■ **Answer:** Five: 3, 9, 12, 13, 24

(c) What are the minimum and the maximum number of comparisons for the binary chop on this array?

■ **Answer:** One and Four

(d) What are the minimum and the maximum number of comparisons for a linear search on this array?

■ **Answer:** One and Fifteen

(e) Could you store this in a hash-table with, say, 17 elements? What would a suitable hash function be?

■ **Answer:** Yes. Modulo 17 is the simplest hash-function.

■ **Example** Give the keys 4, 0, 18, 24, 16, 20, 30, 31, 19, 8, 14 and 13, insert keys (according to the given order) into an initially empty 13-item hash table using hash function ($h(k) = k \bmod 13$) where k represents a key. Show the content of hash table after insertion, assuming collisions are handled by

i. Linear probing

ii. Double hashing where the secondary hash function is $h'(k) = 1 + (k \bmod 11)$.

(a) Given n equal keys, totally how many probes are required when linear probing is used for collision handling? Express the number of probes in term of Big-Oh notation. Explain your answer.

(b) Given n equal keys, can double hashing better than linear probing in term of the number of probes involved? Explain your answer.

Answer: The following table illustrates the hashing with all the given keys.

Keys	$h(k) = k \bmod 13$	$(h'(k) = 1 + (k \bmod 11)$	$(h(k) + jh'(k)) \bmod 13$
4	4		
0	0		
18	5		
24	11		
16	3		
20	7		
30	4	9	0, 9
31	5	10	2
19	6		
8	8		
14	1		
13	0	3	3,6,9,12

Results of Linear probing

0	1	2	3	4	5	6	7	8	9	10	11	12
0	14	13	16	4	18	30	20	31	19	8	24	

Results of double hashing where the secondary hash function is $h'(k) = 1 + (k \bmod 11)$.

0	1	2	3	4	5	6	7	8	9	10	11	12
0	14	31	16	4	18	19	20	8	30		24	13

Answer for (a)

For ith ($i \geq 2$) insertion, we need $i - 1$ probes. So in total we need

$$1 + 2 + ... + n - 1 = O(n^2)$$

Answer for (b)

Double hashing hashes key k according to the index $H(k, j) = (h(k) + jh'(k)) \bmod N$ where $j = 1, 2$ .... Let's see what kind of j would make $H(k, j_1) = H(k, j_2)$, i.e.,

$(h(k) + j_1 h'(k)) \bmod N = (h(k) + j_2 h'(k)) \bmod N \Rightarrow (j_1 - j_2)h'(k)) \bmod N = 0$

This means that either $h'(k) = 0$ or $j_1 - j_2 = 0$. The former is impossible based on the property of hash function. So, $j_1 - j_2$ is the only option. Therefore, double hashing to the same key will require at least N probes before it hashes a key into the same index as before. In this case, double hashing is better than linear probing.

What are the properties of a good hash function?

■ **Answer:** It should distribute values nicely. It is not expensive to compute.

What is bucket hashing (chaining)? Briefly explain.

■ **Answer:** Approach where the hash table contains list of objects.

In the context of hashing, what is the load factor? Briefly explain.

■ **Answer:** Number of entries that are mainatained in hash table/hash table size.

Suppose that 31 distinct integers are arranged in ascending order in an array named values. Suppose also that the following code is used in a method to locate an integer named key in the array.

```
int leftIndex = 0;
int rightIndex = values.length() - 1;
while (leftIndex <= rightIndex) {
        int middleIndex = (leftIndex + rightIndex)/2;
        if (values[middleIndex] == key) {
                return true;
        } else if (values[middleIndex] < key) {
                leftIndex = middleIndex+1;
        } else {
                rightIndex = middleIndex-1;
        }
}
return false;
```

Compute the total number of comparisons necessary to locate all 31 values in the array using the above loop. A comparison occurs in one of the lines

```
if (values[middleIndex] == key) { ...
} else if (values[middleIndex] < key) { ...
```

Now suppose that the 31 integers are stored in a chained hash table with k chains, in which the hash function distributes the integers to chains as evenly as possible. What is the smallest value of k for which the total number of comparisons to find all 31 values in the hash table is less than the answer you computed for part a?

Answer: Search for values[15], the middle element, requires one comparison; search for values[7] or values[23], the middle elements of the first 15 and last 15 elements respectively, requires three comparisons; four keys require five comparisons; and so on. Total comparisons to find all 31 keys is

1*1 + 2*3 + 4*5 + 8*7 + 16*9 = 1 + 6 + 20 + 56 + 144 = 227.

Now, we will try from small values of the hash table size. If we assume there are two chains of evenly distribute, then there can be 15 keys in one chain and 16 in the other. The total number of comparisons to access the keys in the 15-key chain is 1+2+...+15 = 16*15/2 = 8*15 = 120. The corresponding figure for the 16-key chain is 17*16/2 = 17*8 = 136. The total is 256. If there are three chains, there are 10 keys in two chains and 11 in the other. Total comparisons is then 5*11 + 5*11 + 11*6 = 176. The desired value for k is thus 3.

2.4.8 Dynamic Programming

The dynamic programming strategy can be explained by considering the graph in Fig. 2.52. Our problem is to find a shortest route from vertex *S* to vertex *T*. As can be seen, there are three branches from *S*. Thus we have at least three routes, namely going through *A*, going through *B* and going through *C*. We have no idea which one is the shortest. But we have the following principle: *If we go through a vertex X, we should find a shortest route from X to T*

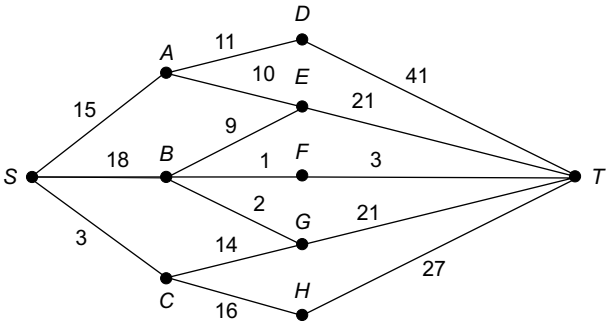


Figure 2.52 A Graph

Let $d(x,y)$ denote the shortest distance between vertices x and y. We have the following equation:

$$d(S,T)=\min\begin{cases}d(S,A)+d(A,T)\\ d(S,B)+d(B,T)\\ d(S,C)+d(C,T)\end{cases}$$

The question is: How do we find the shortest route from, say vertex A, to vertex T? Note that we can use the same principle to find a shortest route from A to T. That is, the problem to find a shortest route from A to T is the same as the problem finding a shortest route from S to T except the size of the problem is now smaller.

The shortest route finding problem can now be solved systematically as follows:

$$d(S,T)=\min\begin{cases}d(S,A)+d(A,T)\\ d(S,B)+d(B,T)\\ d(S,C)+d(C,T)\end{cases}$$

$$=\min\begin{cases}15+d(A,T)\\ 18+d(B,T)\\ 3+d(C,T)\end{cases} \tag{2.1}$$

$$d(A,T)=\min\begin{cases}d(A,D)+d(D,T)\\ d(A,E)+d(E,T)\end{cases}$$

$$=\min\begin{cases}11+d(D,T)\\ 10+d(E,T)\end{cases} \tag{2.2}$$

$$=\min\begin{cases}11+41\\ 10+21\end{cases}=31$$

$$d(S,T)=\min\begin{cases}d(B,E)+d(E,T)\\ d(B,F)+d(F,T)\\ d(B,G)+d(G,T)\end{cases}$$

$$=\min\begin{cases}9+d(E,T)\\ 1+d(F,T)\\ 2+d(G,T)\end{cases}=\min\begin{cases}9+21\\ 1+3\\ 2+21\end{cases}=4 \tag{2.3}$$

$$d(S,T)=\min\begin{cases}d(C,G)+d(G,T)\\ d(C,H)+d(H,T)\end{cases}$$

$$=\min\begin{cases}14+21\\ 16+27\end{cases}=35 \tag{2.4}$$

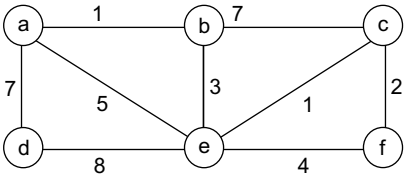
Substituting 2.2, 2.3 and 2.4 into 2.1, we obtain that $d(S, T) = \min\{15 + 31, 18 + 4, 3 + 35\} = 22$, which implies that the shortest route from S to T is $S \rightarrow B \rightarrow F \rightarrow T$. As shown above, the basic idea of dynamic programming strategy is to decompose a large problem into several sub-problems. Each sub-problem is identical to the original problem except the size is smaller. Thus the dynamic programming strategy always solves a problem recursively.

In the next section, we go back to the longest common subsequence problem and show how the dynamic programming strategy can be applied to solve the problem.

All Pairs Shortest Paths (Floyd-Warshall)

A dynamic programming algorithm.

```
FOR k=1TOn
 FOR i=1 TO n
 FOR j=1TO n
  c(i,j,k) = min(c(i,j,k-1),
  c(i,k,k-1)+c(k,j,k-1)
  )
```



$k = \emptyset$	a b c d e f a 1 ∞ 7 5 ∞ b 7 ∞ 3 ∞ c ∞ 1 2 d 8 ∞ e 4 f
k = {a}	a b c d e f a 1 ∞ 7 5 ∞ b 7, b–a–c ∞, b–a–d 3, b–a–e ∞, b–a–f c ∞, c–a–d 1, c–a–e 2, c–a–f d 8, d–a–e ∞, d–a–f e 4, e–a–f f
k = {a, b}	
k = {a, b, c}	
k = {a, b, c, d}	
k = {a, b, c, d, e}	
k = {a, b, c, d, e, f}	

Application of the Dynamic Programming Strategy to Solve the Longest Common Subsequence Problem

The Longest Common Subsequence Problem

In this section, we will introduce another seemingly difficult problem and again show an efficient algorithm to solve this problem.

Consider the following sequence:

S: ABDDRGTY.

The following sequences are all subsequences of *S*

ABDDRGT, DGTY, DDRGT, BDR, ABDTY, DDRT, GY, ADRG ABTY, TY, ABG, DDY, AR, BTY, TY, Y

Suppose that we are given two sequences as follows:

S_1: ABDDRGTY

S_2: CDEDGRRT

Then we can see that the following sequences are subsequences of both S_1 and S_2

DRT
DRGT
DDRGT

All of the above sequences are called *common subsequences* of S1 and S2. The *longest common subsequence problem* is

defined as follows: Given two sequences, find a longest common subsequence of them. If one is not familiar with algorithm design, one may be totally lost with this problem. Algorithm 2.3 is a naive algorithm to solve this problem.

Algorithm A Naive Algorithm for the Longest Common Subsequence Problem

Step 1: Generate all of the subsequences of, say S_1

Step 2: Starting with the longest one, check whether it is a subsequence of S_2

Step 3: If it is not, delete this subsequence and, return to Step 2

Step 4: Otherwise, return this subsequence as a longest common subsequence of S_1 and S_2.

The trouble is that it is exceedingly time-consuming to generate all of the subsequences of a sequence. A much better and much more efficient algorithm employs a strategy called the dynamic programming strategy. Since this strategy is used very often to solve problems in molecular biology, we shall describe it in the next section.

The longest common subsequence problem presented above. It was also pointed out that we can not solve the problem in any naïve and unsophisticated way. In this section, we shall show that this problem can be solved elegantly by using the dynamic programming strategy.

We are given two sequences: $S_1 = a_1\ a_2\ ....\ a_m$ and $S_2 = a_1\ a_2\ ....\ b_n$. Consider a_m and b_n. There are two cases:

Case 1: $a_m = b_n$. In this case, a_m, which is equivalent to b_n, must be included in the longest common subsequence. The longest common subsequence of S_1 and S_2 is the longest common subsequence of $a_1\ a_2\ ....\ a_{m-1}$ and $b_1\ b_2\ ....\ b_{n-1}$ plus a_m.

Case 2: $a_m \neq b_n$. Then we find two longest common subsequences, that of $a_1\ a_2\ ....\ am$ and $b_1\ b_2\ ....\ b_{n-1}$ and that of $a_1\ a_2\ ....\ a_{m-1}$ and $b_1\ b_2\ ....\ b_n$. Among these two, we choose the longer one and the longest common subsequence of S_1 and S_2 must be this longer one.

To summarize, the dynamic programming strategy decomposes the longest common subsequence problem into three identical sub-problems and each of them is of smaller size. Each sub-problem can now be solved recursively.

In the following, to simplify the discussion, let us concentrate our mind to finding the length of a longest common subsequence. It will be obvious that our algorithm can be easily extended to find a longest common subsequence.

Let $LCS\ (i, j)$ denote the length of a longest common subsequence of $a_1\ a_2\ ....\ a_i$ and $b_1\ b_2\ ....\ b_j$. $LCS\ (i, j)$ can be found by the following formula:

$$LCS(i,j) = \begin{cases} LCS(i-1, j-1)+1 & \text{if } a_i \neq b_j \\ \max\{LCS(i-1, j), LCS(i, j-1)\} & \text{if } a_i = b_j \end{cases}$$

$$LCS\ (0, 0) = LCS\ (1, 0) = LCS\ (0, 1) = 0$$

The following is an algorithm to find the length of a longest common subsequence based upon the dynamic programming strategy:

Algorithm An Algorithm to find the length of a Longest Common Subsequence based upon the Dynamic Programming Strategy

Input: $a_1\ a_2\ ....\ a_m$ and $b_1\ b_2\ ....\ b_n$

Output: The length of a longest common subsequence of A and B, denoted as $LCS\ (m, n)$

Step 1: $L\ (0, 0) = L\ (1, 0) = L\ (0, 1) = 0$

Step 2: for $i = 1$ to m **do**

for $j = 1$ to n **do**

if $ai = bj$ **then** $LCS\ (i, j) = \{LCS\ (i - 1, j - 1) + 1$

else $\{LCS\ (i, j) = \{\max\ \{LCS\ (i - 1, j\ ) - 1)\ LCS\ (i, j - 1)$

end for

end for

Let us consider an example.

A = AGCT and B = CGT

The entire process of finding the length of a longest common subsequence of A and B is now illustrated in the following Table 2.7. By tracing back, we can find two longest subsequences CT and GT.

Let us consider another example: A = aabcdec and B = badea. Table 2.7 illustrates the process.

Again, it can be seen that we have two longest common subsequences, namely bde and ade.

Table 2.7 The process of finding the length of Longest Common Subsequence of AGCT and CGT

		i				
		0	1	2	3	4
j			A	G	C	T
0		0	0	0	0	0
1	C	0	0	0	1	1
2	G	0	0	1	1	1
3	T	0	0	0	0	0

Table 2.8 The process of finding the length of a Longest Common Subsequence of A = aabcdec and B = badea

		i							
		0	1	2	3	4	5	6	7
j			a	a	b	c	d	e	c
0		0	0	0	0	0	0	0	0
1	b	0	0	0	1	1	1	1	1
2	a	0	1	1	1	1	1	1	1
3	d	0	1	1	1	1	2	2	2
4	e	0	1	1	1	1	1	3	3
5	a	0	1	2	2	2	2	3	3

The Dynamic Programming Approach to Find a Longest Common Subsequence

Best Case: O (n^2)

Average Case: O (n^2)

Worst Case: O (n^2)

In Table 2.9, we list different time-complexity functions in terms of the input sizes.

Table 2.9 Time-complexity functions

	Problem Size : n			
Time-Complexity Functions	10	10^2	10^3	10^4
$\log n$	3.3	6.6	10	13.3
n	10	10^2	10^3	10^4
$n \log n$	0.33×10^2	0.7×10^3	10^4	1.3×10^5
n^2	10^2	10^4	10^6	10^8
$2n$	1024	1.3×10^{30}	$>10^{100}$	$>10^{100}$
$n1$	3×10^6	$>10^{100}$	$>10^{100}$	$>10^{100}$

As can be seen in this table, it is quite important to be able to design algorithms with low time-complexities. For instance, suppose that one algorithm has O (n) time-complexity and another one has O (log n) time-complexity. When n = 10000, the algorithm with O (log n) takes much less number of steps than the algorithm with O (n) time-complexity.

■ **Example** Largest subsequence with O(1). That is, we were given an array with some values are positive while others are negative (If all are positive, then total of the whole array itself becomes largest. So, no fun). We have to find out subsequence with largest total.

Here, we are taking a 2-D array aux in which first column is used to maintain the index while the second column is for running total.

```
int main(){
   int i,a[100],aux[100][2],j,max,n,index;
   printf("Enter Number of elements\n");
   scanf("%d",&n);
 printf("Enter values\n");
 for(i=0;i<n;i++)scanf("%d", &a[i]);
 aux[0][0]=0;
 aux[0][1]=a[0];
 for(i=1;i<n;i++){
   if(a[i]+aux[i-1][1]>a[i]){
   aux[i][0]=aux[i-1][0];
   aux[i][1]=a[i]+aux[i-1][1];
 }
 else{
   aux[i][0]=i;
   aux[i][1]=a[i];
   }
}
max=aux[0][1];
index=0;
for(i=1;i<n;i++){
  if(aux[i][1]>max){
     max=aux[i][1];
     index=i;
  }
}
printf("Sum %d from %d to %d\n", max,aux[index][0], index);
return 0;
}
```

■ **Example** Given two sequences *S* and *T* each consisting of some letters of a certain alphabet, for example, assume that *S* = (*aabcdacbacdfghello*) and *T* = (*dqrworldxxx*), the edit distance, *d(S; T)* is defined as the minimal number of deletions, insertions and substitutions needed to perform on *S* so that it would become identical to *T*. The input for the problem is two sequences *S* and *T*, and a positive integer parameter *k*. Develop an algorithm whose running time is *O(nk)*, and determines if *d(S; T)* <= *k*. Note that the algorithm does not need to find the exact distance.

■ **Answer:** Assume $|S| = n$, $|T| = m$. We can construct an (m+1)*(n+1) matrix. From the left-top position (1, 1), we can use dynamic programming to fill the edit distance for each position (i, j).

$S(i, j) = MIN(S(i-1, j)+1, S(i, j-1)+1, S(i-1, j-1) + mismatch(A[i], B[j])$ where mismatch(c1, c2) = 0 if c1=c2, otherwise mismatch(c1, c2) = 1. This is to computer the exact edit distance.

■ **Example** Let A = {A1, . . .,An} be a set of distinct coin types, where A1 < A2 < . . . < An. The coin-changing problem is defined as follows. Given an integer C, find the smallest number of coins from A, that add up to C, given that unlimited number of coins of each type is available. Design an efficient dynamic programming algorithm that on inputs A and C,

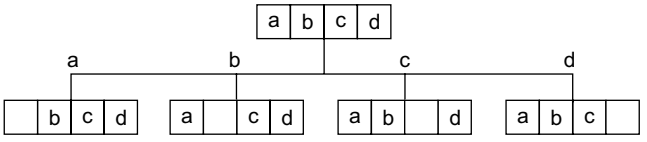
outputs the minimum number of coins needed to solve the coin-changing problem. State and prove time complexity of your algorithm.

■ **Answer:** Let N(i) be an optimal number of coins needed to solve coin-changing problem. Then $N(i) = 1+\min_j\{N(i-A_j)\}$. Hence computation of every N(i) requires O(n) time. We need to compute C such values, hence overall time is O(nC).

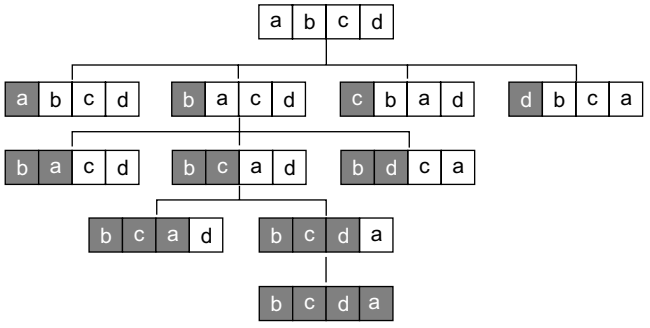
2.4.9 Genrating Permutations

A permutation can be obtained by selecting an element in the given set and recursively permuting the remaining elements.

$$P(a_1,.....,a_N)=\begin{cases}a_i, P(a_1,.....,a_{i-1}, a_{1+1},.....,a_N) & \text{if } N>1\\ a_N & \text{if } N=1\end{cases}$$



At each stage of the permutation process, the given set of elements consists of two parts: a subset of values that already have been processed, and a subset that still needs to be processed. This logical seperation can be physically realised by exchanging, in the i'th step, the i'th value with the value being chosen at that stage. That approaches leaves the first subset in the first i locations of the outcome.




```
permute(i)
  if i == N output A[N]
  else
    for j = i to N do
      swap(A[i], A[j])
      permute(i+1)
      swap(A[i], A[j])
```

■ **Example** We were given some coins of various denominations and we are asked to make some amount minimal number of coins.

- The solution lies in keeping the optimal solution, so far, of sub-problems in minCoins.
- We iterate through each denomination, kept in array coins[j], skipping denominations that are larger than the amount of money we are changing.
- Otherwise, we test whether the number.of coins used, coinsUsed[ ], for the next combination lowers the minCoins obtained so far.
- LastCoin[ ] keeps track of the coins used for solution

The following function does the required action:

```
void makeChange(int coins[ ],int differentCoins, int maxChange, int coinsUsed[ ], int lastCoin[ ])
{ coinsUsed[0] = 0; lastCoin[0] = 1;
 for (int cents = 1; cents <= maxChange; cents++)
```

```
{
 int minCoins = cents; int newCoin =1;
 for(int j = 0; j < differentCoins; j++)
 {
  if (coins[j] > cents) //denominations larger than amt continue;
  if (coinsUsed[cents-coins[j]] +1 < minCoins)
  { minCoins = coinsUsed[cents-coins[j]] + 1;
        newCoin = coins[j];    }
 }
 coinsUsed[cents] = minCoins;
 lastCoin[cents] = newCoin;      }    }
```

■ **Example** Consider an n by n array of positive integers (a_{ij}), $1 \le i, j \le n$, rolled into a cylinder, so that the top and bottom rows are glued together. A path is to be threaded from the entry side of the cylinder to the exit side, subject to the restriction that from the given square (i,j) it is possible to move to $(i + 1, j)$, $(i+1, j-1)$ or $(i+1, j+1)$. The path may begin at any position on the entry side and end at any position on the exit side. The cost of such a path is the sum of the integers in the squares through which it passes. We have to find out the minimum cost path.

a. Write a recursive solution to this problem and analyse its complexity.
b. Write a dynamic programming solution and show its complexity to be $\theta(n^2)$

■ **Answer:**

a. Recursive solution

```
int recursivesearch(int A[][], int n, int i, int j)
{int a,b,c;
if(i>n) return 0;
else{
a=recursivesearch(i+1,j);
b=recursivesearch(i+1,j-1);
c=recursivesearch(i+1,j+1);
return A[i][j]+min(a,b,c);
}
}
int Bestpath()
{
int best=32767;
for(row=0;row<n;row++){
best=min(best, resursivesearch(0,row);
}
return best;
}
```

The above recursive function makes three recursive calls. Also, bestpath method runs for n times. Thus, complexity can be said as $O(n3^n)$.

b. Dynamic programming approach

The following is the dynamic programming solution

```
dynamicsearch(int A[][],int n)
```

```
{
for(j=0;j<n;j++)
cost[n-1][j]=A[i][j];
for(i=n-2;i>=0;i--)
for(j=0;j<n;j++)
cost[i][j]=A[i][j]+min(cost[i+1][j], cost[i+1][j+1%n],cost[i+1][j+n-1%n]);
}
```

If you observe the above code, we may find its complexity as $O(n^2)$.

■ **Example** A University offers a series of courses in various levels of difficulties. A student 'graduates' from a program by finishing a Compulsory Discipline Course (CDC) eg., CDC0005. But before he takes CDC0005, he needs to finish either CM0022 or CM0017 first. CM0022 is elementary and has no prerequisite. However, CM0017 would need the student to take either CM0009 or CM0020 first.

Let $S = s_1, s_2, s_3, \ldots s_n$ be the courses provided. And $k_1, k_2, \ldots k_n$ are the durations of the courses in number of weeks. Each course s_i may have two prerequisites: $s_{p(i)}$ and $s_{q(i)}$ or otherwise it has no prerequisites: $p(i) = q(i) = 0$. Formulate a recursive function to compute the shortest time that a new student can graduate from a program. (Assume that the courses can be started anytime whenever there is a request, and each student cannot take more than one course at the same time.)

■ **Answer:**

Let d(s*i*) be the shortest time that a new student can finish a course:

$$d(s_i) = \begin{cases} k_i & \text{if } p(i)=q(i)=0 \\ \min (k_i+d(s_{p(i)}), k_i+d(s_{q(i)})) & \text{otherwise} \end{cases}$$

Dynamic programming solution

DYNAMIC_SHORTEST_GRAD(program_compulsory_course)

Sort the courses according to increasing difficulties.

For each course s_i taken according to this order

if (p_i=q_i=0)

d_i=k_i

else $d_i = \min (k_i + d_{Pi}, k_i+d_{qi})$

if s_i = program_compulsory_course

return d_i

return "input_error"

Memorization solution

MEMORIZATION_SHORTEST_GRAD(program_compulsory_course_id)

1 For i=1 to n

2 di=∞

3 return LOOKUP_SHORTEST_FINISH_COURSE (program_compulsory_course_id)

LOOKUP_SHORTEST_FINISH_COURSE(course_id)

1 if $d_{course\_id} < \infty$

2 return dcourse_id

3 if ($p_{course\_id} = q_{course\_id} = 0$)

4 $d_{course\_id} = k_i$

5 else

6 $d_{course\_id} = \min (k_{course\_id} + d_{Pcourse\_id}, k_{course\_id} + d_{qcourse\_id})$

7 return d_{course_id}

Memorization solution is more efficient since it needs not solve for those courses that won't be involved. Also, no sorting is needed. These should be good reason to use Memorization although it has some minor overhead on table maintenance and recursion.

■ **Example** A ski rental agency has m pairs of skis, where the height of the i^{th} pair of skis is s_i. There are n skiers who wish to rent skis, where the height of the i^{th} skier is h_i. Ideally, each skier should obtain a pair of skis whose height matches his own height as closely as possible. Now, we want to assign skis to skiers so that the sum of the absolute differences of the heights of each skier and his skis is minimised. Formulate a recursive function to solve this problem.

■ **Answer** I:
Suppose the skis and the skiers are shorted according to their heights, so that $h_1 \le h_2 \le \ldots h_n$ and $s_1 \le s_2 \le \ldots s_m$
There is no advantage to do "cross matching".
Let $A_{i,j}$ be the minimum sum of differences to match the first i skiers with the firs j pairs of skis.
Then

$$A_{i,j} = \begin{cases} 0 & \text{if } i=0 \\ \min(A_{i,j-1}, A_{i-1,j-1} + |h_i - s_j|) & \text{if } 1 \le i \le j \\ \infty & \text{if } i>j \end{cases}$$

$A_{n,m}$ is our solution.

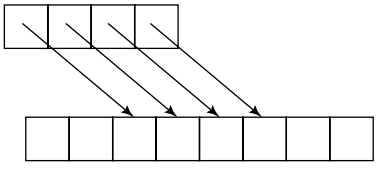
■ **Answer II:**
Suppose the skis and the skiers are shorted according to their heights, so that $h_1 \le h_2 \le \ldots h_n$ and $s_1 \le s_2 \le \ldots s_m$
There is no advantage to do "cross matching".

Define was the 2-D matrix of differences in heights such that $w_{i,j}$ is the difference of the heights of i^{th} skier and the j^{th} pair of skies.

$w_{i,j}$		j^{th} pair of skies							
		0	1	2	3	4	5	..	m–1
i_{th} skier	0	?	?	?	?	?	?	?	?
	1	?	?	?	?	?	?	?	?
	..	?	?	?	?	?	?	?	?
	n-1	?	?	?	?	?	?	?	?

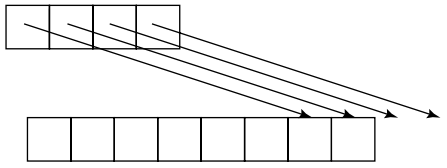
For the matching we may consider:

$w_{i,j}$			j^{th} pair of skies						
		0	1	2	3	4	5	..	m-1
i^{th} skier	0	?	?	?	?	?	?	?	?
	1	?	?	?	?	?	?	?	?
	..	?	?	?	?	?	?	?	?
	n–1	?	?	?	?	?	?	?	?



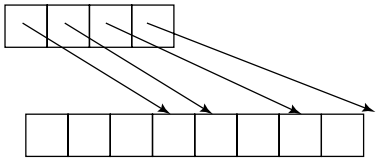
For the matching we may also consider:

$w_{i,j}$		j^{th} pair of skies							
		0	1	2	3	4	5	..	m−1
i^{th} skier	0	?	?	?	?	?	?	?	?
	1	?	?	?	?	?	?	?	?
	..	?	?	?	?	?	?	?	?
	n-1	?	?	?	?	?	?	?	?



For the matching we may even consider:

W_{ij}		j^{th} Pair of skies							
		0	1	2	3	4	5	..	m-1
i^{th} Skier	0	?	?	?	?	?	?	?	?
	1	?	?	?	?	?	?	?	?
	..	?	?	?	?	?	?	?	?
	n-1	?	?	?	?	?	?	?	?



Denote $D_{p,a,q,b}$ as the minimum sum of differences to match *a* skiers starting from p^{th} one: (p^{th}, $(p+1)^{th}$, $(p+2)^{th}$, .. $(p+a-1)^{th}$ skiers) to *b* pairs of skies starting from the q^{th} one: (q^{th}, $(q+1)^{th}$, $(q+2)^{th}$, .. $(q+b-1)^{th}$ skiers)

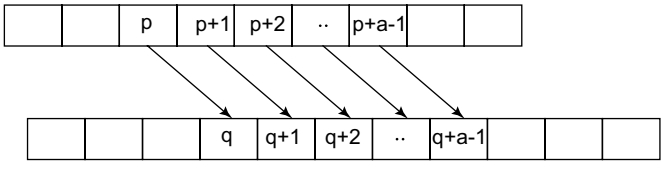
Such that all p,q,a,b are non negative integers and p+a<n, q+b<m.

if a=0, then $D_{p,a,q,b} = 0$

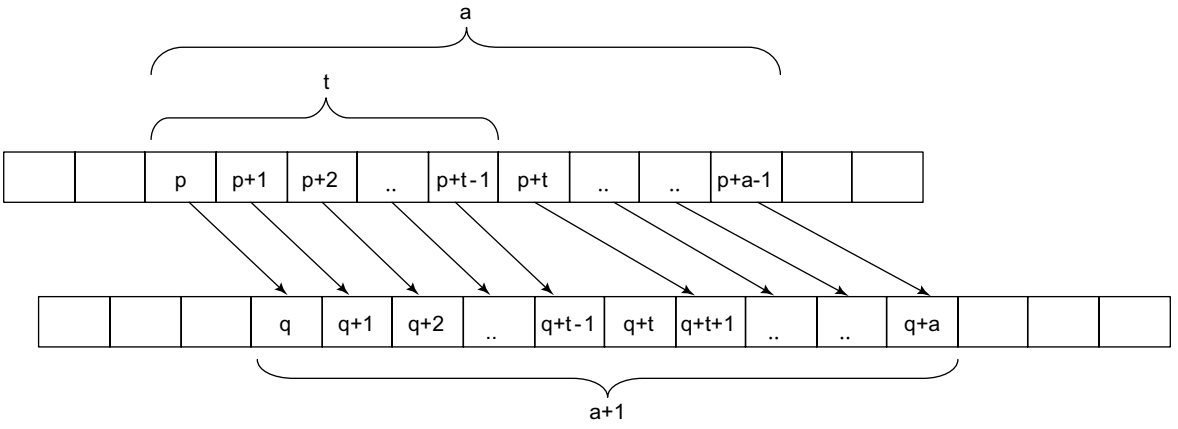
if a>b, then $D_{p,a,q,b} = \infty$

For any p,q,a:

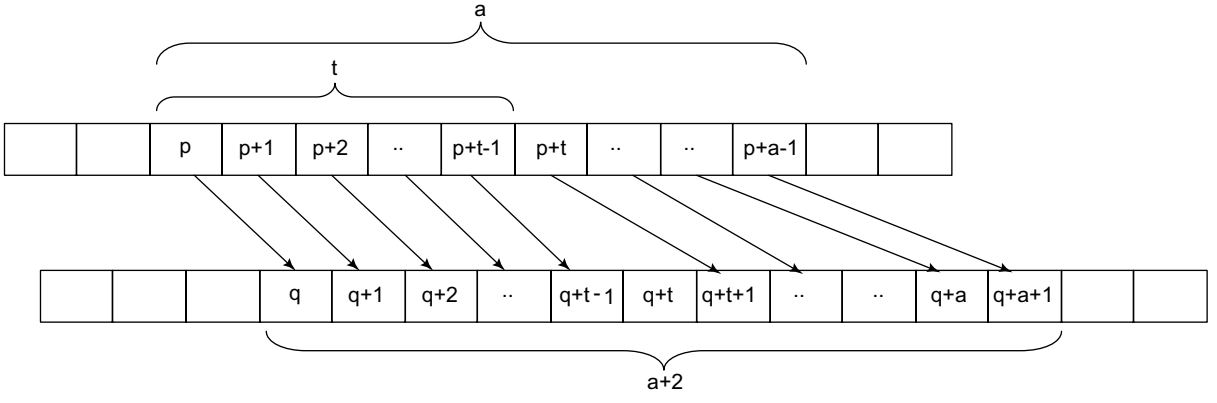
We have $D_{p,a,q,a} = w(p,q) + w(p+1,q+1) + w(p+2,q+2) + .. + w(p+a-1,q+a-1)$ [Easy to calculate]



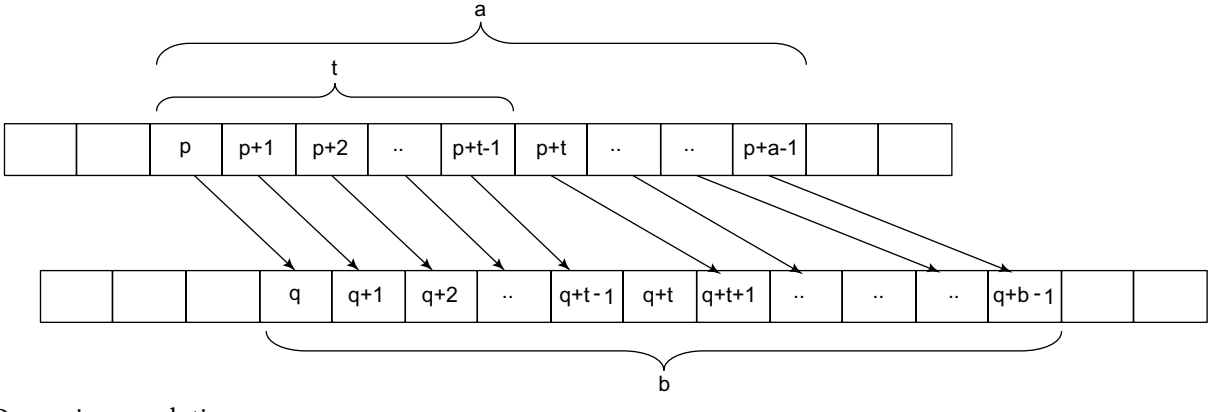
Consider $D_{p,a,q,a+1} = \min_{t=0 \text{ to } a} (D_{p,t,q,t} + D_{p+t,a-t,q+t+1,a-t})$



Consider $D_{p,a,q,a+2} = \min_{t=0 \text{ to } a} (D_{p,t,q,t}+D_{p+t,a-t,q+t+1,a+1-t})$



Similarly: Consider $D_{p,a,q,a+3} = \min_{t=0 \text{ to } a} (D_{p,t,q,t}+D_{p+t,a-t,q+t+1,a+2-t})$
Hence we can see that
$D_{p,a,q,b} = \min_{t=0 \text{ to } a} (D_{p,t,q,t}+D_{p+t,a-t,q+t+1,b-1-t})$



And $D_{0,n,0,m}$ is our solution.

■ **Example:**
A bus company operates k bus routes serving n different bus stops. The i-th route, P_i, passes through m_i stops and can be represented as a sequence of stops, ($v_{i,1} \Rightarrow v_{i,2} \Rightarrow v_{i,3} \Rightarrow \ldots \Rightarrow v_{i,Mi}$), where $v_{i,1}$ and $v_{i,Mi}$ are the first and last stops, respectively. The return route ($v_{i,Mi} \Rightarrow v_{i,Mi-1} \ldots v_{i,1}$) is considered a different route. The bus fare is c_i dollars regardless of where you get on and off along the i-th route. Given 2 bus stops, s and t, we want to find the cheapest cost to travel from s to t. The problem can be specified as follows:

Input:
integer k > 0 specifying the number of routes,

integer n > 0 specifying the number of different bus stops,
array C[1..k] storing the cost of each route,
array R[1..k] of pointers where R[i] points to a linked list containing the sequence of stops for the i-th route,
integers s and t, specifying the start and destination stops.

Output
The cheapest cost to travel from stop s to t.
Design an efficient algorithm to solve the above problem. Present your algorithm in pseudocode. Analyse the running time of your algorithm in terms of n and k. (Hint: Construct a weighted directed graph with n vertices.)

■ **Answer:**
Build a weighted directed graph with n vertices to represent n bus stops. For each pair of vertices, there can be some route that passes through both the source vertices and the target vertices, and that this route pass the source first before reaching the target vertex. If there is such a route or more than one such routes, join that pair of vertices with directed edge, and mark the edge weight as the cheaper route's cost.

Use Bellman-Ford or Dijkstra's algorithm to find shortest path from the source vertex to the target vertex.

```
CHEAPEST_ROUTING(k,n,C,R,,s,t)
1    Build a graph G with n vertices representing the bus stops.
2    for i=1 to k
3       p = first node of R[i]
4.      while p <> NULL
5               q = p.next
6               while q <> NULL
7                       if there is no edge from p.Vertex to q.Vertex
8                         add the edge from p.Vertex to q.Vertex, with weight = C[i]
9                       else
10              if weight_p,q > C[i]
11                              change the weight of the edge from p.Vertex to q.Vertex
12                      q = q.next
13              p = p.next
14   Run Bellman-Ford [O(VE)] or Dijkstra [O(E lg V)] to find shortest path from s to t.
Line 1: O(n)
Line 2-13: O(kn^2)
Line 14: O(n^3) or O(n^2 lg n ) since there are n vertices and O(n^2) edges.
```

■ **Example** Given an array A of n integers: $a_1,a_2,..a_n$, the algorithm Longest(A) is designed to determine the length of the longest non-decreasing consecutive subsequence of integers starting with a_1 (LN1 of A). For example, one such subsequence (LN1 of A) for A=<34,57,53,54,78>, is <34,57>. Hence Longest(A) should return 2. Prove the correctness of this algorithm by using a loop invariant. What are the worst and best cases of the algorithm? Describe the running time of the algorithm in terms of asymptotic tight, upper, and lower bounds

```
Longest(A)
j=1
while j<n and A[j+1]>=A[j]
      j=j+1
return j
```

■ **Answer:** Loop Invariant: At the start of each iteration of the while loop, a_1, a_2, a_3, .. a_j are non-decreasing consecutive subsequence of A starting with a_1.

Initialisation: Before the first iteration, j=1, then there is only one integer a_1, in a_1, a_2, a_3, .. a_j, which is non-decreasing consecutive subsequence.

Maintenance: In each iteration, the cycle is executed only if $a_{j+1}>=a_j$, then j is incremented by one. So a_{j-1}, a_j is non-decreasing. Hence, if the loop invariant is true before an iteration, it remains true before next iteration.
Termination: The while loop continues whenever $a_{j+1}>=a_j$ and j<n. It ends with j=n, or $a_{j+1}<a_j$. In former case a_1, a_2, a_3, .. a_j, is the longest possible LN1 of A. In latter case, a_{j+1} is not involved in LN1 of A. Then a_1, a_2, a_3, .. a_j is LN1 of A.
Best case: LN1 of A only has one element. $\theta(1)$
Worse case: LN1 of A only has n element. $\theta(n)$
Algorithm: $O(n)$, $\Omega(1)$

2.4.10 B –Trees

A **B-tree of order *m*** has the following properties:

- The root has between 2 and *m* children.
- All other internal nodes have between $\lceil m/2 \rceil$ and *m* children.
- All leaves are at the same depth.

WHEN TO USE B-TREES?

- Used for databases which are large, therefore stored on disc (slow access) rather than main memory (fast access).
- Suppose we want to search for a record. While it is slow to go to the disc to find a leaf, it is relatively fast to do the comparisons to determine which branch to follow.
- So we want short branches; thus a record can thus be found in relatively few disc accesses.
- Therefore m must usually be quite large, typically 32 – 256.
- Storing only the keys in the nodes (and not the records) means that we can even store the first level or two of the tree in main memory.

In a B-tree, *all data is stored in the leaves*. We require that each leaf node contains between $\lceil m/2 \rceil$ and *m* keys (these are the actual records) and they are arranged in order.

Do remember that the numbers in the internal nodes are only the keys for the children of those nodes – they are not the records.

To help remember this, draw an internal node as an *ellipse*, and a leaf as a *rectangle*.

As with any data storage procedure, we must be able to (among other things):

1. Locate any record.
2. Insert a new record.
3. Delete any record.
4. Sort the records.

These can be complicated for the B-tree, because there can be many children of any given node. Because all the leaves are on the same level, we don't have to do rotations to shorten branches, as we did for BSTs. In each internal node of the B-tree we store a sequence of numbers to indicate the keys stored in each of the subtrees of that node.

If a node has 3 subtrees T_1, T_2, T_3 then all the keys in T_1 must be less than all the keys in T_2, and all the keys in T_2 must be less than all the keys in T_3.

■ **Example:** In an *order 3* B-tree, if a node contains [10; 20]

then subtree T_1 contains keys less than 10,

subtree T_2 contains keys ≥ 10 and < 20,

and subtree T_3 contains keys ≥ 20.

If a node contains [10; --]

then there are only 2 subtrees and the left subtree contains only keys less than 10 and the right subtree contains keys ≥ 10.

How do we locate a record in a B-tree.

■ **Answer:** Virtually the same way as for a BST. The only difference is that at each node, you usually have more than 2 choices for which child to go to. Note that the keys in any node are always listed in increasing order, including the records in the leaves.

How do we insert a new record in a B-tree?

■ **Answer:** It can be harder, because the new record must go into a leaf on the same level as all others.

INSERTING a number X

- proceed as if to **locate** X.
- If there is room in the leaf node where X would have been located, insert X there.
- If there is not room, split the leaf node into 2 nodes and divide the records between the 2 nodes.
- Adjust the keys in the parent node to show the new nodes.
- If there are already *m* leaf nodes, first split the parent node into 2 nodes. (adjusting the keys in its parent node)
- Repeat this last step if necessary.

How do we delete a record from a B-tree?

■ **Answer:** Locate it first, then delete it. The only problem might be that there are too few records in the leaf.

DELETING a number X:

- Locate X.
- Delete the record X.
- If too few records in the leaf, combine the leaf with a sibling, if there is room.
- If there is not room, borrow a record from the sibling.
- If combining the leaf with its sibling made too few leaves repeat this procedure on the parent node.
  (i.e. merge parent with its sibling or borrow a child from its sibling)

2.4.11 A Note on String or Pattern Matching

Sub-string Searching

The problem is to search for a pattern string, **P[0..m-1],** in a text string **T[0..n-1].** Usually n>>m, and T might be very long indeed, although this is not necessarily so. This problem occurs in text-editors and many other computer applications.

Naive Search

The naive string searching algorithm is to examine each position, i>=0, in T, trying for equality of P[0..m-1] with T[i..i+m-1]. If, there is mismatch, position i+1 is tried, and so on.

The worst-case time-complexity of the naive algorithm is seen to be O(m*n), e.g. $P=a^{m-1}b$ and $T=a^{n-1}b$ (i.e., P = aaaab and T = aaaaaaaaaaab).

■ **Example** Naïve pattern (string) matching program.

```
int main(){
  char P[80],T[80];
  int i,j,k, m,n;
  printf("Enter Pattern and String\n");
  scanf("%s%s", P, T);
  n=strlen(T);
  m=strlen(P);
  k=n-m+1;
  for(i=0;i<k;i++){
        for(j=0;j<m; j++) if( T[i+j] !=P[j]) break;
        if(j==m) break;
}
(j==m)? printf("Found at %d\n", i): printf("Not Found\n");
return 0;
}
```

```
 T  0 0 0 0 1 0 0 0 1 0 1 0 0 0 1
0   0 0 0 1
          ^mismatch
1     0 0 0 1
            ^match
2       0 0 0 1
              ^mismatch
3         0 0 0 1
                ^mismatch
4           0 0 0 1
                  ^mismatch
5             0 0 0 1
                    ^match
```

Rabin Karp Algorithm

Here, the key idea is to think of the pattern **P[0..m-1]** as a key, transform it into an equivalent integer *p*. Similarly, we transform substrings in the text string T[] into integers. For s=0,1,…,n-m, transform **T[s..s+m-1]** to an equivalent integer **ts**. The pattern occurs at position s if and only if **p=ts**. If we compute p and **ts** quickly, then the pattern matching problem is reduced to comparing p with n-m+1 intgers.

Now, the question is "how to calculate integer p given search pattern P?" We can consider the string P as an integer considering like the following:

$\mathbf{p = 2^{m-1}\,P[0] + 2^{m-2}\,P[1] + \ldots + 2^{1}\,P[m-2] + 2^{0}P[m-1]}$

Remember, P[0], P[1], etc., are elements of the search array and for alphabetic strings we consider their ASCII codes. However, the calculation of p needs many multiplications. We can use Horner's rule to make its calculation as simple such as:

$p = P[m-1] + 2 * (P[m-2] + 2* (P[m-3] + \ldots 2* (P[1] + 2* (P[1] + 2*P[0] \ldots).$

This takes O(m) time, assuming each arithmetic operation can be done in O(1) time.

Similarly, to compute the (n-m+1) integers ts from the text string, T. Here, also we need O(m) operations to calculate each number. Thus, we need total computations in the order of O((n-m+1)m). This is little costly.

```
for (s = 0; s <=n - m; s++){
  t[s] = 0;
  for (i = 0; i < m; i++)
         t[s] = 2*t[s] + T [s+i];
}
```

A better method to compute the integers incrementally using previous result:

```
t[0] = 0;
offset = 1;
for (i = 0; i < m; i++)
         offset = 2*offset;
for (i = 0; i < m; i++)
         t[0] = 2*t[0] + T [i]
for (s = 1; s < n - m; s++)
         t[s] = 2*t[s-1] - offset *T [s-1]) + T [s+m -1];
```

This takes O(n+m) time, assuming that each arithmetic operation can be done in O(1) time.

The problem with the previous strategy is that when m is large, it is unreasonable to assume that each arithmetic operation can be done in O(1) time. In fact, given a very long integer, we may not even be able to use the default integer type to represent it. Therefore, modulo arithmetic is used. Let q be a prime number so that 2q can be stored in one computer word. This makes sure that all computations can be done using single-precision arithmetic. Once we use the modulo arithmetic, when p=$\mathbf{t}_s$ for some s, we can no longer be sure that P[0 .. m-1] is equal to T[s .. s+ m -1]. Therefore, after the equality test

$p = t_s$, we should compare P[0..m-1] with T[s..s+m-1] character by character to ensure that we really have a match. So the worst-case running time becomes O(nm), but it avoids a lot of unnecessary string matching's in practice.

```
p = 0;
for (i = 0; i < m; i++)
   p = (2*p + P[i]) % q;
t[0] = 0;
offset = 1;
for (i = 0; i < m; i++)
   offset = 2*offset = % q;
for (i = 0; i < m; i++)
   t[0] = (2*t[0] + T[i])% q;
for (s = 1; s <= n - m; s++)
   t[s] = 2*(t[s-1] - offset *T[s-1] + T[s + m - 1]) %q;
```

Knuth-Morris-Pratt Algorithm

The Knuth-Morris-Pratt or the Morris-Pratt algorithms are extensions of the basic Brute Force algorithm. They use pre-computed data to skip forward not by 1 character, but by as many as possible for the search to succeed.

To illustrate the logic behind algorithm's, we work through a (relatively artificial) run of the algorithm. At any given time, the algorithm is in a state determined by two integers, i and j, which denote respectively the position within T which is the beginning of a prospective *match* for P, and the *index* in P denoting the character currently under consideration. This is depicted, at the start of the run, like

i: 01234567890123456789012
T: ABC ABCDAB ABCDABCDABDE
P: ABCDABD
j: 0123456

We proceed by comparing successive characters of P to "parallel" characters of T, moving from one to the next if they match. However, in the fourth step, we get T[3] is a space and P[3] = 'D', a mismatch. Rather than beginning to search again from T[1], we note that no 'A' occurs between positions 0 and 3 in T except at 0; hence, having checked all those characters previously, we know there is no chance of finding the beginning of a match if we check them again. Therefore, we move on to the next character, setting i = 4 and j = 0.

i: 01234567890123456789012
T: ABC ABCDAB ABCDABCDABDE
P: ABCDABD
j: 0123456

We quickly obtain a nearly complete match "ABCDAB" when, at P[6] (T[10]), we again have a discrepancy. However, just prior to the end of the current partial match, we passed an "AB" which could be the beginning of a new match, so we must take this into consideration. As we already know that these characters match the two characters prior to the current position, we need not check them again; we simply reset i = 8, j = 2 and continue matching the current character. Thus, not only do we omit previously matched characters of T, but also previously matched characters of P.

To give little more emphasis about the logic behind this algorithm, consider the prefixes and suffixes of the matched part.

Proper Prefixes of ABCDAB	Proper Suffixes of ABCDAB
A	B
AB	AB
ABC	DAB
ABCD	CDAB
ABCDA	BCDAB

We may find that "AB" is the largest suffix which is having equivalent prefix. With respect to current position, what is largest suffix of string T which is prefix of P. In this case, it is AB. Ofcourse, this can be calculated with P alone as when match occurs both the contents of T and P will be same till that location. That is, when a mismatch occures at the 6^{th} location of the string P then we need to required to check afresh again.

Now continue our search again:

i: 01234567890123456789012
T: ABC ABCDAB ABCDABCDABDE
P: ABCDABD
j: 0123456

This search fails immediately, however, as the pattern still does not contain a space, so as in the first trial, we return to the beginning of P and begin searching at the next character of T: i = 11, reset j = 0.

i: 01234567890123456789012
T: ABC ABCDAB ABCDABCDABDE
P: ABCDABD
j: 0123456

Once again we immediately hit upon a match "ABCDAB" but the next character, 'C', does not match the final character 'D' of the word P. Reasoning as before, we set i = 15, to start at the two-character string "AB" leading up to the current position, set j = 2, and continue matching from the current position.

i: 01234567890123456789012
T: ABC ABCDAB ABCDABCDABDE
P: ABCDABD
j: 0123456

Now, we have achieved matching.

The above example contains all the elements of the algorithm. For the moment, we assume that the existence of a "partial match" table Next, described below, which indicates where we need to look for the start of a new match in the event that the current one ends in a mismatch. The entries of Next are constructed so that if we have a match starting at T[i] that fails when comparing T[i + j] to P[j], then the next possible match will start at index i + j – Next[j] in T (that is, Next[j] is the amount of "backtracking" we need to do after a mismatch). This has two implications: first, Next[0] = –1, which indicates that if P[0] is a mismatch, we cannot backtrack and must simply check the next character; and second, although the next possible match will *begin* at index i + j – Next[j], as in the example above, we need not actually check any of the Next[j] characters after that, so that we continue searching from P[Next[j]].

Calculation of Next table needs only search string P. Let us consider our example of P = "ABCDABD" and see how Next creation proceeds. We set Next[0] = –1. To find Next[1], we must discover a proper suffix of "A" which is also a prefix of P. But there are no proper suffixes of "A", so we set Next[1] = 0. Likewise, Next[2] = 0.

Continuing to Next[3], we note that there is a shortcut to checking *all* suffixes: let us say that we discovered a proper prefix ending at P[2] with length 2 (the maximum possible); then its first character is a proper prefix of a proper prefix of P, hence a proper prefix itself, and it ends at P[1], which we already determined cannot occur. Therefore we need not even concern ourselves with substrings having length 2, and as in the previous case the sole one with length 1 fails, so Next[3] = 0.

We pass to the subsequent P[4], 'A'. The same logic shows that the longest substring we need consider has length 1, and although in this case 'A' *does* work, recall that we are looking for segments ending *before* the current character; hence Next[4] = 0 as well.

Considering now the next character, P[5], which is 'B', we exercise the following logic: if we were to find a sub-pattern beginning before the previous character P[4], yet continuing to the current one P[5], then in particular it would itself have a proper initial segment ending at P[4] yet beginning before it, which contradicts the fact that we already found that 'A' itself is the earliest occurrence of a proper segment ending at P[4]. Therefore, we need not look before P[4] to find a terminal string for P[5]. Therefore Next[5] = 1.

Finally, we see that the next character in the ongoing segment starting at P[4] = 'A' would be 'B', and indeed this is also P[5]. Furthermore, the same argument as above shows that we need not look before P[4] to find a segment for P[6], so that this is it, and we take Next[6] = 2.

Therefore we compile the following table:

i	0	1	2	3	4	5	6
W[i]	A	B	C	D	A	B	D
T[i]	–1	0	0	0	0	1	2

■ **Example** KMP Algorithm implementation.

```
void preComputeData(char *x, int m, int Next[]) {
int i, j;
i = 0;
j = Next[0] = -1;
while (i < m)
{
while (j > -1 && x[i] != x[j]) j = Next[j];
Next[++i] = ++j;
}
}
void MorrisPrat(char *x, int m, char *y, int n) {
int i, j, Next[1000];
/* Preprocessing */
preComputeData(x, m, Next);
/* Searching */
i = j = 0;
while (j < n) {
while (i > -1 && x[i] != y[j])i = Next[i];
i++;
j++;
if (i >= m) {
printf("\nMatch found at : [%d]\n",j - i);
i = Next[i];
}
}
}
int main(){
char T[80], P[30];
printf("Enter Two strings\n");
scanf("%s%s",T, P);
   MorrisPrat(P,strlen(P),T,strlen(T));
printf("\n\n");
return(0);
}
```

This is how the comparison happens visually assuming T="herepheroero" and P="hero".

hereroheroero
!
hero
hereroheroero
!
hero

hereroheroero
!
hero
hereroheroero
!
hero
hereroheroero
|||| -----> Match found!
hero

hereroheroero
!
hero

Boyer-Moore Algorithm

The first key idea, and it is a good one, is that if the m^{th} character ch=T[m] (numbered from `1' upwards) does not occur in pat at all then any instance of P in T must start at position m+1 or later.

For example, if searching for P=`freddy', in T = `I floated lonely as a cloud', the 6^{th} character of T is `a' which is not in {d, e, f, r, y} and there is no need to consider positions 1 to 6 of T any more! In this way, we can move along T in steps of m positions, i.e. k = m, 2m, 3m, ..., provided we are lucky.

The second idea is that if the *last* occurrence of the character T[k] in pat is $delta_1$[ch] positions from the right hand end of P, then pat can be slid that many positions to the right before we *might* get a match in T. If ch=T[k] does not occur in P, set $delta_1$[ch] equal to m.

e.g., $delta_1$['e']=3, $delta_1$['f']=5, $delta_1$['d']=1, $delta_1$['r']=4, $delta_1$['y']=0.

In general if the last j characters match, i.e. T[k-j..k]=P[m-j..m], but T[k-j-1]~=P[m-j-1], then pat can be "slid" a certain distance along T depending on where, if at all, there is an earlier instance of P[m-j..m] within pat; e.g. consider P=abracadabra, or e.g. P=fababab. An array, $delta_2$[1..m], is used to hold these pre-computed distances.

The Boyer Moore algorithm is the fastest string searching algorithm. Most editors use this algorithm.

It compares the pattern with the actual string from right to left. Most other algorithms compare from left to right. If the character that is compared with the rightmost pattern symbol does not occur in the pattern at all, then the pattern can be shifted by m positions behind this text symbol.

The Boyer Moore algorithm is *always* fast having worst-case time-complexity O(m+n), but on natural-language text it actually gets faster as m increases to a certain extent, e.g. Boyer and Moore suggesting O(n/4)-time on *average* for m=5. The following example illustrates this situation.

■ **Example:**

0 1 2 3 4 5 6 7 8 9 ...
a b b a d a b a c b a
| |
b a b a c |
<------ |
|
b a b a c

The comparison of "d" with "c" at position 4 does not match. "d" does not occur in the pattern. Therefore, the pattern cannot match at any of the positions 0,1,2,3,4, since all corresponding windows contain a "d". The pattern can be shifted to position 5. The best case for the Boyer-Moore algorithm happens if, at each search attempt the first compared character does not occur in the pattern.

Bad character heuristics

This method is called bad character heuristics. It can also be applied if the bad character (the character that causes a mismatch), occurs somewhere else in the pattern. Then the pattern can be shifted so that it is aligned to this text symbol. The next example illustrates this situation.

■ **Example:**

```
0 1 2 3 4 5 6 7 8 9 ...
a b b a b a b a c b a
|
b a b a c
<----
|
b a b a c
```

Comparison between "b" and "c" causes a mismatch. The character "b" occurs in the pattern at positions 0 and 2. The pattern can be shifted so that the rightmost "b" in the pattern is aligned to "b".

Good suffix heuristics

Sometimes the bad character heuristics fails. In the following situation the comparison between "a" and "b" causes a mismatch. An alignment of the rightmost occurence of the pattern symbol a with the text symbol a would produce a negative shift. Instead, a shift by 1 would be possible. However, in this case it is better to derive the maximum possible shift distance from the structure of the pattern. This method is called good suffix heuristics.

■ **Example:**

```
0 1 2 3 4 5 6 7 8 9 ...
a b a a b a b a c b a
| | |
c a b a b
<----
| | |
c a b a b
```

The suffix "ab" has matched. The pattern can be shifted until the next occurence of ab in the pattern is aligned to the text symbols ab, i.e. to position 2.

In the following situation the suffix "ab" has matched. There is no other occurence of "ab" in the pattern.Therefore, the pattern can be shifted behind "ab", i.e. to position 5.

■ **Example:**

```
0 1 2 3 4 5 6 7 8 9 ...
a b c a b a b a c b a
| | |
c b a a b
c b a a b
```

In the following situation the suffix "bab" has matched. There is no other occurence of "bab" in the pattern. But in this case the pattern cannot be shifted to position 5 as before, but only to position 3, since a prefix of the pattern "ab" matches the end of "bab". We refer to this situation as case 2 of the good suffix heuristics.

■ **Example:**

```
0 1 2 3 4 5 6 7 8 9 ...
a a b a b a b a c b a
| | | |
a b b a b
a b b a b
```

The pattern is shifted by the longest of the two distances that are given by the bad character and the good suffix heuristics.

The Boyer-Moore algorithm uses two different heuristics for determining the maximum possible shift distance in case of a mismatch: the "bad character" and the "good suffix" heuristics. Both heuristics can lead to a shift distance of m. For the bad character heuristics this is the case, if the first comparison causes a mismatch and the corresponding text symbol does not occur in the pattern at all. For the good suffix heuristics this is the case, if only the first comparison was a match, but that symbol does not occur elsewhere in the pattern.

Time Complexity

The worst-case time-complexity of the Boyer-Moore algorithm is O(m+n).

The algorithm often runs in O(n/m)-time on natural-language text for small values of m. Note that if T is in slow, block-access, backing store, it is generally not possible to bring just every m^{th} character into main memory, so the time-complexity is then O(n).

Space Complexity

The space-complexity is O(m + |alphabet|), for the arrays delta1[ ] and delta2[ ].

Preprocessing T.

It is intuitively obvious that the worst-case for searching for an arbitrary pattern, P, in a text, T, must take at least O(n)-time, where n=|T|. However, if some time is spent *pre-processing* T, then individual searches can be made more quickly, e.g. in O(m)-time using a suffix-tree, where m=|P|. It takes O(n)-time to build a suffix tree for T.

2.4.12 Convex Hull problem

Convex hull of a set of points is a convex polygon which embodies all the points. Graham's algorthim is O(nlogn) algorithm to find convex hull. Consider the points as shown in Fig. 2.53 (A, 0, 0) (B, –5, –2) (C, –2, –1) (D, –6, 0) (E, –3.5, 1) (F, – 4.5, 1.5) (G, –2.5, –5) (H, 1, –2.5) (I, 2.5, 2.5) (J, –2.2, 2.2)

Graham Scan, as it is called, works by picking the lowest point p, i.e. the one with the minimum p.y value (note this must be on the convex hull), and then scanning the rest of the points in counterclockwise order with respect to p. As this scanning is done, the points that should remain on the convex hull, are kept, and the rest are discarded leaving only the points in the convex hull at the end. To see how this is done, imagine first that, by luck, all the points scanned are actually on the convex hull. In that case, every time we move to a new point we make a left turn with respect to the line determined by the last two points on the hull. Therefore, what Graham Scan does, is to check if the next point is really a left turn. If it is NOT a left turn, then it backtracks to the pair of points from which the turn would be a left turn, and discards all the points that it backs up over. Because of the backtracking, we implement the algorithm with a stack of points.

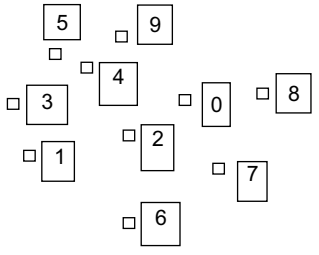


Figure 2.53

The array of input points is shown above labeled by index in the array (rather than their char label). The point labeled A is in index 0, B is in index 1, etc. The lowest point is computed and swapped with the point in index 0 of the array, as shown in Fig. 2.54.

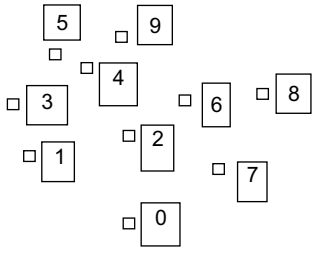


Figure 2.54

The points are then sorted by their polar angles with respect to the lowest point .

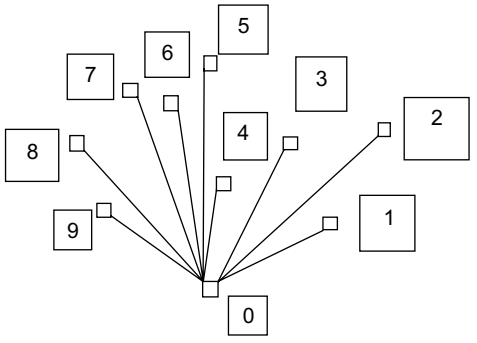


Figure 2.55

The points are sorted and rearranged in the array as shown in Fig. 2.55. The turn from line 0-1 to point 2 is left, from 1-2 to 3 is left, from 2-3 to 4 is left. Now the stack contains the points 01234. This represents the partial hull in Fig. 2.56.

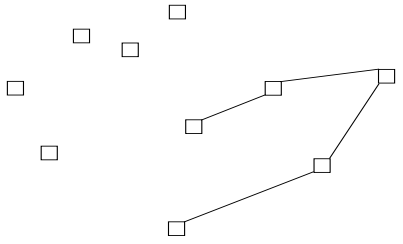


Figure 2.56

The turn from line 3-4 to point 5 is right, so we pop the stack. The turn from 2-3 to 5 is right, so we pop again. The turn from 1-2 to 5 is left, so we push 5 on the stack. The stack now has 0125, and the picture looks as shown in Fig. 2.57.

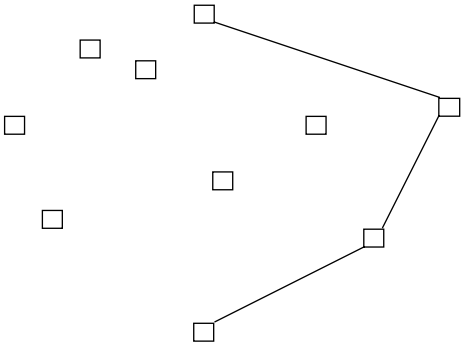


Figure 2.57

The turn from line 2-5 to 6 is left so 6 is pushed on the stack. Then the turn from 5-6 to 7 is right, so 6 is popped and 7 is pushed because the turn from line 2-5 to 7 is left. The rest of the turns are left, so 8 and 9 are pushed on the stack. The final stack is 0125789, and the convex hull is shown in Fig. 2.58.

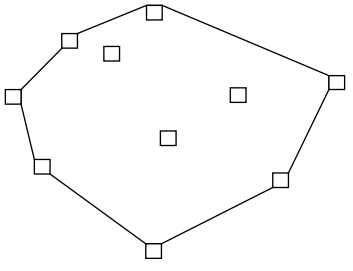


Figure 2.58

Graham Scan Pseudo-code: The algorithm takes an array of points and returns an array of points representing the convex hull.

1. Find the lowest point p, (the point with the minimum y coordinate). If there is more than one point with the minimum y coordinate, then use the leftmost one.
2. Sort the remaining points in counterclockwise order around p. If any points have the same angle with respect to p, then sort them by increasing distance from p.
3. Push the first 3 points on the stack.
4. For each remaining point c in sorted order, do the following:
   b = the point on top of the stack.
   a = the point below that on the stack.
   While a left turn is NOT made while moving from a to b to c do pop the stack.
   b = the point on top of the stack.
   a = the point below that on the stack.
   Push c on the stack.
5. Return the contents of the stack.

The complexity of Graham Scan is O(n log n). It means that the number of steps in the algorithm is bounded asymptotically by a constant times n log n where n is the number of points in the input set. It is true because the most costly step is the sorting in step 2. This is O(n log n). Step 1 takes time O(n). Step 3 takes O(1). Step 4 is trickier to analyze. It is important to notice that although each of the O(n) points are processed, and each might in the worst case have to pop the stack O(n) times, overall this does NOT result in $O(n^2)$. This is because overall, every point is added to the stack exactly once and is removed at most once. So the sum of all the stack operations is O(n).

There are many O(n log n) and $O(n^2)$ algorithms for the convex hull problem, just as there are both for sorting. For the convex hull there is also an algorithm (Jarvis Algorithm) that runs in O(nh), where n is the number of points in the set, and h is the number of points in the convex hull. For small convex hulls (smaller than log n) this algorithm is faster than n log n, and for large convex hulls it is slower.

2.4.13 A Brief Discussion for NP-complete Problem

Deterministic algorithms are where no step is random. If the program/algorithm chooses a step non-deterministically (by some extraneous influence to the algorithm!) such that it is always the right choice, then such an algorithm is called **non-deterministic algorithm.**

Decision problem: A problem whose solution is simply "yes" or "no".

Optimisation problem: A problem that searches for a solution with maximum or minimum value

■ **Examples:**

Traveling salesperson problem
Knapsack problem
Bin Packing problem
Clique problem
Vortex cover problem

In general, optimisation problems are more difficult to solve than their corresponding decision problems.

Polynomial vs. Exponential Growth

An algorithm is said to be polynomial complexity if its complexity order is of form $O(n^k)$, for some k value greater than 0. Examples of polynomial problems:

- Sorting: O(n log n) = $O(n^2)$
- All-pairs shortest path: $O(n^3)$
- Minimum spanning tree: O(E log E) = $O(E^2)$

To understand the significance of polynomially bounded algorithms as a class, let us consider the remaining algorithms, those that violate all polynomial bounds. We usually refer to these as *exponential* algorithms, because $2n$ is the paradigm of nonpolynomial rates of growth. Other examples of exponential rates of growth are kn (any fixed $k > 1$), $n!$, $2n^2$, nn and $n^{\log} n$. It is obvious that, when the size of the input grows, any polynomial algorithm will eventually become more efficient than any exponential one (see Table 2.10).

Table 2.10 Growth of polynomial and exponential functions

Function	Corresponding Values		
n	10	100	1000
$n \log n$	33	664	9966
n^3	1000	1,000,000	10^9
$10^6 n^8$	10^{14}	10^{22}	10^{30}
$2n$	1024	1.27×10^{30}	1.05×10^{301}
$n^{\log} n$	2099	1.93×10^{13}	7.89×10^{29}
$n!$	3,628,800	10^{158}	4×10^{2567}

Another important feature of polynomial algorithms is that, in a sense, they take better advantage of technological advances. For example, each time a technological breakthrough increases the speed of computers tenfold, the size of the largest instance a polynomial algorithm can solve in a fixed amount of time will be multiplied by a constant between 1 and 10. In contrast, an exponential algorithm will experience only an *additive* increase in the size of the instance it can solve in a fixed amount of time (Table 2.11). Finally, it should be noted that polynomial algorithms have nice "closure" properties: (1) they may be combined to solve special cases of the same problem; (2) they may invoke another polynomial algorithm as a "subroutine" with the resulting algorithm still being polynomial.

Table 2.11 Polynomial-time algorithms take better advantage of technology

Function	Size of instance solved in one day	Size of instance solved in one day in a computer 10 times faster
n	10^{12}	10^{13}
n log n	0.948×10^{11}	0.87×10^{12}
n^2	10^6	3.16×10^6
n^3	10^4	2.15×10^4
$10^8 n^4$	10	18
2^n	40	43
10^n	12	13
$n^{\log n}$	79	95
$n!$	14	15

P Problems (Tractable Problems)

In computational complexity theory, P, also known as PTIME or DTIME, is one of the most fundamental complexity classes. It contains all decision problems which can be solved by a deterministic Turing machine using a polynomial amount of computation time, or polynomial time.

P is known to contain many natural problems, including the decision versions of linear programming, calculating the greatest common divisor, and finding a maximum matching.

Unsolveable problems for which no algorithm can be given guaranteed to solve all instances of the problem. These problems are also known as Undecideable. Example: the halting problem (given a description of a program and arbitrary input, decide whether the program will halt on that input), demonstrated by Alan Turing.

NP problems (Intractable Problems)

In computational complexity theory, NP is one of the most fundamental complexity classes. The abbreviation NP refers to "nondeterministic polynomial time". This is the class of decision problems which can be solved by a non-deterministic polynomial algorithm. Intuitively, NP is the set of all decision problems for which the 'yes'-answers have simple proofs of the fact that the answer is indeed 'yes'.

More precisely, these proofs have to be verifiable in polynomial time by a deterministic Turing machine. In an equivalent formal definition, NP is the set of decision problems solvable in polynomial time by a non-deterministic Turing machine.

The complexity class P is contained in NP, but NP contains many important problems, the hardest of which are called NP-complete problems, for which no polynomial-time algorithms are known. The most important open question in complexity theory, the P = NP problem, asks whether such algorithms actually exist for NP-complete, and by corollary, all NP problems (Fig. 2.59). It is widely believed that this is not the case.

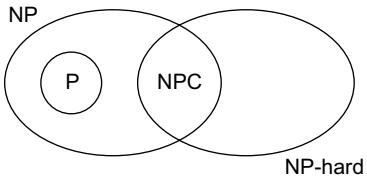


Figure 2.59

NP Complete

In computational complexity theory, the complexity class NP-complete (abbreviated NP-C or NPC), is a class of problems having two properties:

1. Any given solution to the problem can be verified quickly (in polynomial time); the set of problems with this property is called NP (nondeterministic polynomial time).
2. If the problem can be solved quickly (in polynomial time), then so can every problem in NP.

Although any given solution to such a problem can be verified quickly, there is no known efficient way to locate a solution in the first place; indeed, the most notable characteristic of NP-complete problems is that no fast solution to them is known. That is, the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows. As a result, the time required to solve even moderately large versions of many of these problems easily reaches into the billions or trillions of years, using any amount of computing power available today. NP-complete problems are often addressed by using approximation algorithms.

NP-complete is a subset of NP, the set of all decision problems whose solutions can be verified in polynomial time; NP may be equivalently defined as the set of decision problems that can be solved in polynomial time on a nondeterministic Turing machine. A problem p in NP is also in NPC if and only if every other problem in NP can be transformed into p in polynomial time. NP-complete can also be used as an adjective: problems in the class NP-complete are known as NP-complete problems.

A decision problem C is NP-complete if:

C is in NP, and every problem in NP is reducible to C in polynomial time.

The list below contains some well-known problems that are NP-complete when expressed as decision problems.

Boolean satisfiability problem (Sat.)

N-puzzle

Knapsack problem

Hamiltonian path problem

Travelling salesman problem

Subgraph isomorphism problem

Subset sum problem

Clique problem

Vertex cover problem

Independent set problem

Dominating set problem

Graph coloring problem

The following techniques can be applied to solve computational problems in general, and they often give rise to substantially faster algorithms:

Approximation: Instead of searching for an optimal solution, search for an "almost" optimal one.

Randomisation: Use randomness to get a faster average running time, and allow the algorithm to fail with some small probability.

Restriction: By restricting the structure of the input (e.g., to planar graphs), faster algorithms are usually possible.

Parameterisation: Often there are fast algorithms if certain parameters of the input are fixed.

Heuristic: An algorithm that works "reasonably well" in many cases, but for which there is no proof that it is both always fast and always produces a good result. Metaheuristic approaches are often used.

NP Hard

NP-hard (non-deterministic polynomial-time hard), in computational complexity theory, is a class of problems that are, informally, "at least as hard as the hardest problems in NP". A problem H is NP-hard if and only if there is an NP-complete problem L that is polynomial time Turing-reducible to H. It is super set of NP complete.

NP-hard problems may be of any type–decision problems, search problems, or optimization problems

NP-complete means problems that are complete in NP (i.e., any problem reduces to any other problem in polynomial time)

NP-hard–stands for at least as hard as NP (but not necessarily in NP)

NP-easy–stands for at most as hard as NP (but not necessarily in NP)

NP-equivalent–means equally difficult as NP, (but not necessarily in NP)

Reductions vs. Transformations

"P1 polynomially reduces to P2" if P1 can be solved in polynomial time using an algorithm for P2 as a subroutine (each call to the subroutine being counted as 1, so the number of such calls has to be polynomial). This is the Turing reduction concept.

"P1 polynomially transforms to P2" if we can do the same as above but with the limit of only one call to the subroutine.

Definition 1 of NP: A problem is said to be Non-deterministically Polynomial (NP) if we can find a non-deterministic Turing machine that can solve the problem in a polynomial number of non-deterministic moves.

For those who are not familiar with Turing machines, two alternative definitions of NP are commonly used.

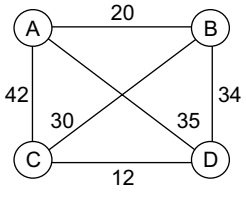
Definition 2 of NP: A problem is said to be NP if

1. its solution comes from a finite set of possibilities, and
2. it takes polynomial time to verify the correctness of a candidate solution

Definition 3 of NP: A problem is said to be NP if there exists an NP algorithm for it.

The Travelling Salesman Problem (TSP)

(TSP) is a problem in combinatorial optimisation studied in operations research and theoretical computer science. Given a list of cities and their pair-wise distances, the task is to find a shortest possible tour that visits each city exactly once. The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of micro-chips. Slightly modified, it appears as a sub-problem in many areas, such as DNA sequencing. In these applications, the concept city represents, for example, customers, soldering points, or DNA fragments, and the concept distance represents travelling times or cost, or a similarity measure between DNA fragments. In many applications, additional constraints such as limited resources or time windows make the problem considerably harder.



Hamiltonian Cycle Problem

A Hamiltonian cycle is a round trip path along n edges of G which visits every vertex once and returns to its starting vertex. Consider the example shown in Fig. 2.60.

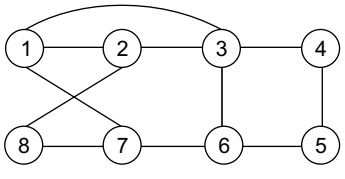


Figure 2.60

Hamiltonian cycle : 1, 2, 8, 7, 6, 5, 4, 3, 1.

In the mathematical field of graph theory the Hamiltonian path problem and the Hamiltonian cycle problem are problems of determining whether a Hamiltonian path or a Hamiltonian cycle exists in a given graph (whether directed or undirected). Both problems are NP-complete. There is a simple relation between the two problems. The Hamiltonian path

problem for graph G is equivalent to the Hamiltonian cycle problem in a graph H obtained from G by adding a new vertex and connecting it to all vertices of G. The Hamiltonian cycle problem is a special case of the traveling salesman problem, obtained by setting the distance between two cities to a finite constant if they are adjacent and infinity otherwise.

Vertex Cover Problem

In the mathematical discipline of graph theory, a vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex of the set. The problem of finding a minimum vertex cover is a classical optimisation problem in computer science and is a typical example of an NP-hard optimisation problem that has an approximation algorithm. Its decision version, the vertex cover problem was one of Karp's 21 NP-complete problems and is therefore a classical NP-complete problem in computational complexity theory. Furthermore, the vertex cover problem is fixed-parameter tractable and a central problem in parameterised complexity theory.

Formally, a vertex cover of a graph G is a set of vertices C such that each edge of G is incident to at least one vertex in C. The set C is said to cover the edges of G. Fig. 2.61 shows examples of vertex covers in two graphs (the set C is marked with different shade).

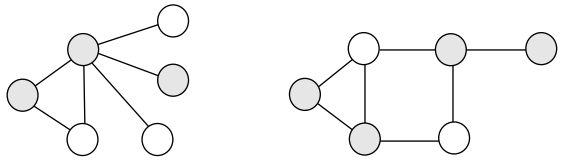


Figure 2.61

A minimum vertex cover is a vertex cover of smallest possible size. The vertex cover number τ is the size of a minimum vertex cover. Fig. 2.62 shows examples of minimum vertex covers in two graphs.

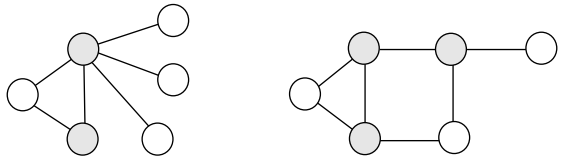


Figure 2.62

Satisfy Ability Problem

Satisfiability is the problem of determining if the variables of a given Boolean formula can be assigned in such a way as to make the formula evaluate to TRUE. Equally important is to determine whether no such assignments exist, which would imply that the function expressed by the formula is identically FALSE for all possible variable assignments.

In this latter case, we would say that the function is unsatisfiable; otherwise it is satisfiable. To emphasize the binary nature of this problem, it is frequently referred to as Boolean or propositional satisfiability. The shorthand "SAT" is also commonly used to denote it, with the implicit understanding that the function and its variables are all binary-valued. In complexity theory, the Boolean satisfiability problem (SAT) is a decision problem, whose instance is a Boolean expression written using only AND, OR, NOT, variables, and parentheses. The question is: given the expression, is there some assignment of TRUE and FALSE values to the variables that will make the entire expression true? A formula of propositional logic is said to be satisfiable if logical values can be assigned to its variables in a way that makes the formula true. The Boolean satisfiability problem is NP-complete. The propositional satisfiability problem (PSAT), which decides whether a given propositional formula is satisfiable, is of central importance in various areas of computer science, including theoretical computer science, algorithmics, artificial intelligence, hardware design, electronic design automation, and verification.

Partition Problem

In computer science, the partition problem is an NP-complete problem. The problem is to decide whether a given multiset of integers can be partitioned into two "halves" that have the same sum. More precisely, given a multiset S of integers, is there a way to partition S into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2? The subsets S_1 and S_2 must form a partition in the sense that they are disjoint and they cover S. The optimisation version asks for the "best" partition, and can be stated as: Find a partition into two subsets S_1, S_2 such that max (sum (S_1), sum (S_2)) is minimised (sometimes with the additional constraint that the sizes of the two sets in the partition must be

equal, or differ by at most 1).

The partition problem is equivalent to the following special case of the subset sum problem: given a set S of integers, is there a subset S_1 of S that sums to exactly t /2 where t is the sum of all elements of *S*? (The equivalence can be seen by defining S_2 to be the difference $S - S_1$.) Therefore, the pseudo-polynomial time dynamic programming solution to subset sum applies to the partition problem as well.

A variation of the partition problem is the 3-partition problem, in which the set S must be partitioned into |S|/3 triples each with the same sum. In contrast to partition, 3-partition has no pseudo-polynomial time algorithm unless P = NP: 3-partition remains NP-complete even when using unary coding.

■ **Example** Does SHORTEST-PATH finding of a graph is NP-complete.

■ **Answer:** No. SHORTEST-PATH is an optimization problem and not a decision problem; the definitions of NP-completeness apply only to decision problems, hence SHORTEST-PATH is not NP-complete.

■ **Example** Suppose there is a polynomial time algorithm FOO (G, u, v, k, proposed_path) which returns TRUE exactly whenever G is a graph, u and v are edges in that graph, k is an integer, and proposed_path is a path from u to v in G consisting of at most k edges. Does this mean PATH finding is NP-complete?

■ **Answer:** No. FOO is a verification algorithm for PATH that runs in polynomial time. For PATH to be NP-complete, we need to prove (i) that there is a polynomial time verification algorithm for PATH and (ii) every problem that is in NP can be reduced in polynomial time to PATH (i.e., PATH is NP-hard.) The existence of FOO proves (i), but not (ii), hence existence of FOO is not enough to prove that PATH is NP-complete.

■ **Example** Consider a reduction of problem A to problem B. What is the most precise claim you can make about problem B for each of the following situations?

(a) A is NP-complete and the reduction is in polynomial time.
NP-hard. (At least as hard as NP-complete problem.)

(b) A is in polynomial time and the reduction is also in polynomial time.
B could be anything.

(c) A is NP-complete and the reduction is in Pspace.
B could be anything.

(d) A is in nondeterministic polynomial time and the reduction is in polynomial time.
B is at least as hard as A, but nothing more can be said.

(e) A requires exponential time and the reduction is in polynomial time.
B must requires polynomial time for deciding.

(f) A is Pspace complete and the reduction is in Pspace.
B could be anything.

Suppose you could reduce an NP complete problem to a polynomial time problem in polynomial time. What would be the consequence?
What if the reduction required exponential time?
If we could reduce an NP-complete problem to a problem in P, then NP will be equal to P. If the reduction required exponential time, then there is not special consequence. (In fact any NP-complete problem can be reduced to a polynomial time solvable problem using exponential time reduction.)

■ **Question 1:** KNAPSACK (Karp's definition of Knapsack is closer to _____)
Dynamic programmingApproximation algorithmNP-completeSubset sum problem

■ **Question 2:** For example, KNAPSACK was shown to be NP-complete by reducing _____ to KNAPSACK.
Exact coverBinary relation3-dimensional matchingVertex cover

■ **Question 3:** CLIQUE (see also _____)
Independent set (graph theory)Graph coloringMatching (graph theory)Vertex cover

■ **Question 4:** One of the most important results in computational complexity theory was Stephen Cook's 1971 demonstration of the first (practically relevant) _____ problem, the Boolean satisfiability problem.
NP-hardNP -completeP versus NP problemNP (complexity)

2.5 Questions on Algorithms

1. What is the complexity of the following algorithm?

For i = 1, 2, ..., n-1 do: Compare the numbers in elements x_i and x_{i+1} and place the larger of the two numbers in x_{i+1} and the smaller in x_i. At the end, print x_n.

Answer: O(n). It needs n-1 comparisons. Thus, the order is linear order.

2. Consider the following algorithm for finding maximum and minimum of elements of an array x. How many comparisons are needed in total?

1. For *i* = 1, 2, ..., *n*/2, compare the two numbers in *xi* and *xi+(n/2)*, and place the smaller number in *xi* and the largest in *xi+(n/2)*.
2. Find largest of the *n*/2 numbers *x(n/2)+1, x(n/2)+2, …, xn*. This is the largest of the *n* given numbers.
3. Find smallest of the *n/2* numbers *x1, x2,…, xn/2*. This is the smallest of the *n* given numbers.

Answer: n/2 + (n/2-1) + (n/2-1) = 3n/2 - 2

3. A circus is designing an act consisting of a tower of people standing atop one another's shoulders. For practical and aesthetic reasons, each person must be both shorter and lighter than the person below her. Given the heights and weights of each person in the circus, what is the largest possible number of people in such a tower? Explain algorithmically and give complexity of the same.

Answer:

(a) Add a node to the graph for each performer.

(b) Examine each pair of nodes and add a directed edge from A to B if A can stand of B's shoulders.

(c) Add a "start" node with a directed edge to every other node in the graph.

(d) Find the longest path starting at the start node. (This can be accomplished by assign each edge a weight of -1 and finding the shortest path, since we are guaranteed not to get a negative cycle.)

(e) Find the farthest node and rebuild the path by following the appropriate edges backwards.

Complexity of the above algorithm is: O(V+E) where V is the number of nodes (here people) and edges of step (b).

4. Consider an array (a) with a series of positive elements and a series of zeros at the end. Propose an algorithm to find number of positive elements in the given array.

Answer: We find out an integer i such that $a[2^{i-1}]$ is positive and $a[2^i]$ is zero. This indicates that the number of positive elements of the array may be between $2^{i-1}+1$ (L+1) to 2^i -1 (U-1). Now, we can apply same logic to find out first zero valued element in between L+1 and U-1.

Element Values	Element indexes
9	1
10	2
22	3
312	4
22	5
90	6
50	7
988	8
88	9
89	10
22	11
0	12
0	13
0	14
0	15
0	16
0	17

2^{0th}= 1^{st} element is 9

2^1th= 2^{nd} element is 10

2^{2nd}=4^{th} element is 312

2^{3rd}=8^{th} element is 988

2^{4th}=16^{th} element is 0

Thus, L is 8 and U is 16. We can repeat the above steps on the array between 9 and 15. That is,

1^{st} element from 9^{th} element is 88

2^{nd} element from 9^{th} element is 22

4^{th} element from 9^{th} element is 0

Thus, now L and U becomes 11 and 13. When we repeat the same on 12 and 12, we can understand that 12^{th} element is the first zeroth valued element. Thus, array contains 11 elements.

Complexity of this problem can be said as log(n). as it is akin to binary searching.

5. How can we find out the existence of a value x in a sorted array whose size is unknown. Assume it contains a series of positive elements followed by zeros. Assume x value is other than zero.

Answer: First find out array size using the above technique explained in question 4. Then, apply binary searching. Complexity of this problem also can be said as O(log n).

6. In insertion sort, we scan back through the sorted portion of the list to determine where the new value should be inserted. We shift all the scanned values downward, and insert the new value in the open location. Explain how to use binary search to find the appropriate insertion spot rather than a linear scan. Explain why this does not help the overall time complexity of the algorithm.

Answer: On the i-th pass, the first i elements, A[1..i] of the array will be sorted. We can then use a binary search on these i elements to find out where the i+1 th element should be placed. While this does cut down the search part from worst case O(i), to worst case O(log i), the insertion part will still be worst case O(i), since we have to shift part or all of A[1,..,i] over by one to make room for A[i+1], so the binary search does not improve the time complexity of the algorithm.

7. Quicksort has worst-case time complexity $O(n^2)$ and average-case O(nlogn). Explain how to redesign the algorithm to guarantee a worst case O(nlogn).

Answer: We can guarantee O(n log n) is the worst case behavior by making the pivot be the median of the list, rather than the first element of the list. It takes O(k) time to find the median of a list of length k, and O(k) to partition it, so we can do both in O(k) time. By making the pivot be the median, each recursive call to quicksort will be on a list of length half as long, so the depth of the recursion tree will be at most lg n, with a recursion $T(n) = T(n/2) + \theta(n)$. The worst case running time for this algorithm is O(n lg n). Randomized quicksort is a few times faster than average, so for most applications this isn't a good algorithm.

8. Solve $T(n)=T(n-1) + \log(n)$.

Answer: By repeated substitution

$T(n) = \log n + \log(n - 1) + ... + \log(2) + \log(1)$

$= O(n \lg(n))$

9. It is proposed to use heap to find kth largest. Indicate the complexity of the approach you employ.

Answer: First, we build the heap with **O(n)** time. Deleting an item (root) from heap takes **O(lg n)** time. As we want kth largest, we can delet k items with **O(k lg n)** time. Thus, finding the k-th largest takes O(n + k lg n) time overall.

10. Finding kth largest with time complexity O(n+k log k).

Answer:

a. First, n elements will be kept in heap form with O(n).

b. We use an extra (second) heap to keep track of the possible candidates for next largest element. At the start the only thing on the new heap is the root of the first heap. At each stage the root of the new heap is the next largest element. We delete this root from the new heap and insert its children from the old heap to the new heap. This adds a net of one more item to the new heap. After finding the kth largest element, there will be k items on the new heap. The time complexity builiding the new heap is k lg k, since it must be built using a top-down method. So the total time complexity of finding the k-th largest will be O(n + k lg k).

11. What is the worst case behavior of the following code fragment that is used for finding greatest common divisor (GCD) of two integers?

```
int GCD(int a, int b){
   int Min = a<b?a:b;
   while (1){
   if( (a%Min==0) && (b%Min==0)) return Min;
   Min --;
}
}
```

Answer: In the worst case, the while loop will execute minimum of a and b times. Thus, worst case complexity can be said as: O(min(a,b))

12. The following two versions are proposed to find out GCD of two integers. Which is better? Mention about the worst case complexity of each.

```
int GCD1(int a, int b){
   if (a = = 0) return b;
   else if (b = = 0) return a;
   else if (a = = b) return a;
   else if (a<b)
            return GCD1(a, b-a);
   else
            return GCD1(a-b, b);
}
int GCD2(int a, int b){
   if (a = = 0) return b;
   else if (b = = 0) return a;
   else if (a = = b) return a;
   else if (a<b)
            return GCD2(a, b%a);
   else
            return GCD2(a%b, b);
}
```

Answer: First one time complexity is O(max(a,b)) while second one's time complexity is O(log(max(a,b)). Thus, second one is better.

13. What is the time complexity of the following?

$T(n) = 2T(n-1) + 1; T(1)=1$

Answer:

$T(n)=2T(n-1)+1$

$=2(2T(n-2)+1)+1$

$=4T(n-2)+2+1$

$=4(2T(n-3)+1) + 2+ 1$

$=8T(n-3)+ 4 + 2 +1$

$=8(2T(n-4)+1) + 4 + 2 + 1$

$=16T(n-4) + 8 + 4 + 2 + 1$

$=2^4T(n-4)+ 2^3+ 2^2+ 2^1 + 2^0$

Like this if we expand, it becomes

$=2^{n-1}T(1)+\ldots\ldots+2^4T(n-4)+ 2^3+ 2^2+ 2^1 + 2^0$

$=2^{n-1}+\ldots\ldots+2^4T(n-4)+ 2^3+ 2^2+ 2^1 + 2^0$

$=2^n-1= O(2^n-1)$

14. Let A[1::n] be a strictly sorted vector of integers. Find any k such that A[k] = k in O(log n) time. For example: see the following figure where 3 is having the required property.

1	2	3	4	5	6	7
–3	1	3	6	7	10	12

Answer: We can employ binary searching algorithm with little modification. We calculated mid as average of low and high indexes of the array then compare with the search element or key x. Now, we take search element value as mid itself. Whenever A[mid] is same as mid we return mid value. Thus, this approach search complexity can be said as: O(logn).

15. Let A[1::n][1::n] be a two-dimensional array of real numbers that is sorted both row-wise (A[x][y] < A[x][y + 1] for all x and y) and column-wise (A[x][y] < A[x + 1][y] for all x and y). How can you find a given number ***k*** in this array in O(n log n) time? How can you do it in O(n) time? For example (k = 5):

	1	2	3	4
1	0	1	2	6
2	2	4	5	7
3	3	6	7	8
4	6	7	8	9

Answer: For each row, check whether given k (search element) is between its first and last column element values or not. If k is in between then apply binary searching on that row and find k's location. Thus, computational complexity becomes O(nlogn).

16. Given a vector A[1::n] of integers find in O(n) time two indices ***i*** and ***j*** such that A[i] – A[j] is the greatest possible.

1	2	3	4	5	6	7	8	9	10	11
5	7	9	4	2	6	3	8	3	1	4
		i							*j*	

Answer: Find largest and smallest values along with their indices. Maximum A[i]-A[j] is same as the difference of largest and smallest values of the array. Thus, complexity of this becomes O(n) as calculation of largest and smallest is O(n).

procedure max_diff(a_1, a_2, …, a_n: integers)

min := a1

max := a1

for i := 2 to n

  if a_i < min **then** min := a_i

  else if a_i > max **then** max := a_i

m := max - min

Comparisons: 2n - 2

Time complexity is O(n).

Anoher solution:

procedure max_diff2(a_1, a_2, …, a_n: integers)

m := 0

for i := 1 to n-1

  for j := i + 1 to n

    if $|a_i - a_j|$ > m **then** m := $|a_i - a_j|$

{m is the maximum difference between any two numbers in the input sequence}

Comparisons: n-1 + n-2 + n-3 + … + 1

$= (n - 1)n/2 = 0.5n^2 - 0.5n$

Thus, time complexity is $O(n^2)$.

17. Given a vector A[1::n] of integers find in O(n) time two indices ***i*** and ***j*** such that A[i] – A[j] is the greatest possible and i > j.

5	7	9	4	2	6	3	8	3	1	4
				j			*i*			

Answer: Find largest second half of the array and smallest of the first half of the array along with their indices. Maximum A[i]-A[j] is same as the difference of largest and smallest values of the array. Thus, complexity of this becomes O(n) as calculation of largest and smallest is O(n).

18. Let A[1::n] be a vector of real numbers. Explain how to implement a function Sum(i; j) that computes the sum A[i] + A[i + 1] + ….. + A[j] in constant time.

Answer: Assume that you can do some preprocessing of A, i.e., before the first invocation of Sum, you can create an auxiliary vector B and use it inside Sum. Assume i<=j.

```
Int Sum(int A[], int i, int j, int n)
{
 int k;
int B[n]; // is the auxialary array
for(k=2, B[1]=A[1]; i<=n;k++)B[k]=A[k]+B[k-1];
// Now array B contains Cumulative sum
// This B can be assumed as a global array also
// We can also assume that precomputed B is sent as another argument to the function
return(B[j]-B[i]+A[i]); //This is constant time function.
}
```

19. Let A be an array of N integers not necessarily sorted. An element x of A is said to be a majority element if the number of elements of A that are equal to X is atleast (n+1)/2. Design and implement an algorithm that determines whether A has a majority element or not, and outputs such an element when A does have a majority element. The worst-case running time of your algorithm must be O(n).

Answer:

Assume max as A[1] and count as 1.

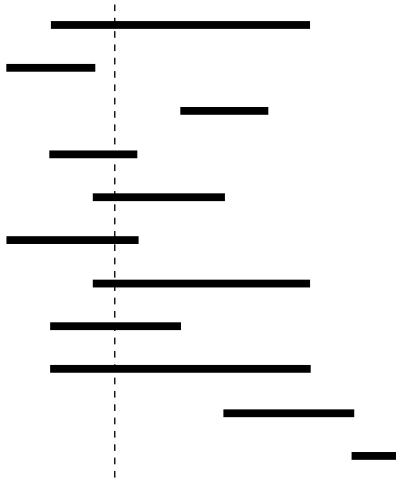
```
for(i=2;i<=n;i++)
{
 if(a[i]>max){ max=a[i]; count=1;}
else if(a[i]==max)count++;
}
if(count is>=(n+1)/2) print max as majority element
```

else print there is no majority element.

You may find the order of the above algorithm as O(n).

20. A department has n research students. An array A contains when a student starts and stops on a day. That is, the i-th student starts working at time A[i][1] and stops working at time A[i][2]. Propose algorithm to determine the greatest number of students that are working simultaneously. In the example below, the answer is 7. All values of the array refers to hours over 0 to 24 scale.

	1	2
1	2	8
2	1	3
3	5	7
4	2	4
5	3	6
6	1	4
7	3	8
8	2	5
9	2	8
10	6	9
11	9	10



Answer: Do find out smallest value (min) over starting hours (first column of A) and largest value (max) over ending hours (second column of A). Take another array B of size max elements and initialize its values to zeros. Now, take each student's starting (k) and ending (l) hours and increment values of elements of B between k to l by one. Now, maximum of the elements of array B gives the required answer.

21. Show that $f(x) = x^2 + 2x + 1$ is $O(x^2)$.

Answer:

For x > 1 we have:

$x^2 + 2x + 1 \le x^2 + 2x^2 + x^2$

$\Rightarrow x^2 + 2x + 1 \le 4x^2$

Therefore, for C = 4 and k = 1:

$f(x) \le Cx^2$ whenever x > k.

$\Rightarrow f(x)$ is $O(x^2)$.

22. Represent number of strings of length k bits that does not contain 11 in a recursive manner and also calculate number of such strings if k value is 10.

Answer: We can represent the number of strings of length k bits that does not contain pattern 11 as:

The number of strings of length k that do not contain 11	=	The number of strings of length k–1 that do not contain 11	+	The number of strings of length k–2 that don't contain 11

Or, in other words (as a recurrence relation):

(1) $s_0 = 1$ $s_1 = 2$

(2) $s_k = s_{k-1} + s_{k-2}$

If we calculate this recurrence for k value of 10, we get 144.

23. Let P(n) denote the following:

if $a_1, a_2, a_3, ..., a_n$ is the sequence defined by:

1. $a_0 = 1$
2. $a_k = a_{k-1} + 2$

Prove $a_n = 1 + 2n$ for all n >= 1

Answer :

Base Case: prove that P(1) is true.

Using the definition of the recurrence relation, we have: $a_1 = a_0 + 2 = 1 + 2 = 3$.

Now, we show that we get the same answer using the formula: $a_n = 1 + 2n$ for n = 1.

$a_n = 1 + 2n = 1 + 2 = 3$. Therefore, the base case holds.

Inductive Case:

The induction hypothesis is to assume P(k): $a_k = 1 + 2k$. We then need to show that P(k+1) = $a_{k+1} = 1 + 2(k + 1)$.

$a_{k+1} = a_{((k+1)-1)} + 2$ from the recurrence relation (this is a "given")

$a_{k+1} = a_k + 2$ algebra

$a_{k+1} = 1 + 2k + 2$ substitution from inductive hypothesis ($a_k = 1 + 2k$)

$a_{k+1} = 1 + 2(k + 1)$ algebra

Thus, we have shown the desired result. Since P(k+1) is true when P(k) is true, and we have shown P(1) holds, we have shown inductively that P(n) is true for all n >= 1.

24. Prove that $n^2=O(n^3)$

Answer:

To prove that $n^2= O(n^3)$, we need to find out positive c,n_0 such that $n^2 \le c^*n^3$, for all n >= n_0.

Dividing by n^2 yields $1 \le c^*n$.

$(1 \le c^*n)$ is true when c =1 and n> =1.

Hence, "$n^2 \le c^*n^3$, for all n >=n_0" is true given n_0=1 and c=1.

ie., we can find out positive c,n_0 such that $n^2 \le c^*n^3$, for all n >= n_0.

ie. $n^2= O(n^3)$ is true.

25. Prove that $2n^2 - 5n=\Theta(n^2)$

Answer:

To prove $2n^2–5n = \Theta(n^2)$, we need to find out positive c_1,c_2,n_0 such that $c_1{}^*n^2 \le 2n^2–5n \le c_2{}^*n^2$, for all n >= n_0.

Dividing by n^2 yields: $c_1 \le 2–5/n \le c_2$

Take $c_1 = 0.1$, c_2=100, $n_0 = 30$

For $2\text{-}5/n \le c_2$

Consider that 5/n is positive whenever n>=1.

$\Rightarrow 2–5/n \le 2$ for all $n \ge 1$

$\Rightarrow 2\text{-}5/n \le c_2$ for all $n \ge n_0$

$\Rightarrow 2n^2–5n \le c_2{}^*n^2$ for all $n \ge n_0$

For $c_1 \le 2–5/n$

Consider that (2–5/n) is an increasing function, and $(2–5/n_0) = 1.833$

$\Rightarrow 2 – 5/n \ge 1.833$ for all $n \ge n_0$

$\Rightarrow 2 – 5/n \ge c_1$ for all $n \ge n_0$

$\Rightarrow 2n^2 – 5n \ge c_1{}^*n^2$ for all $n \ge n_0$

$\Rightarrow c_1{}^*n^2 \le 2n^2 – 5n$ for all $n \ge n_0$

Conclusion, "$c_1{}^*n^2 \le 2n^2–5n \le c_2{}^*n^2$, for all $n \ge n_0$" is true given c_1= 0.1, c_2=100, n_0 = 30. (Indeed, it is also true for any c_1, c_2 and n_0 such that they are $\le 1/3$, ≥ 2, and ≥ 3 respectively.). ie., we can find out positive c_1,c_2,n_0 such that $c_1{}^*n^2 \le 2n^2–5n \le c_2{}^*n^2$, for all $n \ge n_0$. Therefore, $2n^2–5n = \Theta(n^2)$.

26. Given an array A of n integers: $a_1,a_2,..a_n$, the algorithm Longest(A) is designed to determine the length of the longest non-decreasing consecutive subsequence of integers starting with a1 (LN1 of A). For example, one such subsequence (LN1 of A) for A=<34,57,53,54,78>, is <34,57>. Hence Longest(A) should return 2. Prove the correctness of this algorithm by using a loop invariant. What are the worst and best cases of the algorithm? Describe the running time of the algorithm in terms of asymptotic tight, upper, and lower bounds.

```
int Longest(int A[])
{
int j=1;
while (j<n &&A[j+1]>=A[j])
j=j+1;
return j;
}
```

Answer:

Loop Invariant: At the start of each iteration of the while loop, a_1, a_2, a_3, .. a_j are non-decreasing consecutive subsequence of A starting with a_1.

Initialisation: Before the first iteration, j=1, then there is only one integer a_1, in a_1, a_2, a_3, .. a_j, which is non-decreasing consecutive subsequence.

Maintenance: In each iteration, the cycle is executed only if a_{j+1}>=a_j, then j is incremented by one. So a_{j-1}, a_j is non-decreasing. Hence, if the loop invariant is true before an iteration, it remains true before next iteration.

Termination: The while loop continues whenever a_{j+1}>=a_j and j<n. It ends with j=n, or a_{j+1}<a_j. In former case a_1, a_2, a_3, .. a_j, is the longest possible LN1 of A. In latter case, a_{j+1} is not involved in LN1 of A. Then a_1, a_2, a_3, .. a_j is LN1 of A.

Best case: LN1 of A only has one element. $\theta(1)$

Worse case: LN1 of A only has n element. $\theta(n)$

Algorithm: $O(n)$, $\Omega(1)$

27. Propose a recursive numbers procedure to calculate sum of squares of natural between m and n.

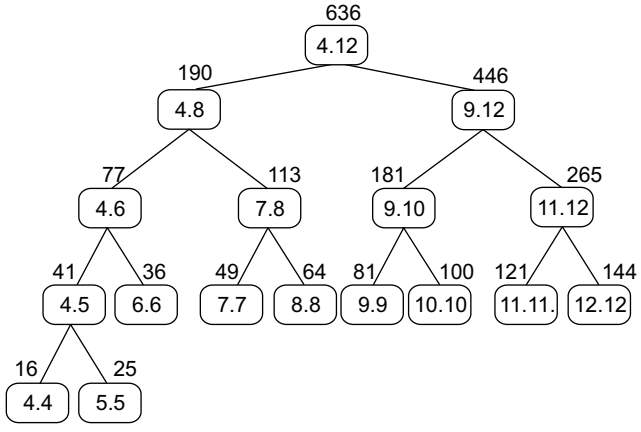
Answer: The following is a divide-and-conquer method for computing the sum of squares of all integers between *m* and *n*, i.e.,

$$\sum_{i=m}^{n} i^2$$

```
int ssq(int m, int n) {
if(m > n) return 1; // error condition
 else if (m == n) return m*m;
 else {
  int middle = (m+n)/2;
  return ssq(m,middle) + ssq(middle + 1,n);
  }
}
```

The following diagram explains the call trace.

- The two numbers inside the nodes are *m* and *n*.
- The number above a node is its return value.



28. An integer number x with initial value 1 is doubled repeatedly till x >=N, where N is another integer number. How many times should it be doubled before x ≥ N? Do comment on the time complexity of this operation.

Answer: After k doubling we have 2^k. So we find the smallest k such that $2^k \geq N$.

Therefore, $k \geq \log_2 N$

It is logarithmic complexity – O(log N)

29. An integer number x with initial value N is halved repeatedly till x ≤1, where N is another integer number. How many times should it be halved before x ≤ 1?

Answer: After k halving we have $N^*(½)^k$. So we find the smallest k such that $N^*(½)^k \geq 1$.

Therefore $k \geq \log_{.5} N$

It is logarithmic complexity – O(log N)

30. Here are two methods for finding the maximum value in an array of integers. Each is correct. Which one is faster? By how much? Explain.

```
int max1(int[] nums, int last) {
 if (last == 0) return nums[0];
 else {
  if (nums[last] > max1(nums,
last - 1)) {
  return nums[last];
 } else {
  return max1(nums, last - 1);
    }
    }
}
```

```
int max2(int[] nums, int last) {
 if (last == 0) return nums[0];
 else {
int temp = max2(nums, last - 1);
 if (nums[last] > temp) {
  return nums[last];
 } else {
    return temp;
    }
    }
}
```

Answer: The second one is much faster. The first takes exponential time, $O(^{nums.length})$, because every call may result in two recursive calls. The second makes one call for each element of the array, so is only linear time, O(nums.length).

31. Does the following code counts the number of digits in a positive integer variable number:

```
int n = number;
int count = 0;
while (n > 0) {
n = n / 10;
count++;
}
```

How long (in Big-O terms) does this method take? Why?

Answer: Yes. It complexity is O(log n) time, because the number of digits in a number is the log10 of the number.

32. What is golden ratio?

Golden ratio Φ is defined as

$$1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+...}}}$$

Approximated value for the above =(1+sqrt(5))/2

Golden ratio Φ_n after truncating n terms can be given as f_n/f_{n-1}, where f_n, f_{n-1} are the fibnocci numbers. Assuming Φ value as 1.61803398874989, we can calculate fibnocci series using the following relationship also.

$$fn = \frac{1}{2\phi - 1}\left(\phi^{n+1} - (1-\phi)^{n+1}\right)$$

The following is called as fibnocci power series, where f_n is the fibnocci number.

$$F(x) = \sum_{n=1}^{\infty} f_n x^n$$

$$= x + 2x^2 + 3x^3 + 5x^4 + 8x^4 + 13x^6 + ...$$

An efficient recursive approach for computing fibnocci power series terms is

$$pk = fk\ xk$$

Then, since

$$fk_{+1}\ xk^{+1} = fk\ xk + fk_{-1}\ xk^{-1}$$

the terms pk satisfy

$$pk_{+1} = pkx + px_{-1}x^2$$
$$= x\ (px + xpk_{-1})$$

We can prove for infinite fibnocci power series F(x) can be given as:

$$F(x) = \frac{x + x^2}{1 - x - x^2}$$

An application:

A businessman opened a retirement account by depositing a certain amount of money (some dollars, no cents) just before the end of a year. He made another deposit (again, some dollars, no cents) one year later. From that time, towards the end of each year he deposited some amount of money. His deposits followed a pattern: From the third year onwards, the amount of each deposit was equal to the total of two previous deposits (from two previous years). His 20th deposit was $1,000,000 exactly. What was his initial deposit?

Answer: If you observe, it follows Fibnocci series.

- 1st year: x
- 2nd year: y
- 3rd year: $x + y$
- 4th year: $x + 2y$
- 5th year: $2x + 3y$
- 6th year: $3x + 5y$
- 7th year: $5x + 8y$
- 8th year: $8x + 13y$
- 9th year: $13x + 21y$
- 10th year: $21x + 34y$
- 11th year: $34x + 55y$
- 12th year: $55x + 89y$
- 13th year: $89x + 144y$
- 14th year: $144x + 233y$
- 15th year: $233x + 377y$
- 16th year: $377x + 610y$
- 17th year: $610x + 987y$
- 18th year: $987x + 1{,}597y$
- 19th year: $1{,}597x + 2{,}584y$
- 20th year: $2{,}584x + 4{,}181y$

That is, $2584x + 4181y = 1{,}000{,}000$

Using golden ratio, 19th year total can be found as: 1000000/1.618034 = 618034

That is,

$2{,}584x + 4{,}181y = 1{,}000{,}000$

$1{,}597x + 2{,}584y = 618{,}034$

Thus, initial year amounts are:

$x = 154$

$y = 144$

33. What is the complexity of the following algorithm to find product of two n-bit integer numbers?

```
int prod(int x, int y){
int z=0;
while(y){
if(y%2)z+=x;
x=x+x;
y/=2;
}
return z;
}
```

Answer: While loop runs for n times at most. Thus, complexity can be said as O(n).

34. What is the complexity of the following algorithm to find product of two n-bit integer numbers?. First, represent the time complexity and then find the same in big-oh notation.

```
int rprod(int x, int y){
if(y==0)return 0;
else if(y%2) return (x+rprod(2*x,y/2));
else return rprod(2*x,y/2);
}
```

Answer: If y value is 0, it ruturns 0. For this situation we need a constant time c. Similarly, Time complexity relation can be represented as:

T(n)=T(n-1)+d for n>1

= c otherwise.

Here, d is the operations needed if y value is not 0.

If we solve the above equation recursively, we get

T(n)=T(n-1)+d

=T(n-2)+d+d

=T(n-3)+d+d+d

=T(1)+d+d+…n times

=c+nd=O(n)

35. General theorem for solving recurrence relationship of time complexity equations. If n is power of c and the recurrence relation is of the form:

T(n)=d if n<=1

=aT(n/c)+bn otherwise

Then, solution for the recurrence relation is:

$T(n)=O(n)$ if $a<c$

$=O(n\log n)$ if $a=c$

$=O(n^{\log_c a})$ if $a>c$

36. Now consider again the problem of multiplying two n-bit numbers. Let us assume n is an integer power of 2. We can represent n-bit number x and y as two n/2 bit numbers as shown below:

$x=a2^{n/2}+b$, where is a and b are most significant and least significant n/2 bits of x.

$y=c2^{n/2}+d$, where is c and d are most significant and least significant n/2 bits of y.

Therefore,

$xy=(a2^{n/2}+b)*(c2^{n/2}+d)=ac2^{n}+(ad+bc)2^{n/2}+bd$

Answer:

If we observe the above equation, we may find that we need three n/2 bit number multiplications and some shifting operations. Thus, time complexity of this approach can be represented as:

$T(n)=c$ if $n=1$

$=3T(n/2)+dn$ otherwise

Here, c, d are some constants.

Time complexity in big-oh notation can be represented as:$O(n^{\log 3})=O(n^{1.59})$

37. Strassen matrix multiplication time complexity is given below. Arrive at its time complexity in big-oh notation.

$$T(n) = \begin{cases} c & \text{if } n=1 \\ 7T(n/2)+dn^3 & \text{otherwise} \end{cases}$$

where c, d are constants.

$$\begin{aligned} T(n) &= 7T(n/2)+dn^2 \\ &= 7(7T(n/4)+d(n/2)^2)+dn^2 \\ &= 7^2T(n/4)+7dn^2/4+dn^2 \\ &= 7^2T(n/8)+7^2dn^2/4^2+7dn^2/4+dn^2 \\ &= 7iT(n/2i)+dn^2\sum_{j=0}^{i-1}(7/4)^j \\ &= 7^{\log n}T(1)+dn^2\sum_{j=0}^{\log n-1}(7/4)^j \\ &= cn^{\log 7}+dn^2\frac{(7/4)^{\log n}-1}{7/4-1} \\ &= cn^{\log 7}+\frac{4}{3}dn^2\left(\frac{n^{\log 7}}{n^2}-1\right) \\ &= O(n^{\log 7}) \\ &\approx O(n^{2.8}) \end{aligned}$$

38. Represent time complexity of binary search algorithm and arrive at its big-oh equivalent.

Time complexity of binary search can be given as:

$T(n)=0$ if $n=1$

$=T(n/2)+1$ Otherwise

$T(n)=T(n/2)+1$

$=(T(n/4)+1)+1$

$=((T(n/8)+1)+1)=1$

$=T(1)+1+1+1+\ldots+1$

$=O(\log n)$

39. Represent the time complexity of towers of honoi problem and arrive at its big-oh notation.

The T(n) number of moves needed to move n disks from peg j to peg k as:

$T(n)=1$ if $n=1$

$=2T(n-1)+1$ Otherwise

To arrive at its big-oh equivalent, we solve the recurrence relationship as shown below.

$T(n)=2T(n-1)+1$

$=2(2T(n-2)+1)+1$

$=4T(n-2)+2+1$

$=4(2T(n-3)+1)+2+1$

$=8T(n-3)+4+2+1$

……

……

$=2^{n-1}T(1)+\ldots+4+2+1$

$=2^{n}-1=O(2^{n})$

40. When an adjacency list is preferred over adjacency matrix?

41. What will be the sum of lengths adjacency list of a directed graph with E edges?. What happens if graph is undirected?

42. If A is adjacency matrix and $A=A^T$ then what can you comment about graph?

43. To find whether an edge is present or not which representation adjacency list or matrix is preferred?

44. Given an adjacency-list of a directed graph how long it takes to compute out-degree of every vertex ? How long does it takes to compute the in-degree?

45. What will be the size of every nodes adjacency list size if a complete binary tree is represented in adjacency list form?. What happens if we assume if every node eve contains father field ?

46. If G =(V,E) is a graph then find out a procedure to find out adjacency matrix of a graph G^T which is created by reversing all the edges of G, given adjacency matrix of G. Analyse its time complexity.

47. Repeat 46 assuming adjacency list of G is given.

48. If G is a graph and A is its adjacency matrix then explain what A^2 will explain?. Also describe efficient procedures to calculate A^2.

49. Show that finding whether a graph contains a sink (a vertex with in-degree of V-1 and out-degree as zero) is O(V), where V is number of vertexes. Prove this both for adjacency matrix and list representations of graph.

50. What is incidence matrix? If B is an incidence matrix what BB^T represents?

51. Average running time of an algorithm is given as: T(n)=8T(n/2) + qn if n >1

= p if n =1

Where p, q are constants then find out the order of the algorithm.

52. Average running time of an algorithm is given as: T(n)=T(n-1) + q if n >1

= p if n <=1

Where p, q are constants then find out the order of the algorithm.

53. Average running time of an algorithm is given as: T(n)=T(n-1) + 1/n if n >1

= 1 otherwise

then, find out the order of the algorithm.

54. Running time of an algorithm is given as:

T(n)=T(n-1)+T(n-2)+T(n-3) if n>3

= n otherwise

then, find out the order of the algorithm.

55. Describe a simple strategy for removing an element from a max-heap.

Swap with the last element in the heap, recursively swap that element with the max of its children until it is in a position where both children are smaller.

56. What is the running time for a method that returns the size of a heap and why?

O(1) because we need to know where to insert the next value anyway, and heaps are implemented in an array. The index of that spot is the size.

57. Describe in detail why recursion is so important in data structures and algorithms.

Answer:

(i) Recursion is a very succinct (shorthand) way of defining an entity or writing programming language code

(ii) Non-recursive code solutions tend in certain cases to be longer and more complicated than recursive solutions

(iii) Recursion changes your mindset i.e. your way of thinking about programming constructs and writing code

(iv) Recursion is also useful in defining data structures (eg a sequence or a tree)

58. How do you calculate "big-O" for an if-then-else statement?

Answer:

If E then S1 else S2 – cost = cost (E) + max(cost(S1), cost(S2))

59. How to print permutations of a string?

Answer:

```
#include < stdio.h >
#define SIZE 3
int main(char *argv[],int argc)
{
char list[3]={'a','b','c'};
int i,j,k;
for(i=0;i < SIZE;i++)
for(j=0;j < SIZE;j++)
for(k=0;k < SIZE;k++)
if(i!=j && j!=k && i!=k)
printf("%c%c%c\n",list[i],list[j],list[k]);
return(0);
}
Recursive permutations
#include < stdio.h >
#define N 5
int main(char *argv[],int argc)
{
char list[5]={'a','b','c','d','e'};
permute(list,0,N);
return(0);
}
void permute(char list[],int k, int m)
{
int i;
char temp;
if(k==m)
{
/* PRINT A FROM k to m! */
for(i=0;i < N;i++){printf("%c",list[i]);}
printf("\n");
}
else
{
```

```
for(i=k;i < m;i++)
{
/* swap(a[i],a[m-1]); */
temp=list[i];
list[i]=list[m-1];
list[m-1]=temp;
permute(list,k,m-1);
/* swap(a[m-1],a[i]); */
temp=list[m-1];
list[m-1]=list[i];
list[i]=temp;
}
}
}
```

If we observe, the recursive version has better scalability.

60. What is the complexity of the following function of calculating an integer power of X?

```
float Power(float X, int N){
if (N == 1) return X ;
else if(N%2==0) return Power(X,N/2)*Power(X,N/2);
else return Power(X,N/2)*Power(X,N/2+1);
}
```

Answer: If we observe the above code fragment, we can represent its time complexity as:

T(n)=2T(n/2)+1
=2(2T(n/4)+1)+1
=4T(n/4)+2+1
=4(2T(n/8)+1)+2+1
=8T(n/8)+4+2+1

Thus, complexity becomes O(n)

61. Identify the complexity of the following algorithm to find out F from an array of values a.

$$F = \sum_{j=1}^{n} \frac{\sum_{i=1}^{j} a_i^a - \sum_{i=1}^{j} i a_i}{j(j+1)}$$

Solution :

Algorithm F (n : integer)
sum ← 0, perfix-sum ← 0
for $j \leftarrow 1$, to n do
perfix-sum ← perfix-sum + $(a_j^2 - j \,.\, a_j)$
sum ← sum + perfix-sum/ $(j \,.\, (j+1))$
return sum

Answer: If we observe the algorithm, we find that it runs for n times. Thus, itc complexity can be said as: O(n).

62. Suppose we are given an acyclic directed graph G with vertices v1, v2, ..., vn. All edges are forward, that is, if (vi, vj) is an edge of G then i < j. Give an efficient algorithm that will compute the number of paths from v1 to vn in G. What is the running time?

Answer:

Algorithm

```
{Input: DAG G = (V,E); V = {v1, ..., vn}; all
edges are of the form (vi, vj) for i < j}
{Output: the number of paths from v1 to vn}
numOfPaths[1]=1
for( j=2;j<n;j++){
numOfPaths[j] =0
for all edges (vi, vj) ∈ E do
numOfPaths[j] =numOfPaths[j] + numOfPaths[i]
end for
}
return numOfPaths[n]
```

Running Time: Let m = |E|. Assume G is represented by adjacency list. The external loop has O(n) iterations, while the internal loop considers each edge only once, so the running time is O(m + n).

63. Assume that we have 81 coins and a balance. Only one coins weighs less than the others. Give an algorithm (in pseudocode) to identify the light coin which uses the balance only 4 times in the worst case.

Answer:

for k = 1 to 4
put $81/3^k$ (=27 if k=1) coins on one pan and $81/3^k$ (=27 if k=1) coins on the other pan
if the two pans weigh the same then throw away the coins in both the pans of the balance else if the left pan weighs less than the right pan then throw away the coins not in the balance and on the right pan else throw away the coins not in the balance and on the left pan

64. If a problem P has best-case time complexity $\Omega(n \log n)$ and worst-case time complexity $O(n^2)$ and algorithm A solves problem P, which of the following is possible?

(a) *A* has best-case time complexity $\Theta(n)$.
(b) A has worst-case time complexity $\Theta(n\sqrt{n})$.
(c) A has average-case time complexity $\Theta(n^3)$.
(d) *A* has worst-case time complexity $\Theta(n)$.

Answer: (a) possible, (b) possible, (c) possible, (d) impossible (violation of worst-case time complexity $\Omega(n \log n)$.

65. Suppose that the votes of n people for different candidates (where there can be more than two candidates) for a particular office are the elements of a sequence. If we assume two candidates in the election whose labels are A and B. Then, we assume there is only one candidate in the voting then possible sequences are A or B. If we assume there exists two candidates who voted then possible sequences will be AA, AB, BA and BB. For the sake of simplicity, we assume number of voters are many. A person wins the election if this person receives a majority (more than half) of the votes. Describe a divide-and-conquer algorithm that determines whether a candidate received a majority and, if so, determine who this candidate is. By applying the Master Theorem to estimate asymptotic complexity of the algorithm you have employed.

Answer: If the sequence has just one element, then the one person on the list is the winner. Otherwise divide the list into two parts, the first half and the second half, as equally as possible. (No one could have gotten a majority of the votes on this list without having a majority in one half or the other, since if a candidate got less than or equal to half the votes in each half, then he got less than or equal to half the votes in all.) Thus apply the algorithm recursively to each half to come up with at most two names. Then run through the entire list to count the number of occurrences of each of these names to decide which, if either, is the winner. This requires at most 2n additional comparisons for a list of length n, thus for the total number of comparisons we get the following recursion:

$T(n) = 2T(n/2) + 2n$; $T(1) = 1$:

Here, a=2, b=2 and d=1. Thus, it is the second case of Masters theorem. Thus, complexity of this algorithm will be O(nlogn).

66. Assume that our sample space is the set of permutations of the first n positive integers (1..n), and assume that each permutation is equally likely (we have a uniform random permutation). What is the probability that a random permutation has (a) $n(n - 1)/2$ inversions? (b) 0 inversions? (c) exactly 1 inversion?

Answer:

(a) The random permutation has $n(n–1)/2$ inversions exactly when it is sorted in decreasing order. Since exactly one of the $n!$ permutations is sorted in decreasing order, its probability is $1/n!$.

(b) The random permutation has 0 inversions exactly when it is sorted in increasing order, its probability is again $1/n!$.

(c) The random permutation has exactly 1 inversion in the following $(n – 1)$ permutations: (2, 1, 3, 4,….. n), (1,3, 2, 4,…. n),..., (1, 2, 3, n–1, n). The probability of drawing one of these permutations is $(n–1)/n!$.

67. Describe an $O(n \log k)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists. Use a min-heap.

Answer: First we remove the smallest element from each sorted list and we build a min-priority queue (using a min-heap) out of these elements in $O(k)$ time. **Then we repeat the following steps:** we extract the minimum from the min-priority queue (in $O(\log k)$ time) and this will be the next element in the sorted order.

(a) From the original sorted list where this element came from we remove the next smallest element (if it exists) and insert it to the minimum-priority queue (in $O(\log k)$ time).

(b) We are done when the queue becomes empty and at this point we have all the numbers in our sorted list.

(c) The total running time is $O(k + n \log k) = O(n \log k)$.

68. Show that the second largest element can be found with n+log n–2 comparisons in the worst case.

Answer: First we get maximum of two adjacent numbers for each of the pairs of the n numbers. If n is odd then last number will be used in the next level. In the next level we find maximum of the maximum of previous level. This we repeat till we get largest. Thus, We have total logn levels for finding maximum. This needs n-1 comparisons. Now, we take those numbers which are involved in comparing with maximum and we find maximum of them which becomes the second maximum. As we have logn levels, we will be having logn such numbers. To find maximum of them we need logn-1 comparisions. Thus, in total we will be using n+logn-2 comparisions. Thus, the complexity of finding second maximum with the approach is O(n+logn). See the following numerical example.

1	2	5	4	9	7	8	7	5	4	1	0	1	4	2	3
2		5		9		8		5		1		4		3	
5				9				5				4			
9								5							
9															

At each step we are comparing adjacent numbers and taking the larger of the two. It takes n-1 comparisons to get down to the largest number, 9. Then, if we look at every number that 9 was compared against (5, 5, 8, 7), we see that the largest one is 8, which is the second largest in the array.

69. Describe a $O(n)$ worst-case time algorithm that, given a set S of n distinct numbers and a positive integer $k <= n$, determines the k numbers in S that are closest to the median of S in the sorted order of S (for simplicity we assume that both n and k are odd, so there is one median).

Answer: Let $m=(n + 1)/2$ (the rank of the median). Using the worst-case linear time selection algorithm twice, we find the two elements $x1$, $x2$ with ranks $m-(k–1)/2$ and $m+(k–1)/2$. Then we go through S and find all elements x for which $x1 <= x <= x2$, these are the solutions. The total worst-case running time is O(n).

70. We have two input arrays, an array A with m elements and an array B with n elements, where $m <= n$. There may be duplicate elements. We want to decide if every element of B is an element of A. Describe an algorithm to solve this problem in $O(n \log m)$ worst-case time.

Answer: Let us first sort A by using MERGESORT (in $O(m \log m)$ time). Then for each element of B we do a binary search in the sorted list of A (in $O(n \log m)$ time). The total worst-case running time is $O((m + n) \log m) = O(n \log m)$.

71. We have a set A having n elements where n>=2. Describe a $O(n)$ worst-case time algorithm to find two elements

$x, y \epsilon A$ such that $|x – y| >= |u – v|$ for all $u, v \epsilon A$.

Answer: Find the minimum and the maximum simultaneously which we can do in $3n/2$ *comparisions. This,* $O(n)$ time.

72. We have a set A having n elements where n>=2. Describe a $O(n)$ worst-case time algorithm to find two elements

$x, y \epsilon A$ such that $|x – y| <= |u – v|$ for all $u, v \epsilon A$.

Answer: For this we sort the numbers first, then x and y must be consecutive elements in the sorted order. We go through the sorted list and we find the smallest difference between two neighbouring elements.

73. An array A containing n elements is said to be *up-to-k-sorted (where (n-1)>=k>=0)* if every element of A is at most k positions from its final position when A is sorted. For instance if A is *up-to-0-sorted*, then it is sorted, and (2, 1, 4, 3, 5, 9, 6, 7, 8) is *up-to-3-sorted* but it is not *up-to-1 sorted*, since element 9 is three positions from its final position when A is sorted.

(a) What is the average-case running time of MERGE-SORT on an array which is *up-to-48-sorted*?

(b) What is the worst-case running time of INSERTION-SORT on an array which is *up-to-48-sorted*?

Answer: (a) MERGE-SORT's running time is always $O(n \log n)$. Thus, in this case also complexity is same.

(b) INSERTION-SORT uses time $O(n)$ to iterate through A, and then time $O(m)$ to swap elements, where m is the number of inversions. Since each element of A is involved in at most 48 inversions, $m <= 42n$, and the algorithm's worst-case running time is $O(n)$.

74. Show that in a set of n distinct elements the 13-th largest element can be found with at most n + 12 ceil (log n) comparisons in the worst case.

Answer: We propose to use the tournament method and sort n elements (using n – 1comparisons). We remove the largest element and we replace it with $–\infty$ in the tournament. We find the winner again using at most *ceil(*log n*)* comparisons (only one path has to be recomputed). We remove this element again and replace it with $–\infty$, etc. we repeat this 12 times. The total number of comparisons is at most n + 12*ceil(*log n*)*

75. How can we limit the size of the queue array?

Answer: One way to limit the size of the queue array is to make the head and tail pointers wrap to the beginning of the array whenever they reach the end of the array. This causes the queue to "recycle" the same array memory cells.

76. Why do we need to limit the size of the queue array?

Answer: If we do not limit the size of the queue array, then the queue will eventually traverse the entire memory of the computer as items are added and removed from the queue.

77. Compare the stack with queue.

Answer: A stack is similar to a list in that the arrangement of the data items is linear. The stack resembles a list that is restricted. With a typical list, we can insert and remove items at various places in the list. With a stack, however, we can only insert and remove items from the top of the stack.

78. What is the difference between peek and pop operations on the stack?

Answer: Pop removes and returns the item on the top of the stack. While peek returns only, will not remove the top item.

79. Let us assume a stack contains items AABC with TOS pointing to C. When we apply peek, peek, pop, peek, peek then what is the output string?

Answer: CCCBB

80. Why we don't worry about stackfull() in linked list based stack realisations?

Answer: It can grow to very large extent. That is, till system's virtual memory is exhausted.

81. In your browser, we have freedom to move to previous page and next page using left and right arrow icons in your browser menu bar. Which data structure is used there?

Answer: Stack.

82. What are the advantages of linked list based stack over array based stack?

Answer: One advantage of using a linked list to implement a stack is that a linked list does not have a fixed size. This means that our linked list can grow to the size of the stack. We do not need to worry about setting aside a certain block of memory for the stack. One disadvantage of using a linked list is the increased complexity of using pointers. With pointers it is easier to make mistakes while implementing the logic of the stack operations.

83. How many pointers are found in any node of a binary tree?

Answer: Usually two. However, in some realisations which refers to parents nodes also. Thus, there 3 pointers are found in a node.

84. When we have traveled a max heap in in-ordered fashion, we have got elements in ascending order? What is the comment on the data in the tree.

Answer: Already, elements are in ascending order.

85. When we traverse a max heap, do we always get ascending ordered data?

Answer: No.

86. A finite set of elements that is either empty or is partitioned into three disjoined subsets is called _____.

Answer: Tree

87. What is the complexity of Dijkstras algorithm?

Answer: O((E+V)logV)

88. A graph contains ith row and column is zero's. Then number of isolated points are ____.

Answer: One

89. In which graph traversal method, all the successors of a visited node will be visited before visiting their successors (successor's of successor's)?

Answer: Breadth First Traversal

90. From the root node of a tree, if we execute the following while loop, which node tree will be pointing?

while(tree&&tree->left) tree=tree->left.

Answer: Left most node.

91. What are minimum number of queues needed to implement the priority queue?

Answer: Two. One queue is used for actual storing of data and another for storing priorities.

92. What is the data structure used to perform recursion?

Answer: Stack. Because of its LIFO (Last In First Out) property it remembers its 'caller' so knows whom to return when the function has to return. Recursion makes use of system stack for storing the return addresses of the function calls.

Every recursive function has its equivalent iterative (non-recursive) function. Even when such equivalent iterative procedures are written, explicit stack is to be used.

93. What are the notations used in Evaluation of Arithmetic Expressions using prefix and postfix forms?

Answer: Polish and Reverse Polish notations.

94. We have n nodes where n is odd number greater than 0. What is the worst possible height of the tree with this n nodes?

Answer: n.

95. We have n nodes where n is odd number greater than 0. What is the worst possible height of the tree with this n nodes if they are organised as strictly binary tree?

Answer: n/2+1

96. We have n nodes where n is odd number greater than 0. What is ther worst possible number of leaf nodes if n nodes are organised as strictly binary tree?

Answer: n/2+1.

97. We have n nodes where n is odd number greater than 0. What is the worst possible number of non-leaf nodes if n nodes are organised as strictly binary tree?

Answer: n/2.

98. We have n nodes which are organised as binary tree. What is the worst possible number of leaf nodes if n nodes are organised as deepest binary tree?

Answer: 1.

99. We have n nodes which are organised as binary tree. What is the worst possible number of non-leaf nodes if n nodes are organised as deepest binary tree?.

Answer: n–1.

100. We have n (where $n=2^k-1$, for integer value of k) nodes which are organised as binary tree. What is the

possible number of leaf nodes if n nodes are organised as least height binary tree?.

Answer: 2^{k-1}.

101. We have n (where $n=2^k-1$, for integer value of k) nodes which are organised as binary tree. What is the possible number of non-leaf nodes if n nodes are organised as least height binary tree?.

Answer: $2^{k-1}-1$.

102. Which tree all the nodes will be having their balance values as 0?.

Answer: Complete binary tree.

103. We have n (where $n = 2^k - 1$, for integer value of k) nodes which are organised as binary tree. What is the possible balance values of nodes if n nodes are organised as least height binary tree?

Answer: 0.

104. A queue is maintained in an array, and F and R are the front location and rear location of the queue respectively. Obtain Formula for N, the number of elements in the queue in terms of F and R.

Answer: Number of Elements = R – F + 1

105. Explain the difference between general queue and circular queue.

Answer: There is no "first" or "last" concept in circular queues in general. However, we can maintain them if needed externally. Main advantage of circular queue is addition and removal can be carried out either at front or rear easily.

106. The following code fragment is proposed to a new node into a binary search tree. Will it work?

```
   struct node {
   int info;
   struct node *left;
 struct node *right;
                  };
struct node *Insert(struct node *newNode, struct
node *Root){
 if(Root == NULL) return newNode;
 if(newNode->info > Root->info)
 Root->right = Insert(newNode, Root->right);
 else
 Root->left = Insert(newNode, Root->left);
 return (Root);
}
```

Answer: Yes.

107. Assume that we have inserted numbers 1,2,7,9, and 10 into a empty binary search tree. How many times, we have to apply rotations to get the balanced tree. What type of rotations are needed?

Answer: Two rotations of left type.

108. Five descending ordered elements are inserted into a binary search tree?. How many rotations and what type of rotations are needed to get AVL tree?

Answer: Two rotations of right type.

109. We have inserted 7 elements into an empty binary search tree. It is observed that all the rotations which took place during the AVL tree formation are right rotations. What is the nature of data entered?

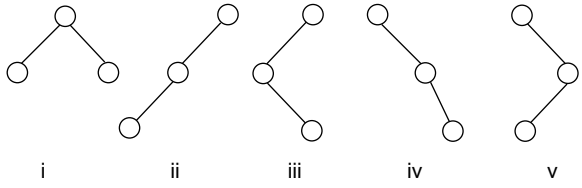
Answer: All the seven elements are ascending ordered elements.

110. How many different trees are possible with n nodes?

Answer: $2^n - n$

111. What are the different trees with three nodes?

Answer:



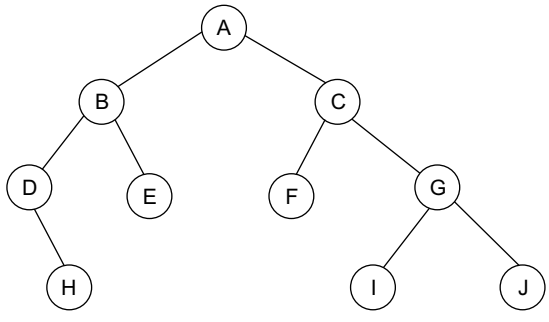
112. In order and pre-order traversals of a tree is given as:

Inorder : D H B E A F C I G J

Preorder: A B D H E C F G I J

Find out the topology of the tree.

Answer:



113. Let G be a graph whose vertices are the integers 1 through 8, and let the adjacent vertices of each vertex be given by the table below:

Vertex	Adjacent Vertices
1	2, 3, 4
2	1, 3, 4
3	1, 2, 4
4	1, 2, 3, 6
5	6, 7, 8
6	4, 5, 7
7	5, 6, 8
8	5, 7

Assume that, in a traversal of G, the adjacent vertices

of a given vertex are returned in the same order as they are listed in the above table.

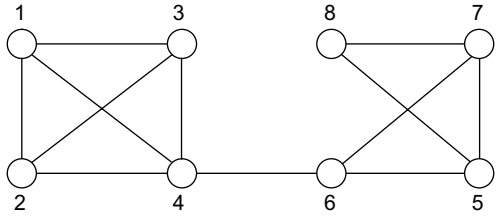
(a) Draw G.

(b) Order the vertices as they are visited in a DFS traversal starting at vertex 1.

(c) Order the vertices as they are visited in a BFS traversal starting at vertex 1.

Answer:

(a)



(b) 1-2-3-4-6-5-7-8

(c) 1-2-3-4-6-5-7-8

114. What is the difference between a **path** and a **cycle**?

Answer: A path is a sequence of alternating vertices and edges that begins with a source vertex and ends with a destination vertex. Each edge is preceded and followed by its endpoints. A cycle is circular sequence of alternating vertices and edges that starts and ends at the same vertex. Each edge is preceded and followed by its endpoints.

115. Suppose a graph with 5 vertices with no self-loops.

(a) What is the maximum number of edges if graph is undirected?

(b) What is the maximum number of edges if graph is directed?

(c) Are the results of (a) and (b) different? Why or why not?

Answer:

(a) (5*4) / 2 = 10

(b) 5*4 = 20

(c) They are different because two adjacent vertices can have at most 1 undirected edge while they can have 2 directed edges.

116. Suppose *last* points to the last node in a singly linked list *list1* whose front is *head1*. Suppose *head2* points to the first node in another singly linked list *list2*. Which statement will append *list2* to the end of *list1* (ie. make both lists into one linked list with *list1*'s elements before *list2*)?

A. last = *head2*.next;

B. head2 = *last*;

C. last = *head2*;

D. head2.next = *last*;

E. last.next = *head2*;

Answer: E

A, B, C do not modify the lists in any way, they only modify the *last* and *head2* pointers. B is fatal and will lose *list2*. C loses track of the end of *list1*. D kills *list2* by dropping all nodes after the first node and pointing the first node to the last node of *list1*.

117. Which of the following shows the correct relationship among some of the more common computing times for algorithms? ($\lg n = \log_2 n$)

A. $O(\lg n) < O(n) < O(n \lg n) < O(2^n) < O(n^2)$

B. $O(n) < O(\lg n) < O(n \lg n) < O(2^n) < O(n^2)$

C. $O(n) < O(\lg n) < O(n \lg n) < O(n^2) < O(2^n)$

D. $O(\lg n) < O(n) < O(n \lg n) < O(n^2) < O(2^n)$

E. $O(\lg n) < O(n \lg n) < O(n) < O(n^2) < O(2^n)$

Answer: D

118. Consider performing sequential search for an element in the array *A[1...n]*. The average number of elements that need to be checked to perform a successful search is

A. (n + 1) / 2

B. n(n + 1) / 2

C. lg n

D. n^2

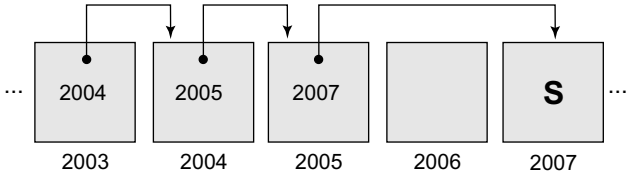
E. n(n + 1)(2n + 1) / 6

Answer: A

The element can be found in either one of the n locations A[1], A[2], A[3], ..., A[n], and the number of respective checks required are 1, 2, 3, ..., n. The total number of checks is therefore 1+2+3+...+n = n(n+1) / 2. However, above are n possible cases, so the average is obtained by dividing the total by n, giving (n+1) / 2.

119. What is indirection? Explain, indirection, double indirection, triple indirection.

Answer: Indirection is the process of accessing data through a pointer rather than accessing it directly. Double indirection occurs when you have a chain of two pointers pointing to a data cell such that pointer 1 refers to pointer 2, and pointer 2 refers to the data. This is called double indirection because you must follow two links to access the data. For triple indirection, you must follow three pointers to reach the stored data. The diagram below illustrates triple indirection.



120. What are sequential lists?

Answer: List structures that are implemented with arrays are known as *sequential lists*. These lists have two disadvantages. First, the list cannot contain gaps so we are often forced to shift many of the list items when we want to insert a new item. Second, the list cannot grow in size because it uses a fixed size array.

121. What happens when a stack realised using arrays becomes full?

Answer: We can not insert any more items till some items are popped. Usually, any attempt of pushing is raised as an error and program will be terminated.

122. Compare and contrast graphs with trees.

Answer: Graphs are similar to trees such that both the data structures are represented with nodes and edges. Graphs are different from trees such that graphs do not have restrictions on the relationships between nodes. For example, tree nodes can only have one parent node. Graphs do not have the concept of parent and child nodes.

123. In terms of space, why is heap sort attractive?

Answer: We can do both heapifying and reheaping in the same array. It is in-core method.

124. What is the maximum depth of a heap with n elements?

Answer: n.

125. A binary tree in which the heigths of the two subtrees of every node never differ by more than 1 is called as___.

Answer: Balanaced Tree

126. A complete binary tree in which the key value in each node is no smaller than the key values in its children. Is this definition correct?

Answer: No.

127. A matrix i.e, a two dimentional array which has many zeroes as its element is called as ___.

Answer: Sparse matrix

128. A two dimensional nxn array, say A, with the property that A[i][j] = 1 if the edge (i,j) (for a directed graph) is in E(G). Then A is called as __.

Answer: Adjacency matrix

129. An array of elements is sorted by choosing a pivot element and by partition.The sorting process is called as ___.

Answer: quick sorting

130. Binary searching requires data to be in what order?

Answer: Ascending order or descending

131. For implementation of recursion tree we use ___.

Answer: stacks

132. If a complete binary tree with n nodes is represented sequentially, then for any nodes with index i, the children are at

Answer: 2*i+1 and 2*i+2

133. If a complete binary tree is stored sequentially, then the number of empty elements before last element are___.

Answer: Zero

134. If an almost complete binary tree is stored sequentially, then the number of empty elements before last element are___.

Answer: Zero

135. If the records that it's sorting are in main memory then the sort is called as ___.

Answer: Internal sorting

136. In a binary tree that has all of its leaf nodes at the lowest level the tree is said to be _____.

Answer: Either complete binary tree or almost complete binary tree

137. In undirected graph in which every node is reachable from each other is termed as _____.

Answer: Source

138. In which of the data structure we can both add/remove at both ends but not in the middle?

Answer: De-Queue

139. The computing time of the recursive merge sort is __.

Answer: nlogn

140. The height of a complete binary tree with n nodes is ___.

Answer: log(n+1)

141. The matrix with meaningful elements along the diagonal is ___.

Answer: Band matrix. Diagonal matrix is a subcase of it.

142. The maximum number of nodes on level i of a binary tree is i=>1

Answer: 2^i

143. The maximum number of nodes in a binary tree of depth k is k=>1

Answer: $2^k - 1$

144. What is the best case running time of bubble sort?

Answer: $O(n^2)$

145. What is the complexity of re-heaping?

Answer: O(nlogn)

146. What kind of data structure would make insertion sort particular in-efficient?

Answer: linked list

147. For implementation of recursion tree we use___.
Answer : stack

148. In the addition of long positive integers using circular lists we use the____.
Answer : circular list or double linked list

149. We perform the folowing operation to traverse a non empty binary tree in an order
1. Traverse the left subtree in that so called order
2. Visit the root.
3. Traverse the right subtree in that so called order
Answer. inorder

150. If the records that it's sorting are in main memory then the sort is called as____.
Answer : internal

151. Given two functions f(n) & g(n),we say that f(n) is on the order of g(n) if there exist positive integers a & b such that f(n)<=a*g(n) for all n=>b. is called f(n)___.
Answer : O(g(n))

152. The efficiency of the buble sort is ___.
Answer : O(log n)

153. An array of elements is sorted by choosing a pivot element & by partition.The sorting process is called ___.
Answer : Partition Exchange Sort or Quicksort

154. The efficiency of the quicksort is ___.
Answer : O(nlogn)

155. The efficiency of the Heapsort is ___.
Answer : O(nlogn)

156. Is it possible to define an heap as an complete binary tree such that the content of each node is greater than or equal to the content of its father?
Answer : Yes, ascending heap or min heap

157. An acyclic graph in which every node has one or no predecessors may be defined as ___.
Answer : forest

158. The space requirements for the quicksort depend on the
Answer : number of nested recursive calls & size of the stack

159. The efficiency of Straight Selection Sort is ___.
Answer : O(n2)

160. The average sorting time for binary tree sort is ___.
Answer : O(nlogn)

161. To denote the empty stack the top is set to ___.
Answer : -1

162. The expression AB/C-DE*+AC*- is in ___.
Answer : postfix

163. The computing time of the recursive merge sort is__.
Answer : O(nlogn)

164. The maximum number of nodes in a binary tree of depth k is k=>1
Answer : (2*2*2*.....to k)-1

165. If a complete binary tree with n nodes is represented sequentially, then for any nodes with index i,1<= i <=n,we have LeftChild(i) is at if it is < or = n
Answer : 2i

166. A complete binary tree in which the key value in each node is no larger than the key values in its children
Answer : min heap

167. A two dimentional nxn array, say A, with the property that A[i][j] = 1 if the edge (i,j) (for a directed graph) is in E(G)
Ans : Adjacency matrix

OBJECTIVE TYPE QUESTIONS

1. We have a chocolate of rectangular size with mxn rectangular pieces. We need to divide into pieces along the grooves. You are not allowed to split two or more stacked pieces at a time. The approach which divides into individual pieces has
A. Best case complexity
B. Worse case complexity
C. Average complexity
D. Any trail needs same number of splits.

2. What is the complexity of third step in the following algorithm to find whether the graph is connected or not.
1. Let n is the number of nodes (vertexes) in the given graph G.
2. Take first node (vertex) in the set of nodes or vertexes (V(G)) and mark it as visited.
3. While there is an edge (i,j) that is member of set of edges (E(G)) with vertex i marked and j unmarked, mark j as visited.
4. If all the vertexes are marked, then graph G is declared as connected otherwise not.

A. O(1) B. O(2)
C. $O(n^2)$ D. O(n)
E. O(logn)

3. Number of possible spanning trees with n vertexes
A. !n B. 2^n
C. n^{n-2} D. n^2

4. Possible number of spanning trees with 4 vertexes

A. 4 B. 1

C. 16 D. None

5. Assuming that bubble sort is applied on the data: 12 10 3 37 57 2 23 9 . After first pass, largest element goes to last location, after second pass of the bubble sort algorithm second largest element goes to last but one place, and vice vers a. Where will be 3 after the completion of 3rd pass of the algorithm. Assume initially 12 is at 0th location, 10 at 1st location, and vice versa.

A. 0 B. 1

C. 2 D. 3

Answer:

```
Initially
12 10 3 37 57 2 23 9
Pass 1
10 12 3 37 57 2 23 9
10 3 12 37 57 2 23 9
10 3 12 37 2 57 23 9
10 3 12 37 2 23 57 9
10 3 12 37 2 23 9 57
Pass 2
3 10 12 37 2 23 9 57
3 10 12 2 37 23 9 57
3 10 12 2 23 37 9 57
3 10 12 2 23 9 37 57
Pass 3
3 10 2 12 23 9 37 57
3 10 2 12 9 23 37 57
Pass 4
3 2 10 12 9 23 37 57
3 2 10 9 12 23 37 57
Pass 5
2 3 10 9 12 23 37 57
2 3 9 10 12 23 37 57
```

6. Assuming that insertion sort is applied on the data: 12 10 3 37 57 2 23 9 . Where will be 37 after the completion of 3rd pass of the algorithm. Assume initially 12 is at 0th location, 10 at 1st location, and vice versa.

A. 0 B. 1

C. 2 D. 3

Answer:

```
Initially
12 10 3 37 57 2 23 9

Pass 1, (next = 10)
12 12 3 37 57 2 23 9
10 12 3 37 57 2 23 9

Pass 2, (next = 3)
10 12 12 37 57 2 23 9
10 10 12 37 57 2 23 9
 3 10 12 37 57 2 23 9

Pass 3, (next = 37)
 3 10 12 37 57 2 23 9

Pass 4, (next = 57)
 3 10 12 37 57 2 23 9

Pass 5, (next = 2)
3 10 12 37 57 57 23 9
3 10 12 37 37 57 23 9
3 10 12 12 37 57 23 9
3 10 10 12 37 57 23 9
3 3 10 12 37 57 23 9
2 3 10 12 37 57 23 9

Pass 6, (next = 23)
2 3 10 12 37 57 57 9
2 3 10 12 37 37 57 9
2 3 10 12 23 37 57 9

Pass 7, (next = 9)
2 3 10 12 23 37 57 57
2 3 10 12 23 37 37 57
2 3 10 12 23 23 37 57
2 3 10 12 12 23 37 57
2 3 10 10 12 23 37 57
2 3 9 10 12 23 37 57
```

7. Assuming that insertion sort is applied on the data: 12 10 3 37 57 2 23 9 . How many elements will be moving when 37 is inserted at its appropriate place? Assume initially 12 is at 0th location, 10 at 1st location, and vice versa.

A. 0 B. 1

C. 2 D. 3

8. Worst case complexity of quick sort

A. O(nlogn) B. O(logn)

C. $O(n^2)$ D. $O(n^2 \log n)$

9. Quick sort gives best behavior when pivot element is

A. Always first element B. Always median

C. Any element D. None

10. One way to build a heap is to start at the end of the array (the leaves) and push each new value up to the root. The respective recurrence relation for its time complexity

A. T(n)=T(n-1)+O(n)

B. T(n)=T(n-1)+O(1)

C. T(n)=T(n-1)+log(n)

D. None

11. One way to build a heap is to start at the end of the array (the leaves) and push each new value up to the root. Its time complexity is

A. O(n) B. O(nlogn)

C. $O(n^2 \log n)$ D. None

12. T(n)=2T(n/2)+logn is

A. O(n) B. O(nlogn)

C. $O(n^2 \log n)$ D. None

13. Let A[1::n][1::n] be a two-dimensional array of real numbers that is sorted both row-wise (A[x][y] < A[x][y + 1] for all x and y) and column-wise (A[x][y] < A[x + 1][y] for all x and y). Pick up the correct statement.

A. If we represent all the elements of 1st row, 2nd row, and vice versa, all elements will be in descending order.

B. If we represent all the elements of 1st row, 2nd row, and vice versa, all elements will be in ascending order.

C. If we represent all the elements of 1st row, 2nd row, and vice versa, all elements will be in descending order if all the elements are unique elements.

D. None

14. We have a complete graph with n vertexes. If we propose to add another vertex to it and resulting graph should also be complete graph, then the number of new edges that are to be added are

A. n–1 B. n
C. 2n D. None

15. A complete graph with n vertexes will be having ____ edges
A. n B. 2n
C. n(n–1)/2 D. n^2

16. Number of moves needed for n disks in Towers of Honoi problem to transfer n disks from one peg to another peg with an auxialary peg are
A. 2n B. 2n–1
C. 2^{n-1} D. NP complete problem

17. The recurrence relation: a0 = 5

$$a_n = na_{n-1}$$

is equivalent to
A. a_n=2n! B. a_n=5n!
C. a_n=2n D. a_n=2n!
E. None

18. Big-oh complexity of an algorithm whose time complexity is
T(1) = a, and T(n) = b + T(n–1) for n > 1
A. O(2n) B. O(n)
C. O(n!) D. $O(n^2)$

19. What is the order of growth of the running time of an algorithm if its running time is:
$T(n) = [n(n \log n + n^2 + 3) + \log(0.5n)] / n^2 + 3.14$
A. 3.14n B. 3.14^n
C. n D. None

20. What is the order of growth of the running time of an algorithm if its running time is:
$T(n) = 3.14^n + \cos(n^*60)$
A. 3.14n B. 3.14^n
C. none D. None

21. _____ data structure is preferred to access element through its index.
A. Linked list B. Tree
C. B-Tree D. Array

22. If one wants to add and delete elements quickly without reshuffling the rest
A. Linked list B. Tree
C. B-Tree D. Array

23. To access first element quickly
A. Linked list B. Queue
C. B-Tree D. Stack

24. To access last element quickly
A. Linked list B. Queue
C. B-Tree D. Stack

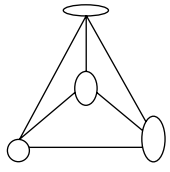
25. If the recursive call keeps calculating the same things over and over again, we can use ___ which stores partial results already calculated and to be used again.
A. Divide and conquer algorithm
B. Recursive
C. Dynamic programming
D. None

26. The tree that is used for Huffman coding of symbols is
A. Binary B. Balanced
C. AVL D. B-Tree

27. A Binany Search Tree search complexity is log2N, where N is the number of elements that are maintained as Binany Search Tree. If, we change number of childs of the nodes to c instead of 2 in Binany Search Tree then search complexity becomes
A. $(\log_2 N)/c$ B. $\log_2(N/c)$
C. $\log_c N$ D. None

28. The Binary Heap is a priority queue that allows insertion of new items and the removal of the minimum item in
A. Constant time B. Linear time
C. Logarithmic time D. None

29. Four cities are connected by a road network as shown in the figure. In how many ways can you start from any city and come back to it without travelling on the same road more than once ?



A. 8 B. 10
C. 12 D. None

30. Worst case time complexity of insertion sorting is
A. O(n) B. O(nlogn)
C. $O(n^2)$ D. $O(n^3)$

31. An isolated vertex degree
A. 0 B. 1
C. 2 D. None

32. A tree with n vertexes or nodes will be having ___ edges
A. n B. n–1
C. n+1 D. n^2

33. _____ is a technique that determines the solution by systematically searching the solution space for the given problem
A. Dynamic B. Backtracking
C. Greedy D. None

34. What is the big-oh complexity for the following code fragment?

```
for (int i = 0; i < 1000; i++) a[i] = i * i;
```

A. O(1000) B. O(1)
C. O(n) D. None

35. Heap

A. A heap is an area of memory from which the programmer can allocate storage.
B. A heap is a binary tree in which every node has the heap property. A node has the heap property if the value in the node is at least as large as the value in any child of that node.
C. Both are valid
D. None

36. An algorithm which "remembers" previous results and uses them when computing new results.

A. Devide and conquer
B. Dynamic programming
C. Travelling salesman problem
D. Recursive mehods

37. The following code counts the number of digits in a positive integer variable number:

```
int n = number;
int count = 0;
while (n > 0) {
n = n / 10;
count++;
}
```

How long (in Big-O terms) does this method take?

A. O(1)
B. O(log(n))
C. O(32767) in the case of 16 bit computer
D. n in the case of n-bit computer
E. O(n)

38. Number of connected components in a Binary Search Tree with N nodes

A. 0 B. 1
C. N D. N–1

39. By adding a new node to a Binary Search Tree with N nodes, the number connected components in it changes to

A. 0 B. N+1
C. N–1 D. No change

40. A graph without self-loops or parallel edges is called

A. Simple graph B. General graph
C. Acyclic graph D. None

41. The algorithm which requires fixed amount of storage is

A. Heap sort B. Quick sort
C. Linked list D. None

42. The data structure in which part of the data resides in main memory and part in secondary memory

A. Binary Search Tree B. Linked list
C. B-Tree D. Red-block trees

43. Internal hashing complexity

A. O(1)
B. O(n), where n is number of records
C. O(b), where b is number of buckets
D. None

44. A set of values (23, 18, 29, 28, 39, 13, 16, 42, 17) are stored in an array having 11 locations (array indexes starts from 0) using linear probing and hashing function of key modulus number of locations. Possible locations of 17 and 29 are:

A. 1,8 B. 0,8
C. 2,8 D. 3,9

45. A set of values (23, 18, 29, 28, 39, 13, 16, 42, 17) are stored in an array having 11 locations (array indexes starts from 0) using quadratic probing and primary hashing function of key modulus number of locations is used. Possible locations of 17 and 29 are:

A. 1,8 B. 4,8
C. 2,8 D. 3,9

46. Finding minimum or maximum out of the numbers that are stored using hashing is

A. O(1) B. O(n)
C. $O(n^2)$ D. None

47. Find invalid statement

A. In hashing, complexity of insertion, deletion, search operations is O(1)
B. In Binary Search Tree complexity of insertion, deletion, search operations is O(nlogn)
C. In sorted array, complexity of insertion, deletion, search operations is O(n)
D. None

48. Proven lower bound for comparision based sorting

A. O(n) B. O(nlogn)
C. $O(n^2)$ D. $O(n^{1.67}\log n)$

49. Find odd man out of the following

A. Bucket sorting B. Counting sorting
C. Bubble sorting D. Radix sorting

50. Linear sorting method

A. Radix sorting B. Bubble sorting
C. Heap sorting D. Quick sorting

51. Worst and average case complexity of merge sorting
A. $O(n^2)$ B. O(n)
C. O(nlogn) D. None

52. Worst and average case complexity of heap sorting
A. $O(n^2)$ B. O(n)
C. O(nlogn) D. None

53. The array operation that has an asymptotic complexity of O(1)
A. Inserting an element
B. Deleting an element and moving others towards 0th element
C. Accessing an element
D. None

54. What is the complexity of the following code fragment in big-oh notation?

```
for (int i=0; i<=n/2; i += n/2) {
        for (int k=0; k<n/2; k++) {
                print (i + k);
        }
}
```

A. O(n/2) B. O(3n)
C. O(nlogn) D. O(n)

55. What is the complexity of the following code fragment in big-oh notation?

```
for (int k=0; k<n-1; k++) {
        for (int m=1; m<n; m*=2)
                print(k*m);
}
```

A. O(n/2) B. O(3n)
C. O(nlogn) D. O(n)

56. In hashing, number of buckets, N Should be chosen such that
A. It is even
B. It is odd
C. It is the largest prime which is less than or equal to number items to be stored
D. It is should be factor to number items to be stored

57. In hashing, if loading factor is greater than 1 then
A. No collisions
B. Probability of collision is 0.01
C. Probability of collision is 1
D. None

58. $\sum_i^n 1/2^i$
A. 0 B. 1
C. <1 D. None

59. Any set of regions defined by n lines in the plane can be coloured with minimum of ___ colours such that no two regions that share an edge will be having same colour.
A. 2
B. 3
C. 4
D. More than 2

60. T(n) = 2T(n/3) + dn
A. $O(n^2)$ B. O(2n)
C. O(n) D. None

61. T(n)=2T(n/2)+dn
A. $O(n^2)$ B. O(nlogn)
C. O(n) D. None

62. T(n)=4T(n/2)+dn
A. $O(n^2)$ B. O(nlogn)
C. O(n) D. None

63. What is the average case big-oh cost of adding a new item to a set of n items that are maintained in hash-table using linear probing and table is less than half full
A. O(n) B. O(0)
C. O(1) D. None

64. If we are trying to add an item to a hash table that uses open addressing and if first cell which we look is already contains an item where will we put the new item
A. In the same cell
B. In the last cell
C. In the next free cell that is right to the first cell
D. None

65. When hash table that uses linear probing will be showing worst performance?
A. When the table is almost full
B. When the table is half full
C. While inserting first item
D. While inserting last item even if the table has 50% empty cells

66. If 15 elements are maintained in Binary Search Tree fashion then minimum and maximum possible heights
A. 3,4 B. 4,15
C. 4,8 D. None

67. The worst case big-oh (asymptotic) cost of searching an element in a Binany Search Tree of depth d
A. O(logn) B. O(d)
C. $O(2^d)$ D. None

68. What property must be true of a Binany Search Tree for insertion/access/deletion to have O(log(n)) complexity in all cases?

A. It should be skewed

B. It should be balanced such that all of its leaf nodes will be at lowest level or one level above the lowest level

C. It should be strictly binary

D. None

69. Run-time analysis of a program involves

A. Counting the number of arithmetic and other operations required for the program to run

B. Accounting the number of megabytes required for the program to run

C. Accounting the number of seconds required for the program to run

D. Accounting both the number of seconds plus the memory needed for the program

E. Accounting the number of seconds times the number of megabytes needed for the program

70. What is the worst-case time complexity of the following loop that uses an integer variable n?

```
while (n > 0){
   n/=10;
}
```

A. O(1) B. O(log n)

C. O(n) D. $O(n^2)$

71. Tree algorithms typically run in time O(d). What is d?

A. The depth of the tree.

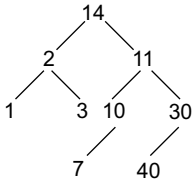
B. The number of divisions at each level.

C. The number of entries in each node.

D. The number of nodes in the tree.

E. The total number of entries in all the nodes of the tree.

72. Consider the following binary tree which is traversed in inorder manner using the following recursive approach. Assuming that currently most recent recursive call is made with the node value 11, how many activation records are seen on the stack including the most recent one.

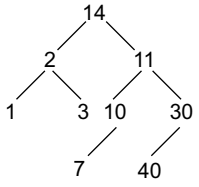



```
int inord(struct Node *H){
if(H){
   inord(H->left);
   printf("%d\n", H->n);
   inord(H->right);
}
}
```

A. 1 B. 2

C. 3 D. 4

73. Consider the following binary tree which is traversed in inorder manner using the following recursive approach. Maximum how many activation records are seen simultaneously in the stack during the execution of the function.




```
int inord(struct Node *H){
if(H){
   inord(H->left);
   printf("%d\n", H->n);
   inord(H->right);
}
}
```

A. 1 B. 2

C. 3 D. 5

74. A language supports array indexes starting from zero. Let A is its base address and employs row major order of storage and the array is 2D array; then find the valid one

A. If address of first element of any row is given then we can find out 2D array dimensions

B. If addresses of two elements (along with their row & column indexes)of any row is given then we can find out 2D array dimensions

C. If addresses of any two elements (along with their row & column indexes)of any row is given then we can find out 2D array column size

D. Addresses of any two elements along with their row & column indexes are adequate to find any dimensional array sizes.

75. If the array $A(b_1:e_1, b_2:e_2, \ldots\ldots, b_n:e_n)$ is stored in row major order, in the language, array name A itself pointer to the array, then $A(i_1,i_2,\ldots i_n)$ can be located at:

A. $A+ i_1*(e_1-b_1) + i_2*(e_2-b_2) + \ldots\ldots + i_n*(e_n-b_n)$

B. $A+ (i_1-b_1)* (e_1-b_1) + (i_2-b_2)*(e_2-b_2) + \ldots\ldots + (i_n-b_n*)(e_n-b_n)$

C. $A + \Sigma_{i=1}^{n} (i_j - b_j)a_j$ where
$a_j = \Pi_{k=j+1}^{n} (e_k - b_k + 1)$, $1 <= j < n$ and $a_n = 1$

D. $A + \Sigma_{i=1}^{n} (i_j - b_j - 1)a_j$ where
$a_j = \Pi_{k=j+1}^{n} (e_k - b_k + 1)$, $1 <= j < n$ and $a_n = 1$

76. Running time of an algorithm is independent of

A. OS B. Input size
C. HW D. None

77. Big O complexity of the following code fragment

```
for(i=4;i<n;i++){
x=x*f;
while(i<4)i++;
}
```

A. O(n) B. $O(n^2)$
C. O(4n) D. None

78. What is the time complexity of the following code fragment?

```
int fact(unsigned n){
if(n)return 1;
else
return n*fact(n-1);
}
```

A. O(n) B. O(log(n))
C. O(1) D. $O(n^2)$

79. What is the time complexity of the following code fragment?

```
int Fib(unsigned n){
if(n<=1)return 1;
else
return Fib(n-1)+Fib(n-2);
}
```

A. Linear B. Cubic
C. exponential D. O(n!)

80. _____ complexity is exponential time complexity

A. Linear search algorithm
B. Brute force
C. Nested loops
D. Recursive calls

81. Binary searching can not be applied on a single linked list as

A. We can not index nodes of a linked list
B. We can index nodes of a linked list
C. We can not move up-wards in a single linked list
D. None

82. Time complexity of finding middle node of a single linked list

A. O(1) B. O(log(n))
C. O(n) D. O(n/2)

83. Time complexity of the following code fragment

```
for(i=1;i<n;i*=3)
for(j=1;j<n;j*=5)
for(k=1;<n;k++) { s++;}
```

A. O(n) B. O(nlogn)
C. $O(n\log^2 n)$ D. $O(n^2)$

84. Minimum possible hight of a binary tree that is formed by adding 13 keys

A. 4 B. 3 C. 4.5 D. 5

85. What is the benefit of using a linked-list over an array-based list?

A. No preset size limit
B. No "shifting" to move data around on insertion or removal, which is good when copying is expensive
C. Both a and b
D. None of the above

86. Removing an item from an arbitrary position in an array-based list takes worst-case

A. O(1) B. O(lgn)
C. O(n) D. O(nlgn)

87. Removing last item from an array-based list takes worst-case

A. O(1) B. O(lgn)
C. O(n) D. O(nlgn)

88. Assuming a binary tree creation does not explicitly keep track of its depth, what is the best Big-O running time of the function that calculates the depth of the tree?

A. O(1) B. O(lgn) C. O(n) D. O(nlgn)

89. A 2-3 Tree is a type of

A. Binary search tree B. Search tree
C. Binary tree D. AVL tree

90. Number of data items in a leaf node of a 2-3 tree

A. 1 B. 1 or 2
C. 2 D. 2 or 3

91. A 2-3 Tree storing n items will generally have a smaller depth than a binary search tree storing n items, but will require about the same number of comparisons to traverse from root to leaf.

A. True
B. False, it will require fewer comparisons
C. False, it will require more comparisons
D. There is no way to compare the two structures

92. The partition step of Quick-sort can be done without allocating additional storage in O(lgn) steps

A. True
B. False, it requires O(n) steps
C. False, it cannot be done in place but does take O(lgn) steps
D. False, it cannot be done in place and requires O(n) steps

93. What happens if delete p; statement is executed?
A. p is deleted from the heap.
B. p is deleted from the stack.
C. The variable which p points to is deleted but p's value remains the same
D. All variables that point to p are deleted, p remains unaffected.

94. All recursive algorithms must have a base case
A. True B. False

95. Which of the following data structures would be most appropriate for representing path of a traveller who will take the reverse path on the return trip?
A. A linked list B. A stack
C. A queue D. A tree

96. Find the correct statement
A. n^3 grows faster than $n^2 \log_{42} n$
B. n grows faster than nlogm where logm is a constant
C. 2^{2n} grows faster than 3^n

97. For a binary search tree, which of the following can be implemented in O(1)
A. Size B. Insert
C. Max D. isEmpty

98. For a binary search tree, which of the following would be easiest to implement without Recursion.
A. Size B. Depth
C. Max D. None of the above

99. An adjacency matrix for a graph with n nodes
A. Uses $O(n^2)$ space and gives O(1) lookup to find edges
B. Uses O(n) space and gives O(1) lookup to find edges
C. Uses $O(n^2)$ space and takes O(n) lookup to find edges
D. Uses O(n) space and takes O(n) lookup to find edges

100. Which of the following is the run-time for the dequeue operation on a linked-list-based queue?
A. O(1) B. O(log n)
C. O(n) D. O(n log n)
E. $O(n^2)$

101. Which of the following is the average-case run-time for the insert operation on a Binany Search Tree?
A. O(1) B. O(log n)
C. O(n) D. O(n log n)
E. $O(n^2)$

102. Which of the following is the run-time for QuickSort if the largest value in the list is always chosen as the pivot?
A. O(1) B. O(log n)
C. O(n) D. O(n log n)
E. $O(n^2)$

103. Two important efficiency measures of an algorithm
A. Processor and memory
B. Complexity and capacity
C. Time and space
D. Data and space

104. The time factor when determining the efficiency of algorithm is measured by
A. Counting nano seconds involved
B. Counting the number of key operations
C. Counting the number of kilos of statements
D. Counting the kilobytes of algorithm

105. The space requirements of an algorithm is represented by
A. Counting the maximum memory needed by the algorithm
B. Counting the minimum memory needed by the algorithm
C. Counting the average memory needed by the algorithm
D. Counting the maximum disk space needed by the algorithm

106. Which of the following case does not exist in complexity theory?
A. Best case B. Worst case
C. Average case D. Null case

107. The Worst case behavior of linear search algorithm is seen when
A. Item is somewhere in the middle of the array
B. Item is not in the array at all
C. Item is the last element in the array
D. Item is the last element in the array or is not there at all

108. The average case occur in linear search algorithm
A. When item is somewhere in the middle of the array
B. When the given item is not in the array.

C. When item is the last element in the array
D. When item is the last element in the array or is not there at all

109. The complexity of the average case of an algorithm is
A. Much more complicated to analyze than that of worst case
B. Much more simpler to analyze than that of worst case
C. Sometimes more complicated and some other times simpler than that of worst case
D. None of the above

110. The indirect change of the values of a variable in one module by another module is called
A. Internal change
B. Inter-module change
C. Side effect
D. Side-module update

111. The operation of processing each element in the list is known as
A. Sorting B. Merging
C. Inserting D. Traversal

112. Finding the location of the element with a given key value is:
A. Traversal B. Search
C. Sort D. None of above

113. Arrays are best data structures
A. For relatively permanent collections of data
B. For the size of the structure and the data in the structure are constantly changing
C. For both of the above situation
D. For none of the above situation

114. Linked lists are best suited
A. For relatively permanent collections of data
B. For the size of the structure and the data in the structure are constantly changing
C. For both of above situation
D. For none of above situation

115. The elements of an array are stored successively in memory cells because
A. By this way computer can keep track only the address of the first element and the addresses of other elements can be calculated
B. The architecture of computer memory does not allow arrays to store other than serially
C. Both of above
D. None of above

116. Which of the following data structures are indexed structures?
A. Linear arrays B. Linked lists
C. Both of above D. None of above

117. Which of the following is not the required condition for binary search algorithm?
A. The list must be sorted
B. There should be the direct access to the middle element in any sub-list
C. There must be mechanism to delete and/or insert elements in list
D. None of above

118. Which of the following is not a limitation of binary search algorithm?
A. Must use a sorted array
B. Requirement of sorted array is expensive when a lot of insertion and deletions are needed
C. There must be a mechanism to access middle element directly
D. Binary search algorithm is not efficient when the data elements are more than 1000.

119. Which of the following data structure can't store the non-homogeneous data elements?
A. Arrays B. Records
C. Pointers D. None

120. Which of the following data structure store the homogeneous data elements?
A. Arrays B. Records
C. Pointers D. None

121. Which of the following statement is false?
A. Arrays are dense lists and static data structure
B. Data elements in linked list need not be stored in adjecent space in memory
C. Pointers store the next data element of a list
D. Linked lists are collection of the nodes that contain information part and next pointer

122. Binary search algorithm can not be applied to
A. Sorted linked list
B. Sorted binary trees
C. Sorted linear array
D. Pointer array

123. When new data are to be inserted into a data structure, but there is no available space; this situation is usually called
A. Underflow B. Overflow
C. Housefull D. Saturated

124. Which of the following is two way list?

A. Grounded header list

B. Circular header list

C. Linked list with header and trailer nodes

D. None of above

125. When in-order traversing a tree resulted E A C K F H D B G; the preorder traversal would return

A. FAEKCDBHG B. FAEKCDHGB

C. EAFKHDCBG D. FEAKDCHBG

126. To represent hierarchical relationship between elements, which data structure is suitable?

A. Deque B. Priority

C. Tree D. All of above

127. In a binary tree, certain null entries are replaced by special pointers which point to nodes higher in the tree for efficiency. These special pointers are called

A. Leaf B. Branch

C. Path D. Thread

128. The inorder traversal of tree will yield a sorted listing of elements of tree in

A. Binary trees B. Binary search trees

C. Heaps D. None of above

129. A _____ tree is a tree where for each parent node, there is only one associated child node.

A. AVL tree

B. Rooted complete binary tree

C. Complete binary tree

D. Degenerate tree

130. In a heap, element with the greatest key is always in the _____ node.

A. Leaf

B. Root

C. First node of left sub tree

D. First node of right sub tree

131. In _____tree, the heights of the two child subtrees of any node differ by at most one.

A. Binary tree B. Red block tree

C. Splay tree D. AVL tree

132. It is observed that an algorithm with problem size of 10 to be taking 1ns and if we increase the problem size by ten fold, algorithm is observed to be taking 2 ns, then probable order of the algorithm

A. Linear B. Quadratic

C. Logarithmic D. Exponential

133. It is observed that a quadratic complexity algorithm with problem size of 10 to be taking 10ns and if we increase the problem size by ten fold then probable time it takes is:

A. 100ns B. 1000ns

C. 10000ns D. 500ns

134. If non lead nodes of a binary tree contains exactly two children then it is

A. Totally balanced tree

B. Complete binary tree

C. Both

D. None

135. Find in correct one

A. A degenerate tree is like a single linked list.

B. A degenerate tree will be having its non-lead nodes with 2 childs.

C. A degenerate tree is also called as totally unbalanced tree

D. If n keys are organized in a degenate tree fashion, we get worst case height

136. From x = 1, how many times should it be doubled before $x \le N$?

A. N/2 B. N^2

C. $\log_2(N)$ D. None

137. From x = N, how many times should it be halved before $x \le 1$?

A. N/2 B. N^2

C. $\log_2(N)$ D. $\log_{0.5}(N)$

138. Worst case complexity of interpolation search is

A. O(N) B. log(N)

C. O(log logN) D. None

139. Assume we have coins of denomination 1, 5, 10, 21 and 25 paisa then minimum and maximum number of coins needed to make 63 paisa is

A. 5,3 B. 7,3

C. 6,3 D. Can not say

140. If u and n are leaves (with a minimum degree of 2) of a depth first traversal tree of an undirected graph G with, then which one of the following statements is true?

A. There must be a vertex w adjacent to both u and n in G

B. There must be a vertex w whose removal disconnects u and n in G

C. There must be a cycle in G containing u and n

D. There must be a cycle in G containing u and all its neighbours in G.

141. Partition is useful in solving all of the following problems, except

A. Finding the k smallest elements
B. Quicksort
C. Selection
D. Splitting an array for Mergesort

142. Which of the following sorts is not based on key comparisons?
A. Insertion-Sort B. LSD Radix-Sort
C. Mergesort D. Quicksort

143. If the list contains same elements then complexity of max heap creation will be
A. $\Theta(1)$ B. $\Theta(n)$
C. $\Theta(n \lg n)$ D. $\Theta(n^2)$

144. The worst-case time complexity for finding maximum key in an ascending ordered circular doubly-linked list with n nodes is
A. $\Theta(1)$ B. $\Theta(\log n)$
C. $\Theta(n \log n)$ D. $\Theta(n)$

145. A variant of quick sorting spends an algorithm of O(n) to select pivot element and then applies normal quick sorting method. Then its complexity of quick sort variant can be said as
A. O(nlogn) B. $O(n^2)$
C. $O(n^3)$ D. $O(n^2 \log n)$

146. Given two arrays of numbers a_1,....., a_n and b_1,....., b_n where each number is 0 or 1, the fastest algorithm to find the largest span (i, j) such that $a_i + a_{i+1} + \ldots\ldots\ldots + a_j = b_i + b_{i+1} + \ldots\ldots\ldots + b_j$
A. $O(n^2)$ B. $O(n^3)$
C. O(nlogn) D. $O(2^n)$

147. In a binary tree, an edge is the one which joins two nodes of two adjacent levels. Maximum possible edges of a tree with height h is
A. h–1 B. h
C. 2^h D. None

148. Maximum possible binary trees with three unlabelled nodes is
A. 1 B. 2
C. 3 D. 5

149. Sorting algorithm with lowest worst case complexity
A. merge sort B. bubble sort
C. quick sort D. heap sort

150. An n-way tree with three levels is having its nodes such that they have either n children or no children. Number of leaf nodes are 41 and internal nodes with n children are 10. What is the value of n?
A. 2 B. 3
C. 4 D. 5

151. The least integer k such that $f(n) = (n^4+n^2+1)/(n^4+1)$ is $O(n^k)$
A. 1 B. 0
C. 4 D. None

152. The least integer k such that $f(n) = (n^3+5\log n)/(n^4+1)$ is $O(n^k)$
A. 1 B. 0
C. –1 D. 3

153. Best case complexity of find second maximum of a set of numbers is
A. O(n) B. O(logn)
C. O(n+logn) D. None

154. _____ comparisons are necessary in the worst case to merge two sorted lists containing n elements each.
A. n–1 B. n^2
C. 2n–1 D. n–1

155. A tree contains its internals nodes such that they contain exactly 3 children. Assuming N^i is the number of nodes and N_L leaf nodes then acceptable relation among N_i and N_L is
A. $N_i = N_L + 1$ B. $N_i = 2N_L + 1$
C. $N_L = 2N_i + 1$ D. None

156. Find the correct statement
A. In an undirected graph with positive edge weights, the shortest edge in the graph always belongs to any tree of shortest paths, provided the edge weights are distinct.
B. To find the longest path between 2 given vertices in a graph G with positive weights, we can change the weight of every edge e from w(e) to k – w(e), where k is a value larger than any edge weight in G, and then find the shortest path in the resultant graph.
C. If T is a tree of shortest paths from vertex s in a graph G, then T is also a tree of shortest paths from vertex s in a graph G' obtained by increasing the weight of every edge by same value C.
D. None

157. Define two operations
T1: complement any two bits
T2 exchange any two bits
starting with 000000 which of the following cannot be generated using sequence T1 and T2?
A. 110 000 B. 001 001
C. 100 001 D. 000 010
E. 001 100

158. Queue with permitted operations
(i) insertion at tail of queue

(ii) output the head of queue
(iii) place the head at the tail

For an input string of 1 2 3 4 5 6 which of the following outputs are possible?

(I) 3 4 1 2 6 5
(II) 5 2 1 3 4 6
(III) 5 6 1 2 3 4

A. I only B. II only
C. III only D. I & III
E. I, II & III

159. Sometimes the object module produced by a compiler includes information (from the symbol table) mapping all source program names to their address. The most likely purpose of this information is

A. For use as input to a debugging aid.
B. To increase the run time efficiency of the program.
C. For the reduction of the symbol table space needed by the compiler.
D. To tell the loader where each variable belongs.
E. To tell the OS what to call the variables.

160. If a, b, c occur with equal probability, which of the following is a valid Huffman coding of these symbols

A. a = 1, b = 0, c = 11
B. a = 0, b = 111, c = 000
C. a = 0, b = 10, c = 11
D. None of the above

161. In an AVL tree with 1000 nodes, path length is the length of a path from root to a leaf node.

A. At least one path has path length >100.
B. All paths have path length >100.
C. No path has path length >100.
D. Cannot be determined from the above info.

162. If n is a power of 2, then the minimum number of multiplications needed to compute a^n.

A. lg n B. sqrt(n)
C. n–1 A. n

163. Adjacency matrix of a graph is having 1's in off diagonal while remaining all elements are same. Then the graph

A. Contains all isolated nodes
B. V/2 connected components
C. If odd no of nodes are there then V/2+1 connected components
D. If odd no of nodes are there then one isolated node will be available
E. B & C & D

164. If T is a full binary tree with r internal nodes then

A. It will have r+1 terminals
B. 2r+1 total vertices
C. A & B
D. None

165. A graph is connected if it contains

A. One connected component
B. More than one connected component
C. If it has a spanning tree
D. A & C

166. If n is even, and assuming that all A[i] are distinct, what does the execution of the code below result in :

```
for (i = 0; i < n; i++)
A[i] = A[n+1-i];
```

A. It results in 2 copies of each value data
B. The values remain unchanged.
C. The array reverses.
D. None of the above

167. int a[10]; Here a is __ pointer

A. Wild B. Constant
C. Dangling D. None

168. A graph is having adjacency matrix of all 1's then

A. It is fully connected
B. Its path matrix is same is adjacency matrix
C. Contains cycles
D. All

169. Complexity of generating all possible subsets of a set of data

A. O(n) B. $O(n^2)$
C. $O(2^n)$ D. O(n!)

170. Complexity of splitting a set of data in half repeatedly and traversing each half

A. O(lg n) B. O(n lg n)
C. $O(n^2 \lg n)$ D. None

171. Which is having highest complexity $O(n^2)$, $O(n^2 \lg n)$, O(n!), $O(2^n)$?

A. $O(2^n)$ B. $O(n^2)$
C. $O(n^2 \lg n)$ D. O(n!)

172. $T_1(n) = 3n \lg n + \lg n$, $T_2(n) = 2^n + n^3 + 25$, $T_3(n,k) = k + n$, k<= n. The highest ordered one is

A. T_1
B. T_2
C. T_3
D. All are same order

173. A graph's adjacency matrix is as follows. Then the graph

0100
0010
0001
1000

A. Directed
B. All nodes are a cycle
C. One connected component
D. All

174. Given a machine with only a stack whose top can be output and on which POP and PUSH are allowed. which of the following strings can be sorted in ascending order

A. 4312 B. 3421
C. 2134 D. 1243
E. 3142

175. Time complexity in big-oh notation of the following recursive relation is :

$T(n)= 2T(n/2) + n \quad n>1$
$= 1 \quad n=1$

A. nlogn + n g B. n^2
C. nlogn –n +2 D. nlogn

176. The number of distinct strings of length 3 that can be obtained using a, a, b, b, c is

A. 7 B. 6
C. 12 D. 15
E. 18

177. In connection of n processors find the minimum. number of connections needed to provide two distinct paths between any two processors.

A. 2^n B. n
C. n+1 D. n(n+1)/2
E. n–1

178. The number of stack locations needed for evaluating (((a+b)*e)–d) using a stack is

A. Four B. Five
C. Six D. Three
E. Seven

Answer: While converting the same into postfix, we may be using at most 4 locations of the stack. While evaluating the postfix expression, the stack will be using at most 2 locations only. Thus, answer is four.

179. We have the recurrence relation

T(n) = 2T(n-1) - 1

What is the order of T(n)?

A. Linearly
B. Quadratic
C. Cubic
D. Exponential
E. Logarithmic

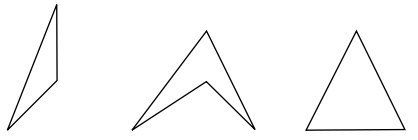
Answer: See the notes. If we expand we get assuming n as 5 we get the above relation as 16T(n–5)-8-4-2-1. Is sum will be -2^5. Thus, we can say order as $O(2^n)$.

ANSWER KEY

1. D	**2.** C	**3.** C	**4.** C
5. A	**6.** D	**7.** A	**8.** C
9. B	**10.** C	**11.** B	**12.** A
13. D	**14.** B	**15.** C	**16.** C
17. B	**18.** B	**19.** C	**20.** B
21. D	**22.** A	**23.** B	**24.** D
25. C	**26.** A	**27.** C	**28.** C

29. C

Consider the top city, the following are the 3 routes possible, starting from the leftmost edge. Since thereare 3 edges emanating from each city and the figure is perfectly symmetrical, these 3 routes are possible from each edge, hence for any given city, the total number of routes = 4 * 3 = 12.



30. C	**31.** A	**32.** B	**33.** B

34. B

O(1), since the number of iterations is fixed, and each pass through the loop does only constant-time operations.

35. C	**36.** B	**37.** B	**38.** B
39. D	**40.** A	**41.** A	**42.** C
43. A	**44.** B	**45.** B	**46.** B
47. C	**48.** B	**49.** C	**50.** A
51. C	**52.** C	**53.** C	**54.** D
55. C	**56.** C	**57.** C	**58.** C
59. A	**60.** C	**61.** B	**62.** A
63. C	**64.** C	**65.** A	**66.** B
67. C	**68.** B	**69.** D	**70.** B
71. A	**72.** B	**73.** D	**74.** C
75. C	**76.** D	**77.** A	**78.** C
79. C	**80.** B	**81.** A	**82.** C
83. C	**84.** B	**85.** C	**86.** C

87. A	**88.** C	**89.** B	**90.** B
91. A	**92.** A	**93.** C	**94.** A
95. B	**96.** A	**97.** D	**98.** C
99. A	**100.** A	**101.** B	**102.** C
103. C	**104.** B	**105.** A	**106.** D
107. D	**108.** A	**109.** A	**110.** C
111. D	**112.** B	**113.** A	**114.** B
115. A	**116.** A	**117.** C	**118.** D
119. B	**120.** A	**121.** A	**122.** C
123. B	**124.** D	**125.** B	**126.** C
127. D	**128.** B	**129.** D	**130.** B
131. D	**132.** C	**133.** C	**134.** C
135. B	**136.** C	**137.** C	**138.** A
139. ?	**140.** A	**141.** D	**142.** B
143. C	**144.** A	**145.** D	**146.** B
147. A	**148.** D	**149.** A	**150.** D
151. A	**152.** C	**153.** C	**154.** C
155. C	**156.** D	**157.** D	**158.** E
159. A	**160.** C	**161.** C	**162.** A
163. E	**164.** B	**165.** D	**166.** C
167. B	**168.** D	**169.** C	**170.** B
171. A	**172.** B	**173.** B	**174.** A,C,D
175. D	**176.** B	**177.** E	**178.** A
179. D			

Previous Years' GATE Questions

1. Consider the polynomial

$p(x)=a_0+a_1x+a_2x^2+a_3x^3$, where $a_i \neq 0$ for all i. The minimum number of multiplications needed to evaluate p on an input x is **(GATE 2006)**

A. 3 B. 4

C. 6 D. 9

Explanation: A. We can write $p(x) = a_0 + x(a_1 + x(a_2x^2 + a_3x))$. If one observes, we need three multiplications only. This is, called as Horners method.

2. In a binary max heap containing n numbers, the smallest element can be found in time **(GATE 2006)**

A. O(n) B. O(logn)

C. O(log logn) D. O(1)

Explanation: A. We know that in maximum heap, every node will be having its values more than its children. Thus, root node will be the largest among all. If we assume the elements are stored in array, then 0^{th} element becomes maximum. That is, to find maximum we need constant time, O(1). However, to find minimum we need to search all elements. Thus, the complexity becomes O(n).

3. Consider a weighted complete graph G on the vertex set [v1, v2, ….vn] such that the weight of the edge (vi, vj) is 2|i-j|. The weight of a minimum spanning tree of G is **(GATE 2006)**

A. n-1 B. 2n–2

C. n/2 D. n^2

Explanation: B. We know minimum spanning tree (MST) will be having n-1 minimum edges joining n vertexes. Evidently, MST in this case will be having edges (v1,v2), (v2,v3),…..(vn-1, vn). According to the given statement, weight of the each of the edges is 2. Thus, weight of the MST becomes n-1 times 2. That is, 2n–2.

4. To implement Dijkstra's shortest path algorithm on unweighted graphs so that it runs in linear time, then the data structure to be used is

A. Queue B. Stack

C. Heap D. B-Tree

5. A scheme for storing binary tree in an array X is as follows. Indexing of X starts at 1 instead of 0. The roots is stored at X[1]. For a node stored at X[i], the left child, if any, is stored in X[2i] and right child if any in X[2i+1]. To be able to store any binary tree on n vertices the minimum size of X should be

A. $\log_2 n$ B. n

C. 2n+1 D. 2n–1

Explanation: D. We know that height of the tree is minimal when n keys are organised in a complete binary tree fashion and if they are organised in a skewed tree then the height will be worst case height which is same as n. That is, the tree can be left skewed or right skewed. In these trees, every node contains exactly one child. Rather, all the nodes are in single linked list fashion with either left or right links only defined. As the given organisation uses array to store nodes, we find many nodes will not be defined if the values are maintained in skewed tree fashion. So, many locations in the array are empty. As worst case height is n, total elements of the array becomes 2n-1. Number of un-used elements of the array:2^n–1–n. Ratio of occupancy of the array: (n)/(2^n–1). Ratio of wastage: (2^n–1–n)/(2^n–1).

6. Which one of the following in place sorting algorithms needs the minimum number of swaps?

A. Quick sort B. Insertion sort

C. Selection sort D. Heap sort

7. Consider the following C program fragment in which i,j, and n are integer variables.

for(i=n,j=0;i>0;i/=2,j+=i);

Let val(j)= denote the value stored in the variable j after termination of the for loop. Which one of the following is true?

A. val(j)=0=Θ(log n)

B. val(j)=0=Θ(√n)

C. val(j)=0=Θ(n)

D. val(j)=0=Θ(nlog n)

Explanation: C. If we observe, we find that the calculation of j involves n/2+n/4+n/8+…1=2(n–1)=Θ(n)

8. An element in an array X is called a leader if it is greater than all elements to the right of it in X. The best algorithm to find all leaders in an array

A. Solves it in linear time using a left to right pass of the array

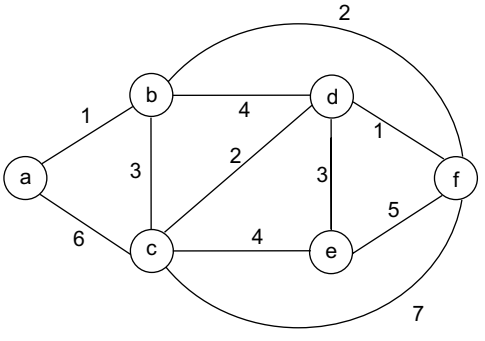
B. Solves in linear time using a right to left pass of the array

C. Solves using divide and conquer in time Θ(nlogn)

D. Solves it in time $\Theta(n^2)$

Explanation: C. Recall insertion sorting. We place each element such that it is more than all elements left (or right to it). In order to find all leaders, if we sort the array it is adequate. Thus, best sorting algorithms such as quick sort, heap sort, merge sort can be used whose time complexity is Θ(nlogn). Thus, option c is valid.

9. Consider the following graph.



Which one of the following cannot be the sequence of edges added, in that order to a minimum spanning tree using Krushkal's algorithm?

A. (a-b), (d-f), (b-f), (d-c), (d-e)

B. (a-b), (d-f), (d-c), (b-f) (d-e)

C. (d-f), (a-b), (d-c), (b-f), (d-e)

D. (d-f), (a-b), (b-f), (d-e), (d-c)

Explanation: We know that in Krushkal's algorithm edges are added in accordance with their increasing cost till all the n vertexes are connected through n-1 edges and MST is formed. In the given graph, edges and their weights:

Edges	Weights
(a-b)	1
(d-f)	1
(b-f)	2
(d-c)	2
(b-c)	3
(d-e)	3
(b-d)	4
(c-e)	4
(e-f)	5
(a-c)	6
(c-f)	7

As we have 6 vertexes, we may have to select only 5 smallest edges. Thus, option d is valid.

10. Let T be a depth first search tree in a undirected graph G. Vertices u and v are leaves of this tree T. The degree of both u and v in G are atleast 2. Which one of the following statements is true?

A. There must exist a vertex w adjacent to both v and u in G

B. There must exist a vertex w whose removal disconnects v and u in G

C. There must be a cycle in G containing u and v

D. There must exist a cycle in G containing u and all its neighbours in G

11. A set X can be represented by an array x[n] as follows:

x[i]=1 if i ϵ X

x[i]=0 otherwise

Consider the following algorithm in which x, y and z are Boolean arrays of size n

```
Algorithm zzz(x[], y[], z[]){
Int I;
For(i=0;i<n; ++i)
 z[i]=(x[i]^ ~y[i]) v (~x[i]^y[i])
}
```

The set z computed by the algorithm is

A. $(X \cup Y)$ B. $(X \cap Y)$

C. $(X-Y) \cap (Y-X)$ D. $(X-Y) \cup (Y-X)$

12. Consider the following recurrence

$T(n)=2T(|\sqrt{n}|)+1$

T(1)=1

Which of the following is true?

A. $T(n)=\Theta(\log \log n)$ B. $T(n)=\Theta(\log n)$
C. $T(n)=\Theta(\sqrt{n})$ D. $T(n)=\Theta(n)$

13. Given two arrays of numbers a1, … an and b1,…bn where each number is 0 or 1, the fastest algorithm to find the largest span(i,j) such that ai +ai+1+…. = aj = bi + bi + 1 +…. + bj, or report there is no such span

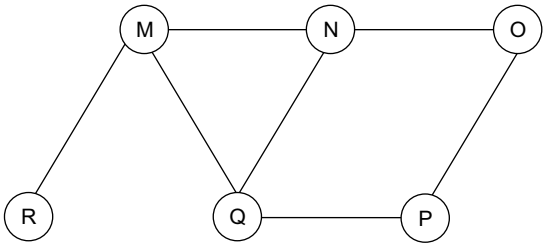
A. Takes $\Theta(3^n)$ and $\Omega(n^{2.5})$ time if hashing is permitted
B. Takes $\Theta(n^2)$ and $\Omega(n^{2.5})$ time in the key comparison model
C. Takes $\Theta(n)$ time and space
D. Takes $\Theta(\sqrt{n})$ time only if the sum of the 2n elements is an even number

14. The most efficient algorithm for finding the number of connected components in an undirected graph on n vertices and m edges has time complexity

(GATE CS 2008)

A. $\Theta(n)$ B. $\Theta(m)$ C. $\Theta(m+n)$ D. $\Theta(mn)$

15. The Breadth First Search algorithm has been implemented using the queue data structure. One possible order of visiting the nodes of the following graph is



A. MNOPQR B. NQMPOR
C. QMNPRO D. QMNPOR

16. Consider the following functions

$f(n)=2^n$

$g(n)=n!$

$h(n)=n^{\log n}$

Which of the following statements about the asymptotic behaviour of f(n), g(n), and h(n) is true?

A. $f(n) = O(g(n)); g(n) = O(h(n))$
B. $f(n) = \Omega(g(n)); g(n) = O(h(n))$
C. $g(n) = O(f(n)); h(n) = O(f(n))$
D. $h(n) = O(f(n)); g(n) = \Omega(f(n))$

17. The minimum number of comparisions required to determine if an integer appears more than n/2 times in a sorted array of n integers is

A. $\Theta(n)$ B. $\Theta(\log n)$
C. $\Theta(\log^* n)$ D. $\Theta(1)$

Explanation: A. We need atleast n/2 elements to be compared. Thus, minimum number of comparisons will be of order $\Theta(n)$.

18. A B-tree of order 4 is built from scratch by 10 sucessive insertions. What is the maximum number of node splitting operations that may take place?

A. 3 B. 4
C. 5 D. 6

19. G is a graph on n vertices and 2n-2 edges. The edges of G can be partitioned into two edge-disjoint spanning trees. Which of the following is not true for G?

A. For every subset of k vertices, the induced subgraph has at most 2k–2 edges
B. The minimum cut in G has at least two edges
C. There are two edge-disjoint paths between every pair of vertices
D. There are two vertex-disjoint paths between every pair of vertices

20. The recurrence relation capturing the optimal time of the Tower of Hanoi problem with n discs is

(GATE 2012)

A. $T(n) = 2T(n - 2) + 2$
B. $T(n) = 2T(n - 1) + n$
C. $T(n) = 2T(n/2) + 1$
D. $T(n) = 2T(n - 1) + 1$

21. Suppose a circular queue of capacity (n – 1) elements is implemented with an array of n elements. Assume that the insertion and deletion operation are carried out using REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT = 0. The conditions to detect queue full and queue empty are

(GATE 2012)

A. Full: (REAR+1) mod n == FRONT, empty: REAR == FRONT
B. Full: (REAR + 1) mod n == FRONT, empty: (FRONT+1) mod n == REAR
C. Full: REAR == FRONT, empty: (REAR+1) mod n == FRONT
D. Full: (FRONT+1) mod n == REAR, empty: REAR == FRONT

22. Four matricies M1, M2, M3 and M4 of dimensions pxq, qxr, rxs and sxt respectively can be multipied is several ways with different number of total scalar multiplications. For example, when multipied as ((M1 X M2) X (M3 X M4)), the total number of multiplications is pqr + rst + prt. When multipied as (((M1 X M2) X M3) X M4), the total number of scalar multiplications is pqr + prs + pst. If p = 10, q = 100, r = 20, s = 5 and t = 80, then the number of scalar multiplications needed is **(GATE 2011)**

A. 248000 B. 44000
C. 19000 D. 25000

23. Which of the given options provides the increasing order of asymptotic complexity of functions f1, f2, f3 and f4 ? **(GATE 2011)**

$f1(n) = 2^n$

$f2(n) = n^{3/2}$

$f3(n) = n\log^2 n$

$f4(n) = n^{\log_2 n}$

A. f3, f2, f4, f1 B. f3, f2, f1, f4

C. f2, f3, f1, f4 D. f2, f3, f4, f1

24. We are given a set of n distinct elements and an unlabeled binary tree with n nodes. In how many ways can we populate the tree with the given set so that it becomes a binary search tree? **(GATE 2011)**

A. 0 B. 1

C. !n D. $(1/(n+1)).2nC_n$

25. An algorithm to find the length of the longest monotonically increasing sequence of numbers in an array A[0:n-1] is given below. Let L_i denote the length of the longest monotonically increasing sequence starting at index i in the array.

Initialise L_{n-1}=1

for all i such that 0<=i<=n–2

Li=1+Li+1 if A[i]<A[i+1]

=1 otherwise,

Finally the length of the longest monotonically increasing sequence is Max (L_0 ,L_1 ,...,). Which of the following statements is TRUE?

A. The algorithm uses dynamic programming paradigm

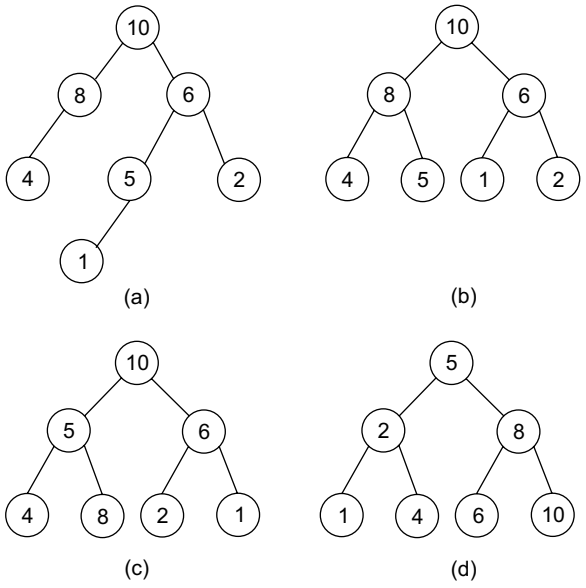
B. The algorithm has a linear complexity and uses branch and bound paradigm

C. The algorithm has a non-linear polynomial complexity and uses branch and bound paradigm

D. The algorithm uses divide and conquer paradigm.

Explanation: As it is using table Li.

26. A max-heap is a heap where the value of each parent is greater than or equal to the value of its children. Which of the following is a max-heap?



Explanation: B. Heap is a complete binary. That is, if nodes are assumed to be stored sequentially, all of them are supposed to occupy consecutive location. Rather, if we give 0 as the index of root and for all other nodes we assign 2*i+1 and 2*i+2 where is the index of their parent then all the numbers should be in sequence. This is violating for option A. Also, for max heap parent node value is supposed to be larger than its children. C and D are violating this rule. Thus, option B is the correct option.

Consider the following recursive C function that takes two arguments

```
unsigned int foo (unsigned int n, unsigned int
r ) {
if (n >0 )return (n%r) foo (n / r, r) );
else return 0;
}
```

27. What is the return value of the function foo when it is called as foo (513, 2)?

A. 9 B. 8

C. 5 D. 2

Explanation: D

1st call	2nd call	3rd call	4th call	5th call	6th call	7th call	8th call	9th call	10th call
513,2 1+foo (256,2)	256,2 return 0+ foo (128,2)	128,2 return 0+ foo (64,2)	64,2 return 0+ foo (32,2)	32,2 return 0+ foo (16,2)	16,2 return 0+ foo (8,2)	8,2 return 0+ foo (4,2)	4,2 return 0+ foo (2,2)	2,2 return 0+ foo (1,2)	1,2 return 1+ foo (0,2)
returns 2	returns 1	returns 1	returns 1	returns 1	returns 1	returns 1	returns 1	returns 1	

28. What is the return value of the function foo when it is called as foo (513, 2)?

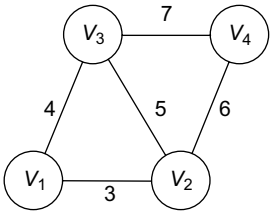
A. 345 B. 12

C. 5 D. 3

Explanation: The following table illustrates how the function call works.

1st call			
345,10 return 5+foo(34,10)	34,10 return 4+foo(3,10)	3,10 return 3+foo(0,10)	0,10 return 0
returns 12	returns 7	returns 3	

29. An undirected graph G(V,E) contains n (n >2) nodes named v1 , v2 ,....vn . Two nodes vi , vj are connected if and only if 0 < = |i – j| <= 2. Each edge (vi ,vj) is assigned a weight i + j. A sample graph with n = 4 is shown below



What will be the cost of minimum spanning tree (MST) of such a graph with n nodes?

A. 1/12(11n2–5n) B. n^2–n+1

C. 6n–11 D. 2n+1

Explanation: We know that MST of n nodes is formed by n-1 smallest edges. Also, MST doesn't contain any cycles. For instance, for the above example MST, edges that makes MST are: 1 + 2, 1 + 3, 2 + 4. If organise them, we can say (1 + 2 + 3 + 4) + (1 + 2). Similarly, if we consider a tree with 5 nodes then MST edges will be 1 + 2, 1 + 3, 2 + 4, 3 + 5. If organise them, we can say (1 + 2 + 3 + 4 + 5) + (1 + 2 + 3). That is, (1 + 2 +….+ n) + (1 + 2 +…n-2) = n^2–n+1

30. The length of the path from v5 to v6 in the MST of previous question with n = 10 is

A. 11 B. 25

C. 31 D. 41

Explanation: Possible edges which can be selected in the MST till v6 can be given as 1+2, 1+3, 2+4, 3+5, and 4+6 . If we draw a graph, we may find all these edges are in the path of v5 to v6. Their edge total is 31.

31. The degree sequence of a simple graph is the sequence of the degrees of the nodes in the graph in decreasing order. Which of the following sequences can not be the degree sequence of any graph? **(GATE 2010)**

I. 7, 6, 5, 4, 4, 3, 2, 1

II. 6, 6, 6, 6, 3, 3, 2, 2

III. 7, 6, 6, 4, 4, 3, 2, 2

IV. 8, 7, 7, 6, 4, 2, 1, 1

A. I and II B. III and IV

C. IV only D. II and IV

32. Consider a B+-tree in which the maximum number of keys in a node is 5. What is the minimum number of keys in any non-root node? **(GATE 2010)**

A. 1 B. 2

C. 3 D. 4

33. Let X be a problem that belongs to the class NP. Then which one of the following is true? **(GATE 2009)**

A. There is no polynomial time algorithm for X.

B. If X can be solved deterministically in polynomial time, then P = NP.

C. If X is NP-hard, then it is NP-complete.

D. X may be undecidable.

34. What is the number of swaps required to sort n elements using selection sort, in the worst case? **(GATE 2009)**

A. Θ(n) B. Θ(n log n)

C. Θ(n^2) D. Θ(n^2 log n)

35. Consider the polynomial p(x) = a0 + a1x + a2x^2 + a3x^3, where ai != 0, for all i. The minimum number of multiplications needed to evaluate p on an input x is: **(GATE 2006)**

A. 3 B. 4

C. 6 D. 9

Explanation: a. We need three only. Remember about Horners procedure. If we write the equation a0 + x(a1 + x(a2 +a3x))), we need three multiplications only.

36. To implement Dijkstra's shortest path algorithm on unweighted graphs so that it runs in linear time, the data structure to be used is: **(GATE 2006)**

A. Queue B. Stack

C. Heap D. B-Tree

37. A 3-ary max heap is like a binary max heap, but instead of 2 children, nodes have 3 children. A 3-ary heap can be represented by an array as follows: The root is stored in the first location, a[0], nodes in the next level, from left to right, is stored from a[1] to a[3]. The nodes from the second level of the tree from left to right are stored from a[4] location onward. An item x can be inserted into a 3-ary heap containing n items by placing x in the location a[n] and pushing it up the tree to satisfy the heap property. Which of the following is a valid sequence of elements in an array representing 3-ary max heap? **(GATE 2006)**

A. 1, 3, 5, 6, 8, 9
B. 9, 6, 3, 1, 8, 5
C. 9, 3, 6, 8, 5, 1
D. 9, 5, 6, 8, 3, 1

Explanation: D. Option a is not valid as the root node 1 is smaller than its children 3,5 and 6. Similarly, option is not valid as 8 is denoted as child of 6 where max heap property is missing. Similarly, option c is not valid as 5 is denoted as child of 3 where max heap property is missing.

38. Suppose the elements 7, 2, 10 and 4 are inserted, in that order, into the valid 3-ary max heap found in the above question, Which one of the following is the sequence of items in the array representing the resultant heap?

A. 10, 7, 9, 8, 3, 1, 5, 2, 6, 4
B. 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
C. 10, 9, 4, 5, 7, 6, 8, 2, 1, 3
D. 10, 8, 6, 9, 7, 2, 3, 4, 1, 5

39. The subset-sum problem is defined as follows. Given a set of n positive integers, S = $\{a_1, a_2, a_3, \ldots, a_n\}$ and positive integer W, is there a subset of S whose elements sum to W? A dynamic program for solving this problem uses a 2-dimensional Boolean array X, with n rows and W+1 columns. X[i, j],1 <= i <= n, 0 <= j <= W, is true if and only if there is a subset of {a1 ,a2 ,...,ai} whose elements sum to j. Which of the following is valid for 2 <= i <= n and a_i <= j <= W? **(GATE 2008)**

A. X[i, j] = X[i - 1, j] ν X[i, j –ai]
B. X[i, j] = X[i - 1, j] ν X[i - 1, j – ai]
C. X[i, j] = X[i - 1, j] ν X[i, j – ai]
D. X[i, j] = X[i - 1, j] ν X[i -1, j – ai]

40. Consider the process of inserting an element into a Max Heap, where the Max Heap is represented by an array. Suppose we perform a binary search on the path from the new leaf to the root to find the position for the newly inserted element, the number of comparisons performed is: **(GATE 2007)**

A. Θ(logn)
B. Θ(loglogn)
C. Θ(n)
D. Θ(nlogn)

41. Let w be the minimum weight among all edge weights in an undirected connected graph. Let e be a specific edge of weight w . Which of the following is false? **(GATE 2007)**

A. There is a minimum spanning tree containing e.
B. If e is not in a minimum spanning tree T, then in the cycle formed by adding e to T, all edges have the same weight.
C. Every minimum spanning tree has an edge of weight w .
D. e is present in every minimum spanning tree.

42. In an unweighted, undirected connected graph, the shortest path from a node S to every other node is computed most efficiently, in terms of time complexity by **(GATE 2007)**

A. Dijkstra's algorithm starting from S.
B. Warshall's algorithm
C. Performing a DFS starting from S.
D. Performing a BFS starting from S.

43. In the following C function, let n >= m.

```
int gcd(n,m)
{
if (n%m ==0) return m;
n = n%m;
return gcd(m,n);
}
```

How many recursive calls are made by this function? **(GATE 2007)**

A. Θ (logn)
B. Ω (n)
C. Θ (loglogn)
D. Θ (sqrt(n))

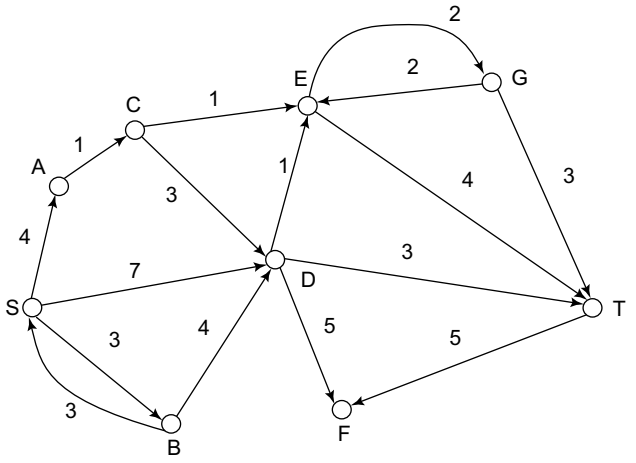
44. In a binary max heap containing n numbers, the smallest element can be found in time **(GATE 2006)**

A. O(n) B. O(logn)
C. O(loglogn) D. O(1)

45. Which one of the following in place sorting algorithms needs the minimum number of swaps? **(GATE CS 2006)**

A. Quick sort
B. Insertion sort
C. Selection sort
D. Heap sort

46. Consider the directed graph shown in the figure below. There are multiple shortest paths between vertices S and T. Which one will be reported by Dijkstra's shortest path algorithm? Assume that, in any iteration, the shortest path to a vertex v is updated only when a strictly shorter path to v is discovered.



A. SDT B. SBDT
C. SACDT D. SACET

47. The height of a tree is defined as the number of edges on the longest path in the tree. The function shown in the pseudocode below is invoked as height (root) to compute the height of a binary tree rooted at the tree pointer root.The appropriate expression for the two boxes B1 and B2 are **(GATE 2012)**

```
int height (treeptr n)
{if (n == NULL) return -1;
 if ( n –> left == NULL)
 if (n –> right == NULL) return 0;
 else return [B1] ;                    //Box 1
else (h1 = height (n __left);
 if (n ___ right == NULL) return (1 - h1)
 else h2 = height (n = right);
   return [B2] ;    //Box 2
   }
  }
}
```

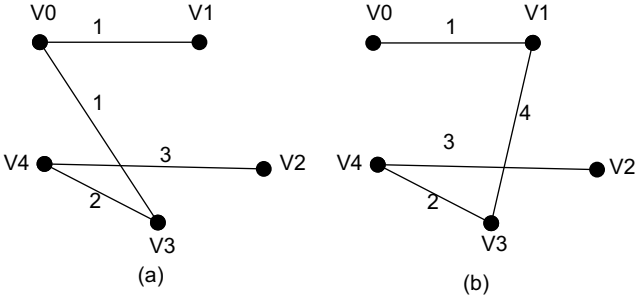
A. B1 : (1 + height(n->right)), B2 : (1 + max(h1,h2))
B. B1 : (height(n->right)), B2 : (1 + max(h1,h2))
C. B1 : height(n->right), B2 : max(h1,h2)
D. B1 : (1 + height(n->right)), B2 : max(h1,h2)

48. Consider a complete undirected graph with vertex set {0, 1, 2, 3, 4}. Entry Wij in the matrix W below is the weight of the edge {i, j}. What is the minimum possible weight of a spanning tree T in this graph such that vertex 0 is a leaf node in the tree T? **(GATE 2010)**

$$W = \begin{pmatrix} 0 & 1 & 8 & 1 & 4 \\ 1 & 0 & 12 & 4 & 9 \\ 8 & 12 & 0 & 7 & 3 \\ 1 & 4 & 7 & 0 & 2 \\ 4 & 9 & 3 & 2 & 0 \end{pmatrix}$$

A. 7 B. 8
C. 9 D. 10

Explanation: D. If we observe, minimum spanning tree is in figure a as shown below with weight of 7. However, in it 0 is not end node. Thus, the tree which is shown in the figure b is the minimum spanning tree with 0 as the end node and its weight is 10.



49. In the graph given in previous question , what is the minimum possible weight of a path P from vertex 1 to vertex 2 in this graph such that P contains at most 3 edges? **(GATE 2010)**

A. 7 B. 8
C. 9 D. 10

50. A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.Which one of the following choices gives a possible order in which the key values could have been inserted in the table? **(GATE 2010)**

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

A. 46, 42, 34, 52, 23, 33
B. 34, 42, 23, 52, 33, 46
C. 46, 34, 42, 23, 52, 33
D. 42, 46, 33, 23, 34, 52

51. The running time of an algorithm is represented by the following recurrence relation: if n <= 3 then T(n) = n else T(n) = T(n/3) + cn. Which one of the following represents the time complexity of the algorithm? **(GATE 2009)**

A. $\Theta(n)$ B. $\Theta(n \log n)$
C. $\Theta(n^2)$ D. $\Theta(n^2 \log n)$

Explanation: Applying masters theorem, we find the complexity is $\Theta(n)$

52. The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function h(k) = k mod 10 and linear probing. What is the resultant hash table?

(GATE 2009)

0	
1	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

(A)

0	
1	
2	12
3	13
4	
5	5
6	
7	
8	18
9	

(B)

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

(C)

0	
1	
2	12.2
3	13,3,23
4	
5	5,15
6	
7	
8	18
9	

(D)

A. A B. B
C. C D. D

53. Consider the following C program that attempts to locate an element x in an array Y[] using binary search. The program is erroneous.

```
1. f(int Y[10], int x) {
2. int i, j, k;
3. i = 0; j = 9;
4. do {
5. k = (i + j) /2;
6. if( Y[k] < x) i = k; else j = k;
7. } while(Y[k] != x && i < j);
8. if(Y[k] == x) printf ("x is in the array ") ;
9. else printf (" x is not in the array " ) ;
10. }
```

On which of the following contents of Y and x does the program fail? **(GATE 2008)**

A. Y is [1 2 3 4 5 6 7 8 9 10] and x < 10
B. Y is [1 3 5 7 9 11 13 15 17 19] and x < 1
C. Y is [2 2 2 2 2 2 2 2 2 2] and x > 2
D. Y is [2 4 6 8 10 12 14 16 18 20] and 2 < x < 20 and x is even

Explanation: C. Program enters into infinite loop

54. Consider the following C program segment where CellNode represents a node in a binary tree:

```
struct CellNode
{
struct CellNOde *leftChild;
int element;
struct CellNode *rightChild;
};
int GetValue(struct CellNode *ptr)
{
int value = 0;
if (ptr != NULL)
{
if ((ptr->leftChild == NULL) &&
(ptr->rightChild == NULL))
value = 1;
else
value = value + GetValue(ptr->leftChild) + Get-
value (ptr->rightChild);
}
return value;
}
```

The value returned by GetValue when a pointer to the root of a binary tree is passed as its argument is:

(GATE 2007)

A. the number of nodes in the tree
B. the number of internal nodes in the tree
C. the number of leaf nodes in the tree
D. the height of the tree

55. An array of n numbers is given, where n is an even number. The maximum as well as the minimum of these n numbers needs to be determined. Which of the following is true about the number of comparisons needed? **(GATE 2007)**

A. At least 2n – c comparisons, for some constant c, are needed.
B. At most 1.5n – 2 comparisons are needed.
C. At least nLog2n comparisons are needed.
D. None of the above.

56. Consider a hash table of size seven, with starting index zero, and a hash function (3x + 4)mod7. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using closed hashing? Note that '_' denotes an empty location in the table. **(GATE 2007)**

A. 8, _, _, _, _, _, 10
B. 1, 8, 10, _, _, _, 3
C. 1, _, _, _, _, _,3
D. 1, 10, 8, _, _, _, 3

57. A complete n-ary tree is a tree in which each node has n children or no children. Let I be the number of internal nodes and L be the number of leaves in a complete n-ary tree. If L = 41, and I = 10, what is the value of n? **(GATE 2007)**

A. 3 B. 4
C. 5 D. 6

58. What is the time complexity of the following recursive function: **(GATE 2007)**

```
int DoSomething (int n)
{
if (n <= 2)
return 1;
   else
return (DoSomething (floor(sqrt(n))) + n);
}
```

A. Θ(n) B. Θ(nlogn)
C. Θ(logn) D. Θ(loglogn)

59. A scheme for storing binary trees in an array X is as follows. Indexing of X starts at 1 instead of 0. the root is stored at X[1]. For a node stored at X[i], the left child, if any, is stored in X[2i] and the right child, if any, in X[2i+1]. To be able to store any binary tree on n vertices the minimum size of X should be. **(GATE CS 2006)**

A. log2n B. n
C. 2n + 1 D. $2^n - 1$

60. An element in an array X is called a leader if it is greater than all elements to the right of it in X. The best algorithm to find all leaders in an array **(GATE CS 2006)**

A. Solves it in linear time using a left to right pass of the array
B. Solves it in linear time using a right to left pass of the array
C. Solves it using divide and conquer in time Θ(nlogn)
D. Solves it in time $\Theta(n^2)$

61. Consider the following C function

```
int f1(int n)
{
if(n == 0 || n == 1)
return n;
else
return (2*f1(n-1) + 3*f1(n-2));
}
int f2(int n)
{
int i;
int X[N], Y[N], Z[N] ;
X[0] = Y[0] = Z[0] = 0;
X[1] = 1; Y[1] = 2; Z[1] = 3;
for(i = 2; i <= n; i++)
{
X[i] = Y[i-1] + Z[i-2];
Y[i] = 2*X[i];
Z[i] = 3*X[i];
}
return X[n] ;
}
```

The running time of f1(n) and f2(n) **(GATE 2008)**

A. theta(n) and theta (n)
B. theta(2^n) and theta(n)
C. theta(n) and theta(2^n)
D. theta(2^n) and theta(2^n)

62. Consider the following C program segment:

```
char p[20];
char *s = "string";
int length = strlen(s);
int i;
for (i = 0; i < length; i++)
p[i] = s[length-i];
printf("%s",p);
```

The output of the program is **(GATE CS 2004)**

A. gnirts
B. gnirt
C. string
D. No output is printed

63. Choose the correct option to fill ?1 and ?2 so that the program below prints an input string in reverse order. Assume that the input string is terminated by a newline character. **(GATE 2008)**

```
void reverse(void)
{
int c;
if (?1) reverse() ;
?2
}
main()
{
printf ("Enter Text") ;
printf ("\n") ;
```

```
reverse();
printf("\n") ;
}
```

A. ?1 is (getchar() != '\n') ?2 is getchar(c);

B. ?1 is (c = getchar()) != '\n') ?2 is getchar(c);

C. ?1 is (c != '\n') ?2 is putchar(c);

D. ?1 is ((c = getchar()) != '\n') ?2 is putchar(c);

64. In the C language **(GATE CS 2002)**

A. At most one activation record exists between the current activation record and the activation record for the main

B. The number of activation records between the current activation record and the activation record for the main depends on the actual function calling sequence.

C. The visibility of global variables depends on the actual function calling sequence.

D. Recursion requires the activation record for the recursive function to be saved on a different stack before the recursive function can be called.

65. What does the following program print? **(GATE 2011)**

```
#include
void f(int *p, int *q)
{
p = q;
*p = 2;
}
int i = 0, j = 1;
int main()
{
f(&i, &j);
printf("%d %d \n", i, j);
getchar();
return 0;
}
```

A. 2 2 B. 2 1

C. 0 1 D. 0 2

66. What is the value printed by the following C program? **(GATE 2010)**

```
#include
int f(int *a, int n)
          {
if(n <= 0) return 0;
else if(*a % 2 == 0) return *a + f(a+1, n-1);
else return *a - f(a+1, n-1);
}
int main()
{
int a[] = {12, 7, 13, 4, 11, 6};
printf("%d", f(a, 6));
getchar();
return 0;
}
```

A. –9 B. 5

C. 15 D. 19

Explanation: See the following snapshop of the above program. Assume array a's address is 2000. Also, assuming integer takes 2 bytes on our machine.

As first element is 7 (odd) it Calls 7-f (2004, 4)					
	As first element is odd (13) it calls 13-f(2006,3)				
		As first element is even (4) it calls 4+f 2008,2)			
			As first element is odd(11) it calls 11-f (2010,1)		
				As first element is even (6) it calls 6+f (2012,0)	
					Return 0
Return 7-3	Return 13-9	Returns 4+9	Returns 11-6	Returns 6+0	

67. Consider the following program fragment for reversing the digits in a given integer to obtain a new integer. Let n = D1D2…Dm

```
int n, rev;
```

```
rev = 0;
while (n > 0)
{
rev = rev*10 + n%10;
n = n/10;
}
```

The loop invariant condition at the end of the ith iteration is: **(GATE CS 2004)**

A. n = D1D2….Dm-i and rev = DmDm-1…Dm-i+1

B. n = Dm-i+1…Dm-1Dm and rev = Dm-1….D2D1

C. n =! rev

D. n = D1D2….Dm and rev = DmDm-1…D2D1

68. Consider the following C program int a, b, c = 0;

```
void prtFun (void);
int main ()
{
static int a = 1; /* line 1 */
prtFun();
a += 1;
prtFun();
printf ( "\n %d %d ", a, b) ;
}
void prtFun (void)
{
static int a = 2; /* line 2 */
int b = 1;
a += ++b;
printf ("\n%d %d", a, b);
}
```

A. 3 1 4 1 4 2 B. 4 2 6 1 6 1

C. 4 2 6 2 2 0 D. 3 1 5 2 5 2

Answer: C. A local variable (even if it is static) will mask the scope of a global variable with the same name. Thus, in the function prtFun a refers to the a of that function only. Because of the same reason, b in the function prtFun refers to the one it itself, not the global b.

69. Consider the C program shown below.

```
# include <stdio.>
# define print(x) printf ("%d", x)
int x;
void Q(int z)
{
z += x;
print(z);
}
void P(int *y)
{
int x = *y+2;
Q(x);
*y = x-1;
print(x);
}
main(void)
{ x=5;
P(&x);
print(x);
getchar();
}
```

The output of this program is **(GATE CS 2003)**

A. 1276 B. 22 12 11

C. 14 6 6 D. 766

Explanation: See the trace of the program execution

	Global=x	Of Function P y = address of global x x= is loacl	Of function Q
x=5	Global x becomes 5		
P(&x)		Y becomes address of global x	
Int x =*y+2		Local x becomes 7	
Q(x)			z becomes 7
z+=x;			Adds global variable x value to z. That z becomes 12. Next statement prints 12 and the function call returns to function P
*y=x-1	Global variable becomes 6		
Print(x)		Local x value 7 will be printed	
Print (x)	Prints global variable x value. Thus, finally we get 1276		

70. Consider this C code to swap two integers and these five statements: the code

```
void swap(int *px, int *py)
{
*px = *px - *py;
*py = *px + *py;
```

```
*px = *py - *px;
}
```

S1: will generate a compilation error

S2: may generate a segmentation fault at runtime depending on the arguments passed

S3: correctly implements the swap procedure for all input pointers referring to integers stored in memory locations accessible to the process

S4: implements the swap procedure correctly for some but not all valid input pointers.

S5: may add or subtract integers and pointers

(GATE 2006)

A. S1 B. S2 and S3

C. S2 and S4 D. S2 and S5

71. Consider the following three C functions :,

```
[PI] int * g (void)
{
int x = 10;
return (&x);
}
[P2] int * g (void)
{
int * px;
*px = 10;
return px;
}
[P3] int *g (void)
{
int *px;
px = (int *) malloc (sizeof(int));
*px = 10;
return px;
}
```

Which of the above three functions are likely to cause problems with pointers? **(GATE 2001)**

A. Only P3

B. Only P1 and P3

C. Only P1 and P2

D. P1, P2 and P3

Explanation: P1 is returning the address of stack (or scratch) variable x. Where as P2 is trying to store 10 in some location which is not yet allocated.

72. Consider the following C-program fragment in which i, j and n are integer variables.

```
for (i = n, j = 0; i >0; i /= 2, j += i);
```

Let val(j) denote the value stored in the variable j after termination of the for loop. Which one of the following is true? **(GATE 2006)**

A. val(j) = theta(logn)

B. vaI(j) = theta (sqrt(n))

C. val(j) = theta(n)

D. val(j) = theta(nlogn)

73. What is printed by the following C program?

```
int f(int x, int *py, int **ppz)
{
int y, z;
**ppz += 1;
z = **ppz;
*py += 2;
y = *py;
x += 3;
return x + y + z;
}
void main()
{
int c, *b, **a;
c = 4;
b = &c;
a = &b;
printf( "%d'', f(c, b, a));
getchar();
}
```

08 **(GATE 2008)**

A. 18 B. 19

C. 21 D. 22

74. Consider the following C-function in which a[n] and b[m] are two sorted integer arrays and c[n + m] be another integer array.

```
void xyz(int a[], int b [], int c[])
{
int i, j, k;
i = j = k = 0;
while ((i<n)&& (j<m))
 if (a[i] < b[j]) c[k++] = a[i++];
else c[k++] = b[j++];
}
```

Which of the following condition(s) hold(s) after the termination of the while loop?

(i) j < m, k = n+j-1, and a[n-1] < b[j] if i = n

(ii) i < n, k = m+i-1, and b[m-1]<=a[i] if j=m

(GATE 2008)

A. only (i)

B. only (ii)

C. either (i) or (ii) but not both

D. neither (i) nor (ii)

75. Assume the following C variable declaration

```
int *A [10], B[10][10];
```

Of the following expressions

I A[2]

II A[2][3]

III B[1]

IV B[2][3]

which will not give compile-time errors if used as left hand sides of assignment statements in a C program?

(GATE CS 2003)

A. I, II, and IV only B. II, III, and IV only

C. II and IV only D. IV only

76. The value of j at the end of the execution of the following C program. **(GATE CS 2000)**

```
int incr (int i)
{
static int count = 0;
count = count + i;
return (count);
}
main ()
{
int i,j;
for (i = 0; i <=4; i++)
j = incr(i);
}
```

A. 10 B. 4

C. 6 D. 7

77. Consider the following C function:

```
int f(int n)
{
static int i = 1;
if (n >= 5)
return n;
n = n+i;
i++;
return f(n);
}
```

The value returned by f(1) is **(GATE CS 2004)**

A. 5 B. 6

C. 7 D. 8

Explanation: 7. Static variable value is shared by recursive function calls also.

78. What does the following fragment of C-program print?

```
char c[] = "GATE2011";
char *p =c;
printf("%s", p + p[3] - p[1]) ;
```

(GATE2011)

A. GATE2011 B. E2011

C. 2011 D. 011

79. Consider the following C function

```
void swap (int a, int b)
{ int temp;
temp = a;
a = b;
b = temp;
}
```

In order to exchange the values of two variables x and y. **(GATE CS 2004)**

A. call swap (x, y)

B. call swap (&x, &y)

C. swap (x,y) cannot be used as it does not return any value

D. swap (x,y) cannot be used as the parameters are passed by value

80. Consider the following declaration of a 'two-dimensional array in C: char a[100][100]; Assuming that the main memory is byte-addressable and that the array is stored starting from memory address 0, the address of a[40][50] is **(GATE CS 2002)**

a. 4040 B. 4050

C. 5040 D. 5050

81. Consider the following C program

```
main()
{
int x, y, m, n;
scanf ("%d %d", &x, &y);
/* x > 0 and y > 0 */
m = x; n = y;
while (m != n)
{
if(m>n)
m = m - n;
else
n = n - m;
}
printf("%d", n);
}
```

The program computes **(GATE CS 2004)**

A. x + y using repeated subtraction

B. x mod y using repeated subtraction

C. the greatest common divisor of x and y

D. the least common multiple of x and y

82. Consider the following recursive C function that takes two arguments unsigned int foo(unsigned int n, unsigned int r) { if (n > 0) return (n%r + foo (n/r, r)); else return 0; } **GATE 2011**

A. 9 B. 8
C. 5 D. 2

83. Consider the following C declaration struct { short s [5] union { float y; long z; }u; } t; Assume that objects of the type short, float and long occupy 2 bytes, 4 bytes and 8 bytes, respectively. The memory requirement for variable t, ignoring alignment considerations, is **(GATE CS 2000)**

A. 22 bytes B. 14 bytes
C. 18 bytes D. 10 bytes

84. What will be the output of the following C program segment?

```
char inchar = 'A';
switch (inchar)
{
case 'A' :
printf ("choice A\n") ;
case 'B' :
printf ("choice B ") ;
case 'C' :
case 'D' :
case 'E' :
default:
printf ("No Choice") ; }
```

A. No choice
B. Choice A
C. Choice A Choice B No choice
D. Program gives no output as it is erroneous

85. Which one of the following is the tightest upper bound that represents the number of swaps required to swap n numbers using selection sorting? **(GATE 2013)**

A. O(logn) B. O(n)
C. O(nlogn) D. $O(n^2)$

Explanation: B. Each time we find index of the largest element and swap it with first or last element. As we have n elements, we need to do this for n-1 times. Thus, number of swaps will be O(n).

86. Which one of the following is the tightest upper bound that represents time complexity of inserting an object into a binary search tree of n nodes? **(GATE 2013)**

A. O(logn) b. O(n)
c. O(n) D. O(nlogn)

Explanation: C. Binary search will be having worst case height of n if it is in degenerate form. In this situation, we need n nodes to be visited in the worst case for inserting a new node. Thus, time complexity of insertion become O(n).

87. Which of the following statements are TRUE? **(GATE 2013)**

1. The problem of determining whether there exists a cycle in undirected graph is P
2. The problem of determining whether there exists a cycle in undirected graph is NP
3. If a problem A is NP-complete, there exists a non-deterministic polynomial time algorithm to solve A

A. 1,2 and 3 B. 1 and 2 only
C. 2 and 3 only D. 1 and 3 only

88. What is the time complexity of Bellman Ford single source shortest path algorithm on a complete graph with n vertices? **(GATE 2013)**

A. $\Theta(n^2)$ B. $\Theta(n^2 \log n)$
C. $\Theta(n^3)$ D. $\Theta(n^3 \log n)$

89. Consider an undirected random graph of eight vertices. The probability that there is an edge between a pair of vertices is ½. What is the expected number of unordered cycles or length three? **(GATE 2013)**

A. 1/8 B. 1
C. 7 D. 8

90. Which of the following statements is/are true for undirected graphs? **(GATE 2013)**

P:Number odd degree vertices is even

Q:Sum of degrees of all vertices is even

A. P only B. Q only
C. Both P and Q D. Neither P nor Q

91. The line graph L(G) of a simple graph G is defined as follows:

There is exactly one vertex v(e) in L(G) for each edge e in G

For any two edges e and e' in G, L(G) has an edge between v(e) and v(e') if and only if e and e′ are incident with the same vertex in G

Which of the following statements is/are true? **(GATE 2013)**

(P) The line graph is a cycle
(Q) The line graph of a clique is clique
(R) The line graph of a planar graph is planar
(S) The line graph of a tree is a tree

A. P only B. P and R only
C. R only D. P, Q and S only

92. The number of elements that can be sorted in Θ(logn) time using heap sort is **(GATE 2013)**

A. Θ(1) B. Θ(√logn)

C. Θ(logn/loglogn) D. Θ(logn)

93. Consider the following function

```
int unknown(int n)
{
int i,j,k=0;
for(i=n/2;i<=n;i++)
for(j=2;j<=n;j=j*2)
k=k+n/2;
return (k);
}
```

The return value of the function is **(GATE 2013)**

A. $\Theta(n^2)$ B. $\Theta(n^2 \log n)$

C. $\Theta(n^3)$ D. $\Theta(n^3 \log n)$

94. Preorder traversal sequence of a binary tree is 30 20 10 15 25 39 35 42. Which one of the following is the postorder traversal sequence of the same tree? **(GATE 2013)**

A. 10 20 15 23 25 35 42 39 30

B. 15 10 25 23 20 42 35 39 30

C. 15 20 10 23 25 42 35 39 30

D. 15 10 23 25 20 35 42 39 30

95. What is worst case time complexity of a sequence of n queue operations on an initially empty queue? **(GATE 2013)**

A. Θ(n) B. Θ(n+k)

C. Θ(nk) D. $\Theta(n^2)$

96. What is the return value if f(p, p), if the p value is initialized to 5 before the call? Note that the first parameter is passed by reference whereas second parameter is passed by value.

```
int f(int &x, int c){
c=c-1;
if(c==0) return 1;
x=x+1;
return f(x,c)*x;
}
```

A. 3024 B. 6561

C. 55440 D. 161051

Explanation: Function call trace.

1st call	2nd call	3rd call	4th call	5th call
f(5,5)				
Initially x is 5 and c is 5 c becomes 4 while x becomes 6 returns f(6,4)*6	x=6 c=4 Changes to x=7 c=3 returns f(7,3)*7	x=7 c=3 changes to x=8 c=2 returns f(8,2)*8	x=8 c=2 changes to x=9 c=1 returns f(9,1)*9	x=9 c=1 changes x=10 c=0 As c is 0 it returns 1
Returns 6*7*8*9	Returns 7*8*9	returns 8*9	returns 9	

ANSWER KEY

1. A	**2.** A	**3.** B	**4.** C
5. D	**6.** C	**7.** C	**8.** C
9. D	**10.** D	**11.** D	**12.** A
13. C	**14.** C	**15.** C	**16.** D
17. A	**18.** A	**19.** D	**20.** D
21. A	**22.** C	**23.** B	**24.** D
25. A	**26.** B	**27.** D	**28.** B
29. B	**30.** C	**31.** D	**32.** C
33. C	**34.** A	**35.** A	**36.** C
37. D	**38.** A	**39.** B	**40.** A
41. D	**42.** D	**43.** A	**44.** A
45. C	**46.** D	**47.** A	**48.** D
49. B	**50.** B	**51.** A	**52.** C
53. C	**54.** C	**55.** D	**56.** B
57. C	**58.** D	**59.** D	**60.** C
61. B	**62.** D	**63.** D	**64.** B
65. D	**66.** C	**67.** A	**68.** C
69. A	**70.** B	**71.** C	**72.** C
73. B	**74.** B	**75.** A	**76.** D
77. C	**78.** C	**79.** D	**80.** C
81. C	**82.** D	**83.** C	**84.** C
85. B	**86.** C	**87.** A	**88.** C
89. D	**90.** C	**91.** B	**92.** A
93. B	**94.** D	**95.** A	**96.** B

CHAPTER THREE

Theory of Computation

3.1 Introduction to Theory of Computation

Theory of Computation is a course of abstractions about what we can compute with the help of Turing Machine, the abstract model of our modern computer, invented by Alan Turing. An important abstract model, Finite state machine is used in string searching algorithms, compiler design, control unit design in computer architecture, and many other modeling applications. Context free grammars and their restricted forms are the basis of compilers and parsing. NP-Complete theory helps us distinguish the tractable from the intractable.

3.1.1 Strings, Languages

A language L is a set of strings over a finite alphabet A (also represented as Σ in some books).

■ **Example** Given the alphabet A = {a, b, c}, the following sets of strings are some of the languages over A: {aa, bb, aaaa}, {bb}, {a, b, c}. The following sets of strings are *not* in a language over A: {aaabf}, {0001, 1010}, {a5b, asas}.

Definition

- The special string of length 0 is called λ. Some languages contain λ and some do not.
- A language L is **infinite** if it contains an infinite number of strings; a language is **finite** if it contains a finite number of strings.
- A* is the language of all strings over the alphabet A.

Note

- Four simple examples of languages over an alphabet A are the sets ∅, {Λ}, *A*, and *A**. For example, if *A*={*a*} then these four simple languages over *A* are ∅, {Λ}, {*a*}, and {L, *a, aa, aaa,* …}.
- A string in a language is often called a well-formed formula or **wff** for short because the definition of the language usually allows only certain well-formed strings.
- *A** is the biggest possible language over A, and every other language over A is a subset of *A**.
- The difference between A (an alphabet), and A*, a language. The first is a set of symbols that we can use to make strings. The second is a set of strings over that set of symbols. Like any set, A can be empty. If A is not empty, A* is infinite.

The process of combining two strings is called concatenation. The string x = acc can be concatenated with the string y = cc to form the string acccc. A shorthand way of indicating this concatenation is cat(acc, cc) or cat(x,y). Note that the concatenation operation is ordered such that cat(x,y) ≠ cat(y,x). The latter forms the string, ccacc.

Definition: cat(x, λ) = x.

Note

The length of any string x concatenated with λ is same as the length of the string x.

Languages say, L and M are collection of strings; by combining strings of L and M (L.M) we can form new language that represents that each string x in L is concatenated with each string y in M. Do remember that in general, L•M ≠ M•L.

■ **Example 1.** Let L = {aa, bb} and let M = {cc, dd}. Then, the new language L•M = {aacc, aadd, bbcc, bbdd}, while the new language M•L = {ccaa, ccbb, ddaa, ddbb}. Note that all of L, M, L•M, and M•L are finite.

■ **Example 2.** Let L = {aa, bb} and M = {bb}. Then the new language L•M = {aabb, bbbb}, while the new language M•L = {bbaa, bbbb}. Note that all of L, M, L•M, and M•L are finite.

■ **Example 3.** Let L = {λ} and let M = {aa}. Note that L and M each contain one string. In this case, the language L•M = {λaa} and the language M•L = {aaλ}. However, from a definition above, λaa = aa =aaλ, so in this atypical case, L•M and M•L contain the same string(s). That is, L•M = M•L *in this atypical case.* Again, note that all of L, M, L•M, and M•L are finite.

The strings in a language L can be combined with all the strings in L. That is, it is possible to have L•L, generally forming a new language. This language L•L can be combined with all the strings in L, forming the language L•(L•L). The same, we write as $L^1 = L$, $L^2 = L•L$, $L^3 = L•L•L$.

Definition

- $L^0 = \{\lambda\}$.
- $L^n = L•L^{n-1}$, for n > 0.
- The *closure* of L is $L^* = \{L^0 \cup L^1 \cup L^2 \cup \ldots\}$. Note that typically L^* is a new language, and that typically L^* is infinite.
- The *positive closure* of L is $L^+ = \{L^1 \cup L^2 \cup \ldots\}$. Note that typically L^+ is a new language, and that typically L^+ is infinite.
- **Note**: $L^* = L^+ \cup \{\lambda\}$.

Consider the difference between L^*, L^+ when L is not empty and λ is not in L. Then, both L^* and L^+ are infinite and $L^+ = L^* - \{\lambda\}$.

However, when L is not empty but λ is in L, then, while L^* and L^+ are still infinite, it is not the case that $L^+ = L^* - \{\lambda\}$. This is because, in this case, λ is in L^1, and so by definition must be in L^+. Note that in this case, $L^+ = L^*$.

Definition: The properties of closure are

$$\{\lambda\}^* = \varnothing^* = \{\lambda\}$$

$$L^* = L^*•L^* = (L^*)^*$$

λ is in L if and only if $L^+ = L^*$

$$(L^*•M^*)^* = (L^* \cup M^*)^* = (L \cup M)^*$$

$$L•(M•L)^* = (L•M)^* • L$$

Kleene's Closure: The union of all powers of a set is called the Kleene closure.

■ **Example 1.** Show three strings in the set X = {a, b}{a, b}* {ab}

■ **Answer:** aaab, aababababab, and abbbbbab

■ **Example 2.** Show three strings not in the set X={a, b}{a, b}* {ab}

■ **Answer:** abb, babb, and aababa

3.1.2 Grammars

A language is described by a grammar. Rather, we can say that a grammar generates the strings of a language.

Definition

- A grammar G consists of the 4-tuple, G = <N, T, S, P>, where N is a finite set of non-terminals, T is a finite set of Terminals ($N \cap T = \varnothing$), S (an element of N) is the unique Start symbol, and P is a finite set of productions (or re-write rules). Terminal symbols are the ones from which no further generations (productions) takes place; whereas from non-terminals productions are possible.
- The elements of P are of the form $\alpha \rightarrow \beta$ (read: "alpha can be re-written as beta"), and where α, β are strings over A = N U T, and $\alpha \neq \lambda$.

Note

By convention used here, unless otherwise stated, the elements of N will be represented by uppercase symbols, and no uppercase symbols will be elements of T. Also, by convention, we list the strings in a language L in order of increasing length.

Each string in L(G) is the result of sequential application of the production rules in P. The particular sequential application of production rules in P that results in some string s in L(G) is called the derivation of s. Each step in a derivation is indicated by the "double arrow" symbol, $\Rightarrow$, such that Xa $\Rightarrow$ Cza means that the non terminal X in the string Xa (a string over the alphabet A) has been re-written as the string Cz , resulting in the string Cza. Do not confuse the single arrow symbol used to denote a production, with the double arrow symbol used to denote a derivation.

The notation $S \Rightarrow^3 aaa$ indicates that the string aaa can be derived through three applications of some productions from the start symbol S. The notation $S \Rightarrow^* aaa$ indicates that the string aaa can be derived through 0 or more applications of some productions from the start symbol S. The notation $S \Rightarrow^+$ aaa indicates that the string aaa can be derived through 1 or more applications of some productions from the start symbol S.

Note

A string made up of terminals and/or nonterminals is called a *sentential form*. Now we can formalise the idea of a derivation. If x and y are sentential forms and $\alpha \rightarrow \beta$ is a production, then the replacement of α by β in xαy is called a *derivation*, and we denote it by writing $x\alpha y \Rightarrow x\,\beta\,y$.

Definition: For the strings s, L(G) = {s| s is in T* and $S \Rightarrow^+ s$}

■ **Example** G = <(N = {S, A, B}, (T = {a,b}), S, (P = {S→aA, A→a})>. Note that not all the elements of N or T are used in P. The language described by this grammar is very simple: L(G) = {aa}. The derivation of this string is: S ⇒aA⇒aa. It would be appropriate to write: $S \Rightarrow^2$ aa. With a little thought, we can see that the following are also true: $S \Rightarrow^*$aaa, and $S \Rightarrow^+$ aaa. By way of contrast, this is not true: $S \Rightarrow^4$ aa as we do not need to use productions for four times to generate aa.

Note

- If the language is finite, then a grammar can consist of all productions of the form $S \rightarrow w$ for each string w in the language. For example, the language *{a, ab}* can be described by the grammar $S \rightarrow a \mid ab$. If the language is infinite, then some production or sequence of productions must be used repeatedly to construct the derivations. That is, there is no bound on the length of strings in an infinite language. That is, there are no bounds on the number of derivation steps used to derive the strings.
- If the grammar has n productions, then any derivation consisting of n + 1 steps must use some production twice.

■ **Example** For the grammar G, P = {S → aaB, B → b| SB}. First, we list the elements in N that are used in the derivation of some string in L(G). These are, N = {S, B}. Next, we list the elements in T that are used in the derivation of some string in L(G). These are, T={a,b}. Now, let's derive a few strings in L(G) (remember, eventually we will list these strings in order of increasing length).

$S \Rightarrow aaB \Rightarrow aab$

$S \Rightarrow aaB \Rightarrow aaSB \Rightarrow aaaaBB \Rightarrow aaaabB \Rightarrow aaaabb$

$S \Rightarrow aaB \Rightarrow aaSB \Rightarrow aaaaBB \Rightarrow aaaaSBB \Rightarrow aaaaaaBBB \Rightarrow aaaaaabBB \Rightarrow aaaaaabbB \Rightarrow aaaaaabbb.$

So, L(G) = {aab, aaaabb, aaaaaabbb, …}. Rather, we can describe L(G) of this production rules as the strings consists of n pairs of a's followed by n b's.

A production is called ***recursive*** if its left side occurs on its right side. For example, the production $S \rightarrow aS$ is recursive. A production $A \rightarrow a$ is ***indirectly recursive*** if A derives a sentential form that contains A. For example, suppose we have the following grammar:

$S \rightarrow b \mid aA$

$A \rightarrow c \mid bS.$

The productions $S \rightarrow aA$ and $A \rightarrow bS$ are both indirectly recursive because of the following derivations:

$S \Rightarrow aA \Rightarrow abS,$

$A \Rightarrow bS \Rightarrow baA.$

A grammar is *recursive* if it contains either a recursive production or an indirectly recursive production. We can make the following general statement about grammars for infinite languages:

Note

A grammar for an infinite language must be recursive.

In the example above, L(G) is an infinite language. ***For a language to be infinite, it must have at least one recursive production used in the derivation of some terminal string***. Recall that productions are of the form $\alpha \rightarrow \beta$, where α, β are strings over A = N U T, and $\alpha \neq \lambda$. A recursive production is defined as a production that directly or indirectly results in the derivation of some string β such that $\alpha \Rightarrow^{+} \beta$ and some non-terminal R is in both α and β. Thus, when a grammar has a recursive production used to derive some string, it effectively allows a "looping" action so that an infinite number of additional strings can be derived.

Also, in the above example, during the derivation of the second and third strings, we came upon the step:

$S \Rightarrow aaB \Rightarrow aaSB$. Now there is a choice; should the S be rewritten first, or should the B be rewritten first. If the non-terminal farthest to the right of the string is always selected to be re-written, the derivation is called a **right-most** derivation. If the non-terminal to the left of the string is always selected to be re-written first, the derivation is called a **left-most** derivation. In the example above we performed a left-most derivation. Here is the right most derivation of the second string from the example:

$S \Rightarrow aaB \Rightarrow aaSB \Rightarrow aaSb \Rightarrow aaaaBb \Rightarrow aaaaaabb.$

■ **Example 1.** Is λ in a language?

■ **Answer:** λ is in language if and only if there is some derivation such that

$S \Rightarrow^{+} \lambda.$

■ **Example 2.** $P = \{S \rightarrow aB \mid \lambda, B \rightarrow a\}$.

■ **Answer:** This language consists of a set of exactly two strings, aa and λ.

■ **Example 3.** Compare the language in the above example to the language described by the grammar with the following productions: $P = \{S \rightarrow aB, B \rightarrow a \mid \lambda\}$.

■ **Answer:** This latter language will also consist of a set of exactly two strings, aa and $a\lambda$. But recall that $a\lambda = a$. So this language does not contain λ, even though λ occurs in the derivation of some string.

■ **Example 4.** Consider the language with the set of productions: $P = \{S \rightarrow a \mid \lambda \mid S\lambda\}$. At first glance this seems to be an infinite language that contains λ. But is it infinite?

■ **Answer:** Let us derive some strings.

$S \Rightarrow \lambda$

$S \Rightarrow a$

$S \Rightarrow S\lambda \Rightarrow \lambda\lambda$

$S \Rightarrow S\lambda \Rightarrow a\lambda$

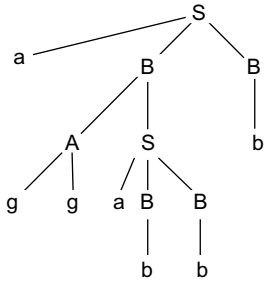
$S \Rightarrow S\lambda \Rightarrow S\lambda\lambda \Rightarrow \lambda\lambda\lambda$

$S \Rightarrow S\lambda \Rightarrow S\lambda\lambda \Rightarrow a\lambda\lambda.$

In this example we see that we are only deriving two strings, λ and a. This grammar produces an infinite number of derivations for these two strings, but the language described by the grammar is finite since it contains a finite number (two) of strings.

Definition: A parse tree (also known as a derivation tree) is a tree rooted at the non-terminal S such that each non-terminal node in the tree has as its children the terminals and non-terminal symbols that are used in rewriting the non-terminal node, and such that each terminal is a leaf node in the tree.

■ **Example 5.** In the grammar where P = {S → aBB, B → b|AS|aAS, A → gg}, the following is a parse tree for the left-most derivation of the string aggabbb. In reading the string of teminals described by a parse tree, the leaf nodes are read, left to right.



Definition: If a string s is in L(G), the meaning of s for G is its parse tree.

A grammar G is ambiguous if there exists some string s in L(G) such that s has more than one parse tree.

Note

- In the example above, we said we were going to provide the parse tree for the left-most derivation of the string aggabbb. There also exists a parse tree for the right-most derivation of the string aggabbb. Does this make the grammar ambiguous? No. The two parse trees are identical (they are the same tree), but when using a left-most derivation, the order in which the tree is drawn will differ from the order in which it is drawn when using a right-most derivation.
- A grammar is called *ambiguous* if its language contains some string that has two different parse trees. This is equivalent to saying that some string has two distinct leftmost derivations or, equivalently, some string has two distinct rightmost derivations.
- Each *string* in L(G) has a meaning; the term ambiguous, however, describes certain *grammars*.

Definition: Two grammars, G_1 and G_2, are equivalent if and only if $L(G_1) = L(G_2)$.

■ **Example** Let $L(G_1) = \{aa, bb, cc\}$ and $L(G_2) = \{\lambda, aa, bb, cc\}$. Because $L(G_1) \neq L(G_2)$, G_1 and G_2 are not equivalent.

Note

It is possible for two equivalent grammars to have different sets of productions, as long as they both generate the same language.

Grammars can be combined with the resulting combined grammar typically generating a new language. To combine two grammars, $G_1 = <N_1,T_1,S_1,P_1>$ and $G_2 = <N_2,T_2,S_2,P_2>$, first assure that $N_1 \cap N_2 = \varnothing$ (If initially this is not the case, simply choose new names for the non-terminals in one of the grammars). Then, the new grammar:
$G = <(N = N_1 \cup N_2), (T = T_1 \cup T_2), S, (P = P_1 \cup P_2$, and some additional productions)>.
Then, by adding the following new productions to P, we get new languages by combining G_1 and $G_{2:}$

Production added to P	Description of new language	Name of rule applied	Description of new language
$S \rightarrow S_0 \mid S_1$	$L(G_1) \cup L(G_2)$	Union rule	$L(G_1) \cup L(G_2)$
$S \rightarrow S_0S_1$	$L(G_1) \bullet L(G_2)$	Product rule	$L(G_1) \bullet L(G_2)$
$S \rightarrow S_1S_0$	$L(G_2) \bullet L(G_1)$	Product rule	$L(G_2) \bullet L(G_1)$

Suppose M and N are languages whose grammars have disjoint sets of nonterminals. (Rename them if necessary.) Suppose that the start symbols for the grammars of M and N are A and B, respectively. Then we have the following new languages and grammars:

Union Rule: The language $M \cup N$ starts with the two productions

$$S \rightarrow A \mid B.$$

Product Rule: The language $M \bullet N$ starts with the production

$$S \rightarrow AB.$$

Closure Rule: The language M^* starts with the production

$$S \rightarrow AS \mid \Lambda$$

For example, suppose we want to write a grammar for the following language:

$$L = \{L, a, b, aa, bb, ..., a^n, b^n, ..\}.$$

After a little thinking we notice that L is the union of the two languages $M = \{ a^n \mid n \in \}$ and $N = \{b^n \mid n \in \}$

Thus we can write a grammar for L as follows:

$S \rightarrow A \mid B$ union rule,

$A \rightarrow \Lambda \mid aA$ grammar for M,

$B \rightarrow \Lambda \mid bB$ grammar for N.

For another example, suppose we want to write a grammar for the following language:

$L = \{ a^m b^n \mid m,n \in \}$

After a little thinking we notice that L is the product of the two languages $M = \{a^m \mid m \in \}$ and $N = \{b^n \mid n \in \}$). Thus we can write a grammar for L as follows:

$S \rightarrow AB$ product rule,

$A \rightarrow \Lambda \mid aA$ grammar for M,

$B \rightarrow \Lambda \mid bB$ grammar for N.

For another example, suppose we want to construct the language L of all possible strings made up from zero or more occurrences of *aa* or *bb*. In other words, $L = \{aa, bb\}^*$. So we can write a grammar for L as follows:

$S \rightarrow AS \mid \Lambda$ closure rule,

$A \rightarrow aa \mid bb$ grammar for $\{aa, bb\}$.

We can simplify this grammar. Just replace the occurrence of A in $S \rightarrow AS$ by the right side of $A \rightarrow aa$ to obtain the production $S \rightarrow aaS$. Also replace A in $S \rightarrow AS$ by the right side of $A \rightarrow bb$ to obtain the production $S \rightarrow bbS$. This allows us to write the the grammar in simplified form as

$$S \rightarrow aaS \mid bbS \mid \Lambda$$

■ **Example 1.** $L(G_1)^*$ can be formed by creating a new grammar G that combines G_1 with itself. Using the union rule above, derivations in the new grammar would start with the production $S \rightarrow S_1S \mid \lambda$.

■ **Example 2.** Describe using sets, concatenation, and Kleene closure the set of all strings made up of a's and b's with an even number of letters.

■ **Answer:** {aa, ab, ba, bb}*

3.1.3 Regular languages

We define Regular Languages (RLs) recursively:

Basis: For all a in A, the following are regular languages: ∅, {λ}, {a}.

Inductive step : If L and M are regular languages, then any product or union of L and M is a regular language.

So, L U M, L•M, M•L, L*, and M* are all regular languages, as are (LUM)•M, and (M•L)*, etc.

■ **Example 1.** {a}{a, b}*{b} is the set of all strings over A = {a,b} that begin with an "a" and end with a "b".

■ **Example 2.** {a,b}* {a} {a,b}* {a} {a,b}* is the set of all strings over A = {a,b} with at least 2 a's.

A Regular Expression (RE) is a way of describing a certain class of sets (regular sets).That is, a regular expression is an algebraic expression describing an RL. REs use parenthesis to group terms for clarity, and three operators, the unary operator*, and the binary operators • and +.

Note

- The empty set ∅ is regular.
- Any set containing one character is regular.

Definition: We define REs recursively:
Basis: ∅, λ, a are REs for all a in A.
Inductive step: If B and C are REs, the following are also REs:
(B), B + C, B • C, B *
The precedence for these operators, from highest precedence to lowest is: *, •, +.

Note

- If it does not affect the clarity of the meaning of an RE, we sometimes write AB rather than A • B.
- Even though the operator + is not explicitly provided by the algebra of REs, it is often used in REs as a shorthand for the following. Let B be a regular expression. Then, by induction, we know that B* is a regular expression. We also know, by induction that BB* is a regular expression. B+ is shorthand for the regular expression, BB.
- Every single alphabet symbol is a regular expression. The empty string, lambda, is a regular expression. The empty set, that is nothing, is a regular expression.
- If R and S are regular expressions then so are R+S, RS, R* and S*. For example 0*1 + 1 is a regular expression. The semantic interpretation of R+S is the union of all strings in R and S. RS is the set of strings xy, where x is in R and y is in S. R* consists of all the strings which are zero or more concatenations of strings in R. For example. 00011101 is in (00+01)*(11+00)01.

Definition: The language L(E) is the set of strings described by the regular expression, E.

Note the following:

- L(∅) = {} = ∅
- L(λ) = {λ}
- L(a) = {a}, for each a in A
- L(R+S) = L(R) U L(S). This is called the language union rule.
- L(R•S) = L(R) • L(S). This is called the language product rule.
- L(R*) = L(R)*. This is called the language closure rule.

■ **Examples**

1. L(a+b) = {a,b}
2. L(a(a+b)) = {aa,ab}
3. L((aa)+b) = {aa, b}
4. L(a+) = L(aa*) = {a, aa, aaa, …}
5. L(a*) = {λ, a,aa, aaa, …}
6. L((ab*) + c) = {c, a, ab, abb, abbb, …}
7. L((ab)* + c) = {λ, c, ab, abab, ababab, …}
8. a(a + b)*b is the set of all strings that starts with an "a" and ends in a "b".
9. (b*abb*) is the set of all strings in which each "a" is followed by at least one "b".

The properties of RE's are given as follows:

1. **The + properties**

   R + T = T+R is commutativity property.

   R + ∅ = R is "zero" property.

   R + R = R is identity property.

   (R + S) + T + R + (S + T). associative property.

2. **The • properties**
   R∅ = ∅ is "zero" property.
   Rλ = R is identity property.
   (RS)T = R(ST) is associative property.
   Notice that the • operator is not commutative.
3. **Distributive properties**
   R(S + T) = RS + RT
   (S +T)R = SR + TR
4. **Closure properties**
   ∅* = λ* = λ.
5. R* = R*R* = (R*)* = R+R*,
   R* = λ+R* = (λ+R)* = (λ+R)R* = λ+RR*,
   R* = (R+,…,+Rk)*, for any k ≥ 1,
   R* = λ +R+,…,+Rk-1+RkR*, for any k ≥ 1.
6. R*R = RR*
7. (R+S)* = (R*+S*)* = (R*S*)* (R*S)*R* = R*(SR*)
8. R(SR)* = (RS)*R
9. (R*S*)* = λ + (R+S)*S,
   (RS*)* = λ + R(R+S)*

Identities with regular expressions

1. ϕu = uϕ = ϕ (ϕ = empty set)
2. λu = uλ = u
3. ϕ* = l
4. λ* = λ
6. u + ϕ = u
7. u + u = u
8. u* = (u*)*
9. u(v+w) = uv + uw
10. (u+v)w = uw + vw
11. (uv)*u = u(vu)*
12. (u + v)* = (u*v)*u*

■ **Example 1.** Find the language of the regular expression $a + bc^*$.

■ **Answer:** We can evaluate the expression $L(a + bc^*)$ as follows:

$$\begin{aligned} L(a + bc^*) &= L(a) \cup L(bc^*) \\ &= L(a) \cup (L(b) \cdot L(c^*)) \\ &= L(a) \cup (L(b) \cdot L(c)^*) \\ &= \{a\} \cup (\{b\} \cdot \{c\}^*) \\ &= \{a\} \cup (\{b\}. \{L, c, c^2, ., c^n, \ldots\}) \\ &= \{a\} \cup \{b, bc, bc^2, ... bc^n, \ldots\} \\ &= \{a, b, bc, bc^2, ..., bc^n, ...\}. \end{aligned}$$

■ **Example 2.** For example, the language $\{a, b, c\}$ is represented by the regular expression $a + b + c$.

Note: All infinite languages need not be regular.

■ **Example 3.** The language $\{\Lambda, a, b, ab, abb, abbb, ..., ab^n, ...\}$ is regular as it can be represented by the regular expression $\Lambda + b + ab^*$.

Note: Distinct regular expressions do not always represent distinct languages. For example, the regular expressions $a + b$ and $b + a$ are different, but they both represent the same language, $L(a + b) = L(b + a) = \{a, b\}$.

■ **Example 4.** Prove the following equality:

$ba^*(baa^*)^* = b(a + ba)^*$.

■ **Answer:**

By cancelling b from both sides, we get

$a^*(baa^*)^* = (a + ba)^*$.

Let $R = a$ and $S = ba$. Then we have

$(a + ba)^* = (R + S)^*$. ***By using the closure property of RE, this can be written as:***

$R^*(SR^*)^* = a^*(baa^*)^*$.

Therefore the equality is proved.

■ **Example 5.** Show that $(\varnothing + a + b)^* = a^*(ba^*)^*$

By starting with the left side as follows:

$$(\varnothing + a + b)^* = (a + b)^*$$
$$= a^*(ba^*)^* \quad (\text{using } (u + v)^* = (u^*v)^*u^*)$$

■ **Example 6.** Show that

$b^*(abb^* + aabb^* + aaabb^*)^* = (b + ab + aab + aaab)^*$

By starting with the left side and proceeding towards right side.

$b^*(abb^* + aabb^* + aaabb^*)^*$ *by using* $b^*((ab + aab + aaab)b^*)^*$

$= (b + ab + aab + aaab)^* \quad ((u + v)^* = (u^*v)^*u^*)$.

■ **Example 7.** Show that $R + RS^*S = a^*bS^*$, where $R = b + aa^*b$ and S is any regular expression:

$$R + RS^*S = R\Lambda + RS^*S$$
$$= R(\Lambda + S^*S)$$
$$= R(\Lambda + SS^*)$$
$$= RS^*$$
$$= (b + aa^*b)S^*$$
$$= (\Lambda + aa^*)bS^*$$
$$= a^*bS^*$$

■ **Example 8.** If $\Sigma = \{a,b\}$, describe using regular expressions the following languages:

1. All strings that contain the substring "ab"
2. All strings that start with a "b", end in a "b" and have at least one "a" in between
   All strings that *do not* have the substring ab
   All strings that contain an *odd* number of b's
3. All strings in which the total number of a's is divisible by 3.
4. All strings that end in a double letter (either "aa" or "bb")
5. All strings that have a double letter somewhere in them
6. All strings that *do not* contain the string aaa
7. All strings with length <4
8. All strings in which each "a" is preceded by a "b"
9. All strings with an even number of a's and an odd number of b's
10. All strings that contain exactly 2 b's
11. All strings that contain an odd number of occurences of the substring "ab"

12. All strings with odd length
13. All strings with odd length containing exactly one "a"

Note

- The symbol Φ is a regular expression and L(Φ) = Φ ;
- Any single character c is a regular expression and L(c) = {c}
- If α and β are regular expressions then $\alpha\beta$, $\alpha U\beta$ and α^*, β^* are regular expressions. Also, L($\alpha U\beta$) = L(α)UL(β), L($\alpha\beta$) = L(α)L(β), L(α^*) = (L(α))*.
- A language L is regular if and only if there is a regular expression E with L = L(E).

■ **Example 9.** Smallest possible string length under this Regular expression a(a + b)*b is------

■ **Answer:** 2

■ **Example 10.** Does the string "ab" is smallest acceptable string in both the languages which are represented by regular expressions a(a+b)*b and (b*abb*)?

■ **Answer:** Yes

3.1.4 Finite State Machines

A Finite State Machine (FSM) is a kind of very limited type of computation that has no memory or data structure except for what you can encode directly into the machine. Every FSM looks like a directed graph where the nodes are called states and the edges are called transitions. The edges are labeled with symbols of the alphabet, and we run the machine by reading the input one symbol at a time from left to right, and following the transitions in the graph. We start at a designated start node. When we are finished reading the string, acception of that string depends on the state that we end up in. If the state is marked final, then we say yes, otherwise no. **Any set of strings accepted by an FSM is called a regular set.**

Examples of Regular sets for the alphabet {0,1}.

a. Even number 1's.
b. Strings containing 101.
c. Even number of 0's and contains 101.
d. Even number of 0's or contains 101.
e. All strings. Divisible by three as a binary number.
f. Everyone has at least two zeros that follow it.
g. Not divisible by three.
h. Second symbol not a one.
i. Some two zeros are separated by a number of symbols that is a multiple of three.
j. End with 00 or 01.

A Finite Automaton (FA) is a directed graph completely described by a 5-uple: T<S, A, T, s, F>,where, S is a finite set of nodes or States, A is the finite Alphabet of terminal symbols, T is the Transition function providing the mapping S X A -> S, s in S is the unique Starting node or state, and F (a subset of S) is a set of Final or accepting states.

There are two large groupings of FA's: Deterministic Finite Automata (DFA's) and Non-Deterministic Finite Automata (NFA's). The above definition of FA's applies to both groups, although the structure of T will differ according to whether the FA being described is a DFA or an NFA.

Remember that an FA is a mathematical structure (a directed graph). One (popular) way to depict this mathematical structure is with a drawing in which all the following occur:

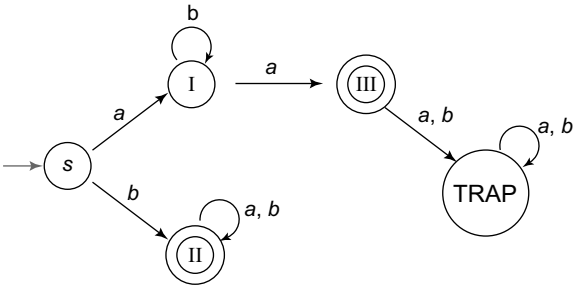
- Elements of S are represented as labeled nodes in a graph (usually drawn as circles with the label in the circle).
- T is represented by edges in the graph, with each edge labeled with some symbol from A.

- Elements of F are indicated by drawing another circle around each circle (or labeled node) that represents a final state.

An FA can be used to generate a language by applying the following algorithm:

1. Start at the start state s and with the empty string (the string of length 0) as the current string.
2. Each time a transition from a state to a state is effected, the label on the edge of that transition is concatenated with the current string.
3. When a final state is reached, either
   - Place the current string in the set of strings generated by the FA and go to step 1, or
   - (if possible) go back to step 2.

Here is an example of an FA: <(S = {s, I, II, III, TRAP}, A = {a, b}, T (provided later), s, (F = {II, III})>.
And here is a graphic representation of that FA:

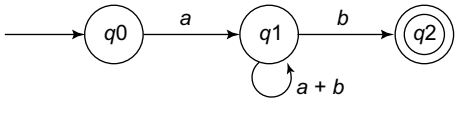


The function T can be completely specified and described for FA as shown in Table 3.1 :

Table 3.1

				Comments about the table and T (not actually part of T)
				Each member of S is listed in column 2, with any special designations for that member listed in column 1; each symbol from the alphabet is listed as a header for subsequent columns (that is, after column 2).
		A	b	
Start	S	I	II	Move from state S to state I and concatenate 'a' with the current string; or, from S to state II and concatenate 'b' with the current string
	I	III	I	
Final	II	II	II	Final state means: if here, the current string is in the language generated by the FA
Final	III	TRAP	TRAP	Another final state Can not get anywhere from trap state.
	TRAP	TRAP	TRAP	

In some TOC books, FSA **state diagram** is represented as shown, also where q0,q1, ... representing states, etc., Here, q0 is always assumed as initial state, S is set of alphabet.



Q = {q0, q1, q2}

S = {a, b}

δ(q0, a) = q1//If the input alphabet is a then next state becomes q1.

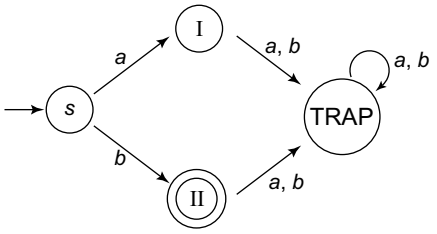
δ(q1, a) = q1//If the input alphabet is a then next state becomes q1.

δ(q1, b) = q1//If the input alphabet is b then next state becomes q1.

δ(q1, b) = q2//If the input alphabet is a then next state becomes q2.

F = {q2}

The above FSA recognises strings which starts with a and ends with b.
Consider the FA given in the picture below:

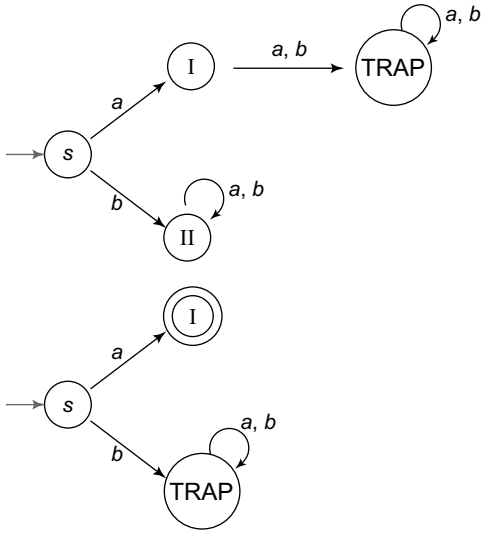


■ **Example** What is the language generated by the FA in the above figure?

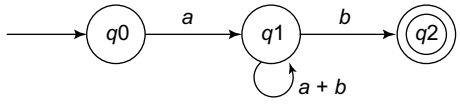
■ **Answer:** {b}.
A language L is *recognised* by a FA if L is exactly the language generated by the FA. So, we might ask the question: is the language b* recognised by the FA in the given figure. If we can find at least one string that is in L but cannot be generated by the FA, the answer is 'no.' We can start by enumerating some of the strings in L(b*). L(b*) = {λ, b, bb, bbb, …}. Because the string bbb (among others) is not in the set of strings generated by the FA, the FA does not recognise the language L(b*).

■ **Eaxmple** What are the languages generated by the FA's in the following figures?

■ **Answer:** First figure: Strings starts with b's.
Second figure: {a}



■ **Example 1.** Show the set of transitions used to test whether the string aabbab is acceptable in the following FSA.



■ **Answer:**

δ(q0, a) = q1

δ(q1, a) = q1

δ(q1, b) = q1

$\delta(q1, b) = q1$

$\delta(q1, a) = q1$

$\delta(q1, b) = q2$

The string terminates in a final state (i.e. q2), so this automaton recognises the string aabbab or aabbab is in the language recognised by this automaton.

Regular Expressions and FA

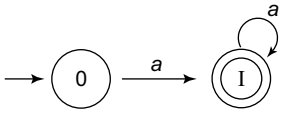
The set of languages described by the set of Regular Expressions is exactly the set of Regular Languages.

- There exists an algorithm to transform any Regular Expression into a Finite Automaton.
- There exists an algorithm to transform any Finite Automaton into a Regular Expression.
- Therefore, the set of languages generated/recognised by the set of Finite Automata is exactly the set of Regular Languages.

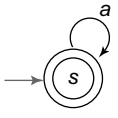
An FA is fully specified for an alphabet A if from every state in the FA there is exactly one transition specified for each symbol in A. If an FA is not fully specified, it can be made so by adding states and transitions as needed, so long as these do not affect the language generated by the FA.

Some Regular Expressions and their corresponding FA (parts) are given below.

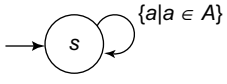
- a^+



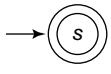
- a^*



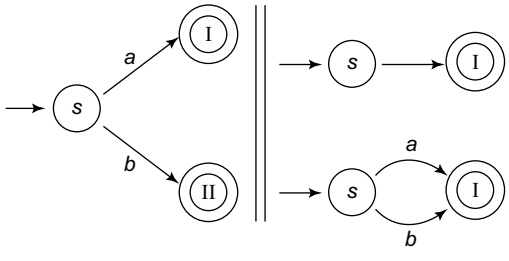
- ∅



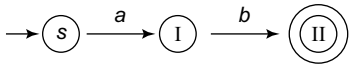
- λ



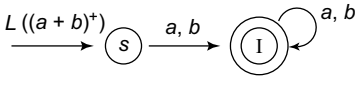
- a + b



- a b

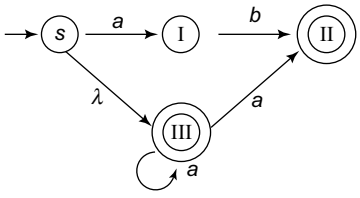


- (a + b)+)

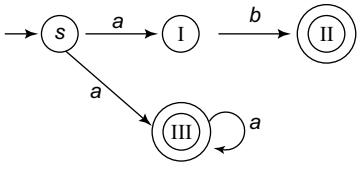


λ-transitions

If an edge of an FA is labeled with λ, this signifies that a transition from a state to a state along this edge can be made without "consuming" a character from the input string. For example, try to construct an FA to recognise the language L(ab+a*a). One such machine is shown in the following figure:



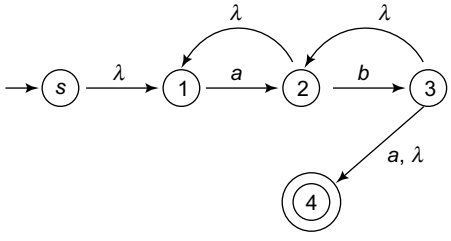
Note: For each FA with λ-transitions, there exists at least one equivalent FA without λ-transitions. Because they recognise the same language, the machine in the following figure (without λ-transitions) is equivalent to the machine in above figure (having λ-transitions).



Definition: The λ-closure of s is the set of states that can be reached from s by traversing zero or more edges labeled with λ ("λ-edges").

The λ-closure of s is denoted by λ(s).

Consider the following example,



The transition table for above figure is given as

		A	B	λ
Start	s	--	--	{1}
	1	{2,3}	--	--
	2	--	{3}	{1}
	3	{4}	--	{2,4}
Final	4	--	--	--

So,

$\lambda(s) = \{s,1\}$

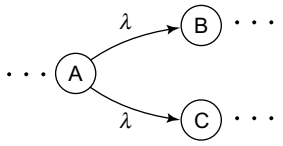
$\lambda(1) = \{1\}$

$\lambda(2) = \{1,2\}$

$\lambda(3) = \{1,2,3,4\}$

$\lambda(4) = \{4\}$

In other words, if an FA contains the states A, B, C, and if there exists a λ-edge from state A to state B and another λ-edge from state B to state C, as shown in the given figure,

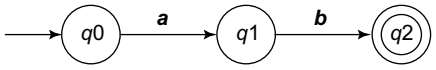


then by applying the recursive definition of λ-closure,

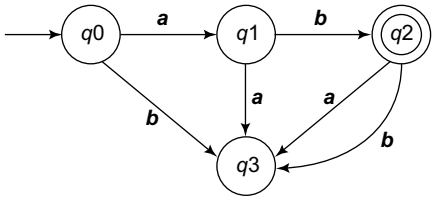
$\lambda(A) = \{A,B,C\} \cup \lambda(B) \cup \lambda(C)$.

Theorem 1: If M = (Q, Σ, δ, q0, F) is a finite automaton, then M' = (Q, Σ, δ, q0, Q – F) is a finite automaton with L(M') being the set complement of L(M).

A **complete** finite automaton is an automaton in which *each* node has a transition leaving that node *for each character in the alphabet*. Any finite automaton can be transformed into a complete finite automaton by creating a new node and having each node that "needs" a transition given a transition to the new node. In this way the new node acts as a sink. See the following which contains an FSA with its completed version.

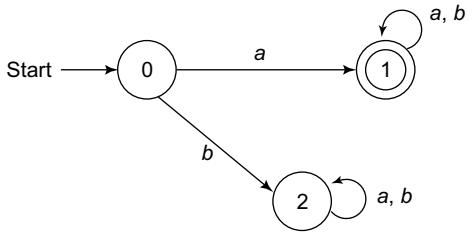


A completed automaton would be :



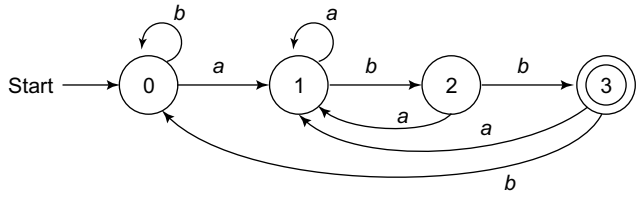
Theorem 2: If M = (Q, Σ, δ, q0, F) is a **complete** finite automaton, then M' = (Q, Σ, δ, q0, Q – F) is a finite automaton with L(M') being the set complement of L(M).

■ **Example 1.** Which is trap state in the following DFSA?



■ **Answer:** 2

■ **Example 2.** Build a DFA to recognise the regular language represented by the regular expression *(a + b)*abb* over the alphabet *A = {a, b}*. The language is the set of strings that begin with anything, but must end with the string *abb*.



3.1.4.1 Non-deterministic Finite State Machines

We now consider a variation of the FSM that will make it much easier to design FSM's. A deterministic FSM has an arrow coming out of each state for each symbol in the alphabet. If an arrow is missing, we assume that it actually goes to a dead state, that is a state with two self-looping arrows (for 0 and 1). A non-deterministic machine may have any number of arrows coming out of each states with 0 and 1 appearing on the transitions arbitrary number of times.

How do we run a non-deterministic machine? We have potentially many choices of directions to move after reading each symbol of the input. Well we don't really run the machine, at least not in the sense that we run a deterministic machine. However, there is a very formal and precise definition as to which strings are accepted by a given non-deterministic FSM. If we can read the symbols of a string and find some path that ends up in a final state then we accept that string. On the other hand, if every choice of path ends up in a non-final state then we reject the string.

Let's consider what *T* (the transition function) of an FA might be. We know we can represent T as either a transition diagram or as a transition table. As a mathematical expression, we can state T as:

$$T(i, a) = j$$

which might be read as "*There exists a transition from state i to state j labeled with 'a'.*" We can define T more precisely using induction.

Basis step: T(i, lambda) = i

Inductive step: T(i, as) = T(T(i, a), s)

Note that because T is a transition function, T(i, a) is a state.

Converting an arbitrary DFA, M into an NFA is trivial. All we have to do is add a non-accepting state to M and duplicate some existing edge label into that state so that there will exists alternative edges with the same label from some state in M.

Algorithm to convert an NFA into a DFA

1. The DFA state is lambda(s), where s is the NFA start state.
2. For each state in the DFA {s_1, s_2, … , s_n}, and for each a in A, construct the transition table as follows
   T_D({s_1, s_2, …, s_n}, a) =
   lambda(T_N(s_1,a) U … U T_N(s_1,a)).
   (Where T_D is the transition function for the DFA and T_N is the transition function for the NFA)
3. A DFA state is final if any of its elements is a final state in the corresponding NFA.

3.1.4.2 Minimising FSM's

The algorithm to minimise an FSM is based on the fact that the states can be partitioned into disjoint sets where all the states in each state are equivalent. This equivalence relation is defined as follows: two states A and B are equivalent if a given string could start in either place and its acceptance status would be identical.

For all FA's that recognise the same language, L are equivalent. Among these there exists a unique DFA with the smallest number of states. This is called the *minimum-state DFA* that recognises L. Here we are using "minimum" in the strict sense that there does not exist an equivalent DFA with a smaller number of states.

First, recall that the set of regular languages is exactly the set of languages generated by regular expressions. Recall, too, that there is an algebra of regular expressions that would permit us to simplify any RE. That is, suppose we have an RE and wish to simplify it. We can use the RE to create an FA that recognises the same language. If the FA is an NFA, we can use the algorithm previously detailed to transform the FA into a DFA. Then, we can use the algorithm to be presently described to find the unique minimum-state DFA equivalent to that RE. If necessary, there exists an algorithm that we could then apply to transform the minimum-state DFA into a (now) simplified RE.

This means for any regular expression, or for any NFA, or for any DFA, we can construct a unique minimum-state DFA that recognises the same (regular) language.

The algorithm we are about to present is based on the following notion of equivalence. For any string w, two states (s, and t) in the DFA are said to be equivalent if T(s, w) and T(t, w) are either both accepting states or both non-accepting states.

Recall that an equivalence relation is a relation that is RST (Reflexive, Symmetric, and Transitive).

The idea behind this is fairly simple. Suppose, whatever path through the DFA was used to get to states s and t, that with the string w left to "consume" when moving from states s and t, the machine will end up in an accepting state at the end of the string w. Then s and t are equivalent in the sense that both T(s, w) and T(t, w) are accepting states.

And the contrary is true. Suppose, whatever path through the DFA was used to get to states s and t, that with the string w left to "consume" when moving from states s and t, the machine will end up in a non-accepting state at the end of the string w. Then s and t are equivalent in the sense that both T(s, w) and T(t, w) are non-accepting states.

This notion of equivalence allows us to partition the states of a DFA according to the following algorithm.

Algorithm for constructing a minimum-state DFA from a DFA, M

Step 0. Create the initial sets I_N and I_A, where each element of I_N is a set containing a single non-accepting state, and each element of I_A is a set containing a single accepting state.

Step 1. Create a set of states $E_{i,}$ (i = 0) for each pair of accepting states and for each pair of non-accepting states.

Step 2. Form subsets of equivalencies in the following manner. For all sets {s, t} in E_i, if there exists some letter a such that {T(s, a), T(t, a)} is not in E_i, then {s, t} are not equivalent, so we discard {s, t}. Place the non-discarded members of E_i in a set E_{i+1} such that E_{i+1} is a subset of E_i.

Step 3. Increment i and repeat Step 2 until, at some value of i, no pairs are discarded, that is, until $E_i = E_{i-1}$.

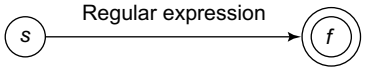
Step 4. Now use E_{i-1} to partition I_N and I_A into equivalence classes of sets of states.

Step 5. The start state of the DFA will be the equivalence class containing M's original start state.

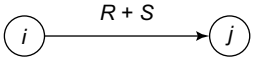
Step 6. Any equivalence class containing an accepting state is an accepting state.

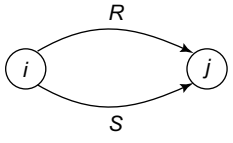
Regular Expression to Finite Automaton

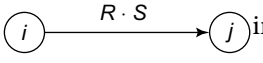
Given a regular expression, we start the algorithm with a machine that has a start state, a single final state, and an edge labeled with the given regular expression as follows:

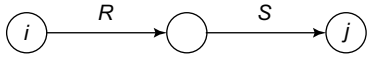


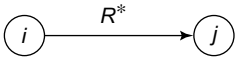
Now, transform this machine into a DFA or an NFA by applying the following rules until all edges are labeled with either a letter or Λ:

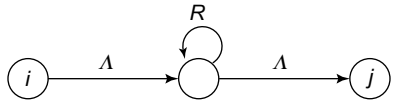
1. If an edge is labeled with 0, then erase the edge.
2. Transform any diagram like 

in to the diagram 

3. Transform any diagram like 

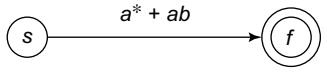
into the diagram 

4. Transform any diagram like 

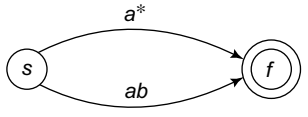
into the diagram 

End of Algorithm

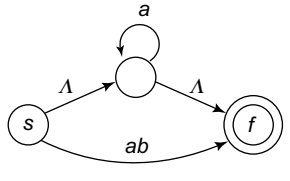
■ **Example** To construct an NFA for $a^* + ab$, we'll start with the diagram



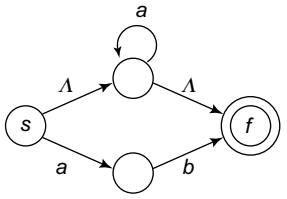
Next we apply rule 2 to obtain the following NFA:



Next we will apply rule 4 to a^* to obtain the following NFA:



Finally, we apply rule 3 to *ab* to obtain the desired NFA for $a^* + ab$:



Transforming Finite Automata into Regular Expressions

Starting with either a DFA or an NFA, the algorithm performs a series of transformations into new machines, where these new machines have edges that may be labeled with regular expressions. The algorithm stops when a machine is obtained that has two states, a start state and a final state, and there is a regular expression associated with them that represents the language of the original automaton.

Assume that we have a DFA or an NFA. Perform the following steps:

1. Create a new start state *s*, and draw a new edge labelled with Λ from *s* to the original start state.
2. Create a new final state *f*, and draw new edges labelled with Λ from all the original final states to *f*.
3. For each pair of states *i* and *j* that have more than one edge from *i* to *j*, replace all the edges from *i* to *j* by a single edge labelled with the regular expression formed by the sum of the labels on each of the edges from *i* to *j*.
4. Construct a sequence of new machines by eliminating one state at a time until the only states remaining are *s* and *f*. As each state is eliminated, a new machine is constructed from the previous machine as follows:

Eliminate State k

For convenience we will let *old (i,j)* denote the label on edge $\langle i,j \rangle$ of the current machine. If there is no edge $\langle i, j \rangle$, then set *old(i, j)* = ∅. Now for each pair of edges $\langle i, k \rangle$ and $\langle k,j \rangle$, where $i \neq j$ and $j \neq k$, calculate a new edge label, *new(i, j)*, as follows:

$$new(i,j) = old(i,j) + old(i, k)\ old(k, k)^*\ old(k,j).$$

For all other edges $\langle i,j \rangle$ where $i \neq k$ and $j \neq k$, set

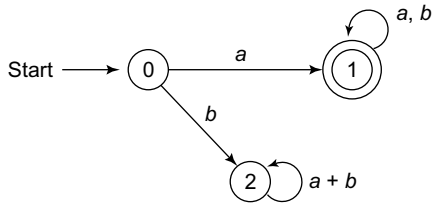
$$new(i, j) = old(i, j).$$

The states of the new machine are those of the current machine with state *k* eliminated. The edges of the new machine are the edges $\langle i, j \rangle$ for which label *new(i, j)* has been calculated.

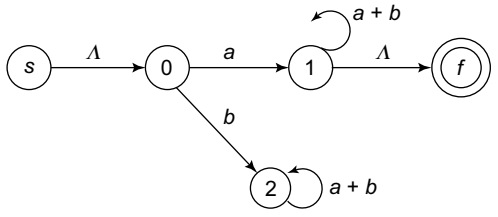
Now s and f are the two remaining states. If there is an edge $\langle s, f \rangle$, then the regular expression *new(s, f)* represents the language of the original automaton. If there is no edge $\langle s,f \rangle$, then the language of the original automaton is empty, which is signified by the regular expression ∅.

End of Algorithm

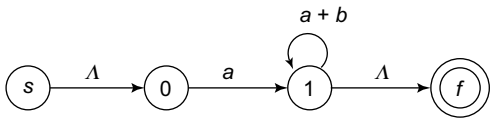
■ **Example 1.** Suppose we start with the DFA:



1. The first three steps transform this machine into the following machine, where *s* is the start state and *f* is the final state:



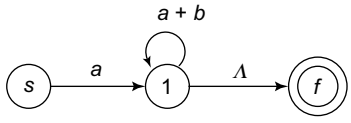
Now let us eliminate the states 0, l, and 2. We can eliminate state 2 without any work because there are no paths passing through state 2 between states that are adjacent to state 2. In other words, *new(i,j) = old(i,j)* for each edge ⟨*i, j*⟩, where $i \neq 2$ and $j \neq 2$. This gives us the machine



Now we will eliminate state 0 from this machine by adding a new edge ⟨*s*, 1⟩ that is labelled with the following regular expression:

$$\begin{aligned} new(s, 1) &= old(s, 1) + old(s, 0)\ old(0, 0)^*\ old(0, 1) \\ &= \varnothing + \Lambda\varnothing^* a \\ &= a \end{aligned}$$

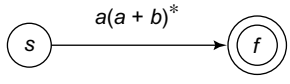
Therefore, we delete state 0 and add the new edge ⟨*s*, 1⟩ labeled with *a* to obtain the following machine:



Next we eliminate state 1 in the same way by adding a new edge ⟨*s, f*⟩ labelled with the following regular expression:

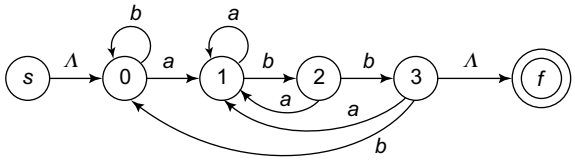
$$\begin{aligned} new(s,f) &= old(s, f) + old(s, 1)\ old(1, 1)^*\ old(1, f) \\ &= \varnothing + a(a + b)^*\Lambda \\ &= a(a + b)^*. \end{aligned}$$

Therefore, we delete state 1 and label the edge ⟨*s, f*⟩ with $a(a + b)^*$ to obtain the following machine:



This machine terminates the algorithm. The label $a(a + b)^*$ on edge ⟨*s, f*⟩ is the regular expression representing the regular language of the original DFA given in the figure.

■ **Example 2.** Verify that the regular expression $(a + b)^*abb$ represents the regular language accepted by the DFA from. We start the algorithm by attaching start state *s* and final state *f* to the DFA to obtain the following machine:

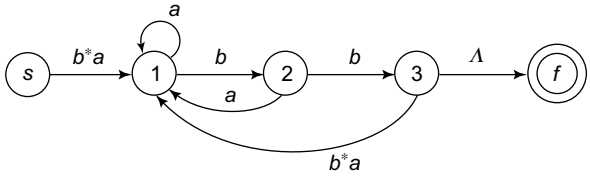


Now we need to eliminate the internal states. As we construct new edge labels, we'll simplify the regular expressions as we go. First we will eliminate the state 0. To eliminate state 0, we construct the following new edges:

$$new(s, 1) = \varnothing + \Lambda b^* a = b^* a,$$

$$new(3, 1) = a + bb^* a = (\Lambda + bb^*)a = b^* a.$$

With these new edges we eliminate state 0 and obtain:



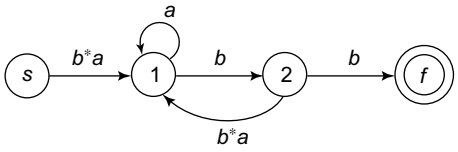
The states can be eliminated in any order. For example, we will eliminate state 3 next, which forces us to create the following new edges:

$new(2, f) = \varnothing + b\varnothing^*\Lambda = b,$

$new(2, 1) = a + b\varnothing 0^*b^*a = a + bb^*a$

$= (\Lambda + bb^*)a = b^*a.$

With these new edges we eliminate state 3 and obtain the following machine:

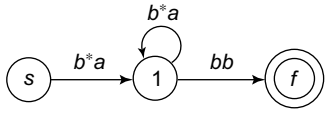


Next, we will eliminate state 2, which forces us to create the following new edges:

$new(1, f) = \varnothing + b\varnothing^*b = bb,$

$new(1, 1)\ a + b\varnothing^*b^*a = a + bb^*a = (\Lambda + bb^*)a = b^*a.$

With these new edges we eliminate state 2 and obtain the following machine:



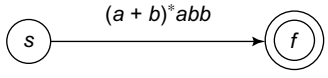
Finally, we remove state 1 by creating a new edge

$new(s,f) = \varnothing + b^*a(b^*a)^*bb$

$= b^*(ab^*)^*abb$ (by 8 of Properties of Reg Exp)

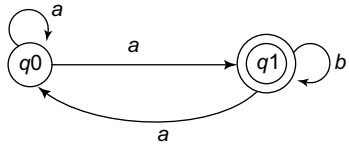
$= (a + b)^*abb$ (by 7 of Properties of Reg Exp).

So we obtain the last machine with the desired regular expression for the DFA as given:



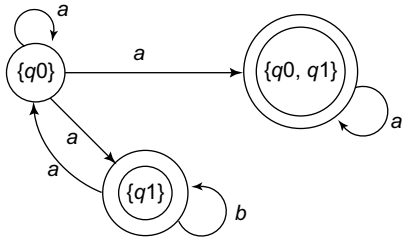
The process of constructing a regular expression from a finite automaton can produce some complex regular expressions. If we remove the states in different orders, then we might obtain different regular expressions, some of which might be more complex than others. So the algebraic properties of regular expressions are nice tools to simplify these complex regular expressions. As we have indicated in the example, it is better to simplify the regular expressions at each stage of the process to keep things manageable.

■ **Example 3.** Convert the following NFA machine to DFA.

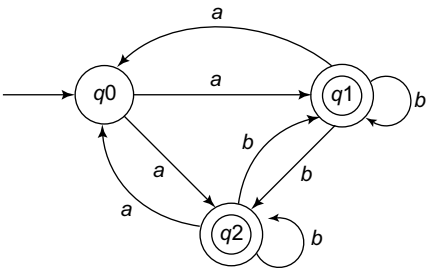


There would be four possible states for resulting DFA. {} {q0} {q1} {q0, q1} Note: {} is usually unnecessary unless we want a complete deterministic FA. We have the original 4 transitions: $\{q^1\} \xrightarrow{a} \{q^0\}$ $\{q^0\} \xrightarrow{a} \{q^1\}$ $\{q^0\} \xrightarrow{a} \{q^0\}\{q^1\} \xrightarrow{b} \{q^1\}$ but we also have: $\{q^0\} \xrightarrow{a} \{q^0, q^1\}$ $\{q^0, q^1\} \xrightarrow{a} \{q^0, q^1\}$

The resulting automaton is:

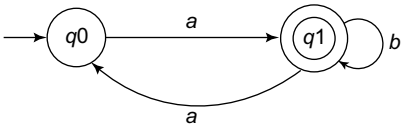


■ **Example 4.** Explain what is meant by equivalence states. Identify which are the equivalence states in the following FA.



Finite automation can be reduced by removing EQUIVALENT STATES. Two states are equivalent if they have the same transitions entering from the same nodes and have the same transitions leaving to the same nodes. Also, they must either both be final states or both non-final states.

Here, q1 and q2 are equivalent states. So, the finite automation can be reduced to an equivalent finite automation with 2 states:



3.1.5 Right Linear Grammars

We have already understood that a grammar is described by *productions.* Best to start with an example and then give some terminology.

$S \rightarrow 0B \quad B \rightarrow 0S \quad S \rightarrow 1S \quad B \rightarrow 1B \quad S \rightarrow \lambda$

S and B (normally all upper case letters) are called non-terminal symbols because they do not appear in the final generated strings. 0, 1, and λ are called terminal symbols, where λ is the empty string. All strings generated by the grammar consist of non- λ terminal symbols.

There is a unique non-terminal symbol called the start symbol usually designated as S. A production is a substitution of one set of symbols for another. The left side is replaced by the right side. A sequence of productions starting with S and ending with a terminal string x, is said to generate the string x. For example, $S \rightarrow 0B \rightarrow 00S \rightarrow 001S \rightarrow 0011S \rightarrow 0011$, shows that the grammar generates the string 0011. See if you can figure out which production was used in each step of the *derivation* of this string.

The grammar above is a very special kind of grammar called a right-linear grammar. In a right-linear grammar, every production has a single non-terminal symbol on the left, and either a single terminal symbol on the right or a combo terminal followed by non-terminal symbol.

Every right-linear grammar generates a regular set, and that every regular set can be generated by a right-linear grammar.

The grammar above corresponds to the set of strings over the alphabet {0,1} that have an even number of zeros. The idea is that the non-terminal symbols correspond to states, and the terminal productions correspond to final states.

Context free grammars keep the restriction on the left side of each production but lose the restriction on the right side. This lets us use a production $S \rightarrow 0S1$, which combined with $S \rightarrow \lambda$ generates the language 0^n1^n, which is not a regular set. There are grammars in between context free and right-linear grammars called LR-grammars. There are less restricted grammars called context-sensitive grammars, which demand only that the length of the left side be smaller than the length of the right side. There are unrestricted grammars, which can recognise any set that a Turing machine can recognise. The LR grammars have a restriction that is complicated to describe but they represent the most practical use of grammars, and are fundamental in the design of compilers and for parsing.

3.1.5.1 The Pumping Lemma

The pumping lemma for regular languages describes an essential property of all regular languages. It says that all sufficiently long words in a regular language may be pumped; that is, have a middle section of the word repeated an arbitrary

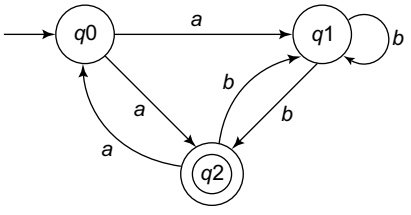
number of times to produce a new word which also lies within the same language. According to pumping lemma, for any regular language L there exists a constant p such that any word w in L with length at least p can be split into three sub-strings, w = xyz, where the middle portion y must not be empty, such that the words xz, xyz, xyyz, xyyyz, … constructed by repeating y an arbitrary number of times (including zero times) are still in L. This process of repetition is known as "pumping". Moreover, the pumping lemma guarantees that the length of xy will be at most p, imposing a limit on the ways in which w may be split. Finite languages trivially satisfy the pumping lemma by having p equal to the maximum string length in L plus one.

Let R be a Regular set. For any z in R, there exists an n (specifically, the number of states in a machine accepting R), such that z can be written as vwx, where |vw| <= n and |x| >= 1, and for i >= 0 vw^ix is also in R.

The lemma is usually used in its contra positive form. That is, if the *pumping* condition is not true then the set is not Regular. Note that if the pumping condition is true, the set might be Regular and might not be. That is, the converse of the pumping lemma is not true. It is not if and only if. You will be asked in a problem set to identify a non-Regular set where the pumping lemma is true.

There are many sets which can be shown to be non-Regular using the pumping lemma.

This lemma can be also said as L: Let M be a finite automaton with k states and let z be a string that M recognizes with the length of z being >= k (|z| >=k), then z can be written as z = uvw where |uv| <= k, |v| >= 0, and M recognizes uv^iw for all i >=0.



Let us take an example FA as shown above.

k = 3 and suppose z = abbaa

Since |z| = 5 > k, then we can break z into

u = a, v = bb, and w = aa

Then uv^0w = aaa

uv^1w = abbaa

uv^2w = abbbbaa

The pumping lemma is used to show that a language IS NOT regular.

■ **Example 1.** The language $\{a^ib^i \mid i >=0\}$ is not regular

Proof:

L contains strings of arbitrary length.

Suppose L is a regular language. (We will show this is impossible). Let M be a finite automaton that generates L and let z ε L and |z| >=k, where k is the number of states of M. Then by the pumping lemma, z = uvw where |uv| <= k and |v| >=0.

Case 1: $v = a^n$ where n>=1 (i.e. v contains all a's)

Then uv^2w ε L (by the pumping lemma)

but uv^2w has more a's than b's, so cannot be in L.

Case 2: $v = a^nb^m$ m,n >=1 (i.e. v contains both a's & b's)

Then $uv^2w = u(a^nb^m)^2w = u(a^nb^ma^nb^m)w$ which is NOT in L

Case 3: $v = b^m$ (i.e. v is all b's)

Then uv^2w contains more b's than a's

In each possible case we have shown that uv^2w cannot possibly be in L. This violates the pumping lemma and therefore L must not be a regular language!

Diagonalisation – Another Way to Identify Sets that are Not Regular

The pumping lemma is very convenient for showing that certain sets are not regular, but it is not the only way. A more general method that works for all kinds of different machines is called diagonalisation.

We consider binary strings that represent FSM's. Imagine that we encode an FSM itself in binary. There are many ways to do this and one will be discussed in class. Then some FSM's will accept their own representation and some will not. The ones that do are called self-aware, and the ones that do not are called self-hating. Now consider the set of all binary strings that represent self-hating FSM's. Is this set Regular?

If it were Regular, then there is an FSM Q that accepts the set. Does Q accept itself? If it does, then it is a self-hating FSM, which means that it should reject itself. But if it rejects itself, then it is not a self-hating machine, and it should accept itself!

Strange, Q can neither accept or reject itself logically, hence Q cannot exist.

The neat thing about this trick is that it does not depend on the fact that the machine is an FSM. It could just as well be any kind of a machine. There are generalisations of the pumping lemma but the technique of diagonalisation works generally.

Using Closure Properties to Show a Set is not Regular

One can also use closure properties to show that a set is not regular. For example, to show that the set of binary strings with an equal number of zeros and ones is not regular, we note that the intersection of that set with 0^*1^* is 0^n1^n. If the equal 0's and 1's set was regular then so would 0^n1^n, and we know that is not the case, hence equal zeros and ones is also not regular.

Regular Expression	Regular Grammar
a^*	$S \to \Lambda \mid aS$
$(a + b)^*$	$S \to \Lambda \mid aS \mid bS$
$a^* + b^*$	$S \to \Lambda \mid A \mid B$
	$A \to a \mid aA$
	$B \to b \mid bB$
a^*b	$S \to b \mid aS$
ba^*	$S \to bA$
	$A \to \Lambda \mid aA$
$(ab)^*$	$S \to \Lambda \mid abS$

■ **Example 2.** Construct a regular grammar for the language of the regular expression a^*bc^*. First we observe that the strings of a^*bc^* start with either the letter a or the letter b. We can represent this property by writing down the following two productions, where S is the start symbol:

$$S \to aS \mid bC.$$

These productions allow us to derive strings of the form bC, abC, $aabC$, and so on. Now all we need is a definition for C to derive the language of c^*. The following two productions do the job:

$$C \to \Lambda \mid cC.$$

Therefore a regular grammar for a^*bc^* can be written as follows:

$$S \to aS \mid bC$$

$$C \to \Lambda \mid cC$$

■ **Example 3.** Construct grammar for the language of strings of a's followed by strings of b's. The largest language of this form is the language $\{ a^m b^n \mid m, n \in \mathbb{I} \}$, which is represented by the regular expression a^*b^*. A regular grammar for this language can be written as follows:

$$S \to \Lambda \mid aS \mid B$$

$$\mathrm{B} \to \mathrm{b} \mid \mathrm{bB}$$

■ **Example 4.** Four sublanguages of $\{ a^m b^n \mid m, n \in \mathbb{I} \}$ that are defined by whether each string contains occurrences of a or b. The following table shows each language together with a regular expression and a regular grammar.

Language	Expression	Regular Grammar
$\{a^m b^n \mid m \geq 0 \text{ and } n > 0\}$	a^*bb	**S → aS | B**
		B → b | bB
$\{a^m b^n \mid m > 0 \text{ and } n \geq 0\}$	aa^*b^*	**S → aA**
		A → aA | B
		B →Λ | bB
$\{a^m b^n \mid m > 0 \text{ and } n > 0\}$	aa^*bb^*	**S → aA**
		A → aA | B
		B → b | bB
$\{a^m b^n \mid m > 0 \text{ or } n > 0\}$	$aa^*b + a^*bb^*$	**S → aA | bB**
		A → Λ | aA | B
		B → Λ | bB

3.1.6 Context-free Languages

We now jump a level in power and we consider a new collection of sets from a grammar point of view. A context-free grammar is a grammar where every production has a single non-terminal symbol on the left side. Context free grammars can generate non-regular states. For example, the 0^n1^n set is described by the language S → 0S1 | λ. The design of context-free grammars is more of an art than a science but there are least two identifiable strategies. One is inductive, and the other semantic.

If a set is described inductively, then you can often leverage the inductive definition and turn it into a grammar. For example, consider the set of balanced parentheses. This can be defined inductively: λ is a balanced string. If x is a balanced string then so is (x). If and x and y are balanced strings then so is xy (concatenation not multiplication). This can be turned into a grammar by having each inductive rule be a production. We get S → (S) | S → SS | S → λ.

If a set is described semantically, we can sometimes make a grammar by assigning semantic information to each non-terminal symbol. For example, consider the set of strings that have an equal number of zeros and ones, but not necessarily in any order. Let S generate these strings. Then we have S → 0A and S → 1B, where A generates strings that have one more 1 than 0, and B generates strings that have one more 0 than 1. We then continue A → 1S and B → 0S, and A → 0AA and B → 1BB. Finally, S → λ, A → 1 and B → 0. All this is consistent with the semantic interpretation of A, B and S.

There is a third strategy that is used for grammars that is not necessarily context-free. The strategy is to use the grammar to simulate a machine-like computation. For example, consider the set of binary strings $0^n1^n0^n$.

In class we will consider the following context sensitive grammar and explain how it is essentially simulating a computation. D and C move back and forth like duck in a shooting gallery. L and R are bookends making sure that no symbols move off the edges. A, B and C represent what will eventually be the terminal symbols 0^n 1^n and 0^n, respectively.

Here is the grammar:

S → LDABCR	ADA→AAD	BDB→BBD	CDC→CCD
DR → ER	LDA→LAAD	ADB→ABBD	BDC→BCCD
CE → EC	BE → EB	AE → EA	LE → LD
LD → λ	R → λ	A → 0	B → 0 C→0

Meaning of context free

The term "context-free" comes from the requirement that all productions contain a single non-terminal on the left. When this is the case, any production $S \to w$ can be used in a derivation without regard to the "context" in which S appears. For example, we can use this rule to make the following derivation step:

$$aS \Rightarrow aw.$$

A grammar that is not context-free must contain a production whose left side is a string of two or more symbols. For example, the production $Sc \to w$ is not part of any context-free grammar. A derivation that uses this production can replace the non-terminal S only in a "context" that has c on the right. For example, we can use this rule to make the following derivation step:

$aSc \Rightarrow aw.$

Most programming languages are context-free. For example, a grammar for some typical statements in an imperative language might look like the following, where the words in boldface are considered to be single terminals:

S → **while** *E* **do** *S* | **if** *E* **then** *S* **else** *S* | {*S L*} | *I* := *E*

L → *S L* | Λ

E → ... (description of an expression)

I → ... (description of an identifier).

■ **Example 1.** What is the language for which production rule is given as:

S → b | aSc

■ **Answer:** Answer the language if of the form

$a^n bc^n$, for n ≥ 0

That is, L = {b, abc, aabcc, aaabccc, …}, which can be described as: "The set of all strings consisting of a single b preceded by zero or more a's and followed by an equal number of c's."

Combining Context-Free Languages

Suppose *M* and *N* are context-free languages whose grammars have disjoint sets of nonterminals (rename them if necessary). Suppose also that the start symbols for the grammars of *M* and *N* are *A* and *B*, respectively. Then we have the following new languages and grammars:

1. The language $M \cup N$ is context-free, and its grammar starts with the two productions
   $S \rightarrow A \mid B.$
2. The language $M \bullet N$ is context-free, and its grammar starts with the production
   $S \rightarrow AB.$
3. The language M^* is context-free, and its grammar starts with the production
   $S \rightarrow \Lambda \mid AS.$

Note

CFL's are *not* closed under intersection or complement. Thus, we do not know from the properties of CFL's whether L_0 ∩ L_1 or whether L_0′ is/are CFL.

3.1.7 Context-Sensitive Grammars

There are grammars that are more general than the context-free ones described above. A ***context-sensitive grammar*** is a grammar whose productions are of the form

$\alpha \rightarrow \beta$

where the length of the string a is less than or equal to the length of the string β.

The language generated by such a grammar is similarly called a ***context-sensitive language.*** (Every context-free language is context-sensitive).

Of course, one implication of this definition is that the empty string, Λ, is not in any context-sensitive language since any string derivable from the start symbol, *S*, must have length > 1. It is, however, common to allow Λ to be in context-sensitive languages by extending the definition of CSGs to permit the production $S \rightarrow \Lambda$ providing *S* does not appear as a substring in the right hand side of any production.

We will follow this convention. Then, if *L* is a context sensitive language generated by G = ⟨*N, T, S, P* ⟩ and $\Lambda \notin L$ we can easily construct the context sensitive grammar:

$G' = \langle N \cup \{S'\}, T, S', P \cup \{S' \rightarrow S, S' \rightarrow \Lambda\},\rangle$ where $S' \notin N \cup T,$

which generates $L \cup \{\Lambda\}$.

The following result is important since it shows that the membership problem for context sensitive grammars is solvable.
Theorem

If $G = \langle N, T, S, P \rangle$ is a context sensitive grammar then $L(G)$ is a recursive language.

Note

- (Definition 1) A grammar is context-sensitive if all productions are of the form $x \to y$, where x, y are in $(\mathbf{N} \cup T)^+$ and $|x| \le |y|$.
- (Definition 2) A grammar is context-sensitive if all productions are of the form $xAy, \to xvy,$ where x,v, y are in $(\mathbf{N} \cup T)^*$, $A \in V$.

■ **Example 1.**
The language $\{ a^n b^n c^n \mid n \ge 1\}$ is context-sensitive but not context free.
A grammar for this language is given by:

$$\begin{aligned} S &\to aSBC \mid aBC \\ CB &\to BC \\ aB &\to ab \\ bB &\to bb \\ bC &\to bc \\ cC &\to cc \end{aligned}$$

Note that the left hand side of the productions are not all single non-terminals.Thus, context-sensitive. A derivation from this grammar is:-

$$\begin{aligned} S &\Rightarrow aSBC \\ &\Rightarrow aaBCBC && \text{(using } S \to aBC) \\ &\Rightarrow aabCBC && \text{(using } aB \to ab) \\ &\Rightarrow aabBCC && \text{(using } CB \to BC) \\ &\Rightarrow aabbCC && \text{(using } bB \to bb) \\ &\Rightarrow aabbcC && \text{(using } bC \to bc) \\ &\Rightarrow aabbcc && \text{(using } cC \to cc) \end{aligned}$$

which derives $a^2b^2c^2$.
Thus we can form a hierarchy of languages called the ***Chomsky hierarchy***. This consists of :

Type 0 – Phrase Structure
Type 1 – Context Sensitive
Type 2 – Context Free
Type 3 – Regular

Parse Trees and Ambiguity
A parse tree is a way to represent the derivation of a string from a grammar. A given string may have more than one parse tree. If every string in the language has exactly one parse tree then the grammar is called unambiguous. If there is even one string with two or more parse trees, then the grammar is called ambiguous.

What is considered a different parse tree? Two parse trees are distinct if they have a different structure, which means you cannot lay one on top of the other and have the symbols match up. An equivalent formulation of this is to say that there are two different leftmost (or rightmost) derivations of the string. Note it is not correct to say that two parse trees being distinct is the same as the string having two different variations. That is many different derivations can give the same parse tree, but every different leftmost derivation gives a unique parse tree.

Chomsky Normal Form

It is convenient to assume that every context-free grammar can without loss of generality be put into a special format, called a normal form. One such format is Chomsky Normal Form. In CNF, we expect every production to be of the from

$A \to BC$ or $D \to d$, where A, B, C and D are non-terminal symbols and d is a non-lambda terminal symbol. If lambda (the empty string) is actually part of the language, then $S \to \lambda$ is allowed.

We will use CNF in three different places:

1. A proof of a pumping lemma for CFG's.
2. A proof that every language generated by a CFG can be accepted by a non-deterministic pushdown machine.
3. An algorithm (dynamic programming style) for determining whether a given string is generated by a given context free grammar.

There are many steps needed to turn an arbitrary CFG into CNF. The steps are listed below:

1. Get rid of Useless symbols.
2. Get rid of lambda-productions.
3. Get rid of Unit Productions.
4. Get rid of Long Productions.
5. Get rid of Terminal symbols.

The steps are completely algorithmic with step one repeatable after each of the other steps if necessary. We will do a complete example in class.

Useless Symbols

Delete all productions containing non-terminal symbols that cannot generate terminal strings.

Delete all productions containing non-terminal symbols that cannot be reached by S.

The details of the two steps for finding useless symbols are very similar and each is a bottom-up style algorithm. To find all non-terminal symbols that generate terminal strings, we do it inductively starting with all non-terminal symbols that generate a single terminal string in one step. Call this set T. Then we iterate again looking for productions whose right sides are combinations of terminal symbols and non-terminal from T. The non-terminals on the left sides of these productions are added to T, and we repeat. This continues until T remains the same through an iteration.

To find all non-terminal symbols that can be reached by S, we do a similar thing but we start from S and check which non-terminals appear on the right side of its productions. Call this set T. Then we check which non-terminals appear on the right side of productions whose left side is a non-terminal in T. This continues until T remains the same through an iteration.

The steps need to be done in this order. For example, if you do it in the opposite order, then the grammar $S \to AB$, $S \to 0$, $A \to 0$, would result in the grammar $S \to 0$, $A \to 0$. If we do it in the correct order, then we get the right answer, namely just $S \to 0$.

Subsequent steps may introduce new useless symbols. Hence useless symbol removal can be done after each of the upcoming steps to ensure that we do not waste time carrying useless symbols forward.

Lambda-Production Removal

The basic idea is to find all non-terminals that can eventually produce a lambda (nullable non-terminals), and then take every production in the grammar and substitute lambdas for each subset of such non-terminals. We add all these new productions. We can then delete the actual lambda productions. For example, $A \to 0N1N0$ $N \to \lambda$. We add $A \to 0N10 \mid 01N0 \mid 010$, and delete $N \to \lambda$.

The problem with this strategy is that it must be done for all nullable non-terminals, simultaneously. If not, here is a problem scenario: $S \to 0 \mid X1 \mid 0Y0$ $X \to Y \mid \lambda$ $Y \to 1 \mid X$. In this case, when we try to substitute for $X \to \lambda$, we add $S \to 1$ and $Y \to \lambda$, and then we sub for $Y \to \lambda$, we add $S \to 00$ and $X \to \lambda$. The trick is to calculate all nullable non-terminals at the start which include X and Y, and then sub for all simultaneously, deleting all resulting lambda productions, except perhaps for $S \to \lambda$. If lambda is actually in the language, then we add a special start symbol $S' \to S \mid \lambda$, where S remains the old start symbol.

Unit Productions

To get rid of unit productions like $A \to B$, we simply add $A \to$ anything, for every production of the form $B \to$ anything, and then delete $A \to B$. The only problem with this is that B might itself have unit productions. Hence, like we did in lambda productions, we first calculate all Unit non-terminals that A can generate in one or more steps. Then the $A \to$ *anything* productions are added for all the Unit non-terminals X in the list, where $X \to$ anything, as long as *anything* is not

a Unit production. The original Unit productions are then deleted. The Unit non-terminals that can be generated by A can be computed in a straightforward bottom-up manner similar to what we did earlier for lambda productions.

For example, consider $S \to A \mid 11 \quad A \to B \mid 1 \quad B \to S \mid 0$. The Unit non-terminals that can be generated by S, A and B are A,B and B,S and A,S respectively. So we add $S \to 1$ and $S \to 0$; $A \to 11$ and $A \to 0$; and $B \to 1$ and $B \to 11$. Then we delete $S \to A$, $A \to B$ and $B \to S$.

Long Productions

Now that we have gotten rid of all length zero and length one productions, we need to concentrate on length > 2 productions. Let $A \to ABC$, then we simply replace this with $A \to XC$ and $X \to AB$, where X is a new non-terminal symbol. We can do this same trick inductively (recursively) for longer productions.

If we have long terminal productions or mixed terminal/non-terminal productions, then for each terminal symbol, say our alphabet is {0,1}, we add productions $M \to 0$ and $N \to 1$. Then all 0's and 1's are replaced by M's and N's, where M and N are new non-terminal symbols.

Pushdown Machines

A pushdown machine is just like a finite state machine, except is also has a single stack that it is allowed to manipulate. (Note, that adding two stacks to a finite state machine makes a machine that is as powerful as a Turing machine). PDM's can be deterministic or nondeterministic. The difference here is that the nondeterministic PDM's can do more than the deterministic PDM's.

In the previous sections, we explained how a DFSA is used to recognise (accept strings in) regular languages. Context-free languages also have a machine counterpart: the *pushdown automata* (PDA). To recognise context-free languages, we need to define a machine that solves the "memory problem" we noted above. The solution comes from adding a stack data structure to a finite-state machine.

To understand how the stack is used in conjunction with a finite-state machine, let us visualise a pushdown automaton for our example context-free language, $L(G) = \{a^n b^n \mid n \geq 1\}$. Let us define a machine with two states, as follows:

- When the machine is in q_0 : If an a is read, push a marker on the stack and stay in q_0; if a b is read and there is a marker on the stack, pop the stack and go to q_1.
- When the machine is in q_1: If a b is read and there is a marker on the stack, pop the stack and stay in q_1.
- Assume that all other transitions are undefined and cause the machine to halt, rejecting the input.
- The computation ends when both the input and the stack are empty.

Why does this machine accept $L(G) = \{a^n b^n \mid n \geq 1\}$? In the start state q_0, the only possible moves are a) read one or more a's, adding a marker to the stack for each a which is seen; b) read exactly one b, popping the stack and moving to q_1. Assuming we have read n a's and 1 b, then there will be n – 1 markers left on the stack. In state q_1, the only possible move is to read a b and pop the stack. Since the machine will halt (and reject) if there is input remaining and the stack is empty, the only way to exhaust the input and end with an empty stack is to read exactly the same number of a's and b's.

A pushdown automaton can be formally defined $M = (Q, \Sigma, \Gamma, \partial, q_0, F)$:

Q, a finite set of states

Σ, the alphabet of input symbols

Γ, the alphabet of stack symbols

∂, $Q \times \Sigma \times \Gamma \to Q \times \Gamma$

q_0, the initial state

F, the set of final states

Intuitively, if $\partial(q, s, \beta) = (q', \gamma)$, then M, whenever it is in state q with β at the top of the stack, may read s from the input, replace β by γ on the top of the stack, and enter state q'.

Pushing, popping, and preserving the stack are possible:

$\partial(q,a,\varepsilon) = (q',A)$ push A on the stack without popping

$\partial(q,a,A) = (q',\varepsilon)$ pop A from the stack without pushing

$\partial(q,a,\varepsilon) = (q',\varepsilon)$ stack unchanged

Now we can define a PDA M, such that $L(M) = L(G) = \{a^n b^n \mid n \geq 1\}$:

$M = (\{q_0, q_1\},\{a, b\},\{A\}, \partial, q_0, \{q_1\})$

$\partial(q_0,a,\varepsilon) = (q_0, A)$

$\partial(q_0,b,A) = (q_1, \varepsilon)$

$\partial(q_1,b,A) = (q_1, \varepsilon)$

(Self-test: Trace the operation of M on some strings in L(G), and some strings not in L(G). Assume computation is successful (accept) only if the input is empty, the stack is empty, and the machine is in final state q_1.)

Closure Properties

Deterministic PDMs have different closure properties than non-deterministic PDM's. Deterministic machines are closed under complement, but not under union, intersection or reverse. Non-deterministic machines are closed under union and reverse but not under complement or intersection. They are however closed under intersection with regular sets. We will discuss all these results in detail in class.

Non-CFL's

Not every language is a CFL, and there are NPDM's which accept languages that no PDM can accept. An example of the former is 0^n1^n0^n, and and example of the latter id the union of $0^n 1^n$ and $0^n 1^{2n}$.

Every CFL has a Non-deterministic PDM that Accepts it

One side of this equivalence is easier than the other. We omit the proof of the harder direction. The easier direction shows that every CFG G has an NPDM M where M accepts the language generated by G. The proof uses Chomsky Normal Form.

Without loss of generality, assume that G is given in CNF. We construct an NPDM M with three main states, an initial, a final and a *simulating* state. The initial state has one arrow coming out of it labeled lambda, lambda, SZ. All the rest of the arrows come out of the simulating state and all but one loops back to the simulating state. A detailed example can be give by using the grammar $S \rightarrow AB \quad A \rightarrow BB \mid 0 \quad B \rightarrow AB \mid 1 \mid 0$.

The idea of the proof is that the machine will simulate the grammar's attempt at generating a leftmost derivation string. The stack holds the current state of the derivation, namely the non-terminals that remain. At any point in the derivation, we must guess which production to use to substitute for the current leftmost symbol, which is sitting on top of the stack. If the production is a double non-terminal then that symbol is popped off the stack and the two new symbols pushed on in its place. If it is a single terminal, then a symbol on the tape is read, while we just pop the symbol off the stack.

CYK Algorithm

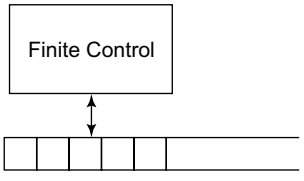
The most important decision algorithm about CFG's is given a string x and a CFG G, is x generated by G? This is equivalent to given a text file, and a Java compiler, is the file a syntactically correct Java program?

The details of such an algorithm is essentially one third of a compiler, called parsing.

One way to solve this problem is to turn the CFG into CNF and then generate all parse trees of height 2n-1 or less, where n is the length of x. (In a CNF grammar, all strings of length n are generated by a sequence of exactly 2n-1 productions). We check to see if any of these trees actually derive x. This method is an algorithm but runs in exponential time.

3.1.8 Turing Machine

The basic model of a Turing machine has a finite control, an input tape that is divided into cells, and a tape head that scans one cell of the tape at a time. The tape has a leftmost cell but is infinite to the right. Each cell of the tape may hold exactly one of a finite number of tape symbols. Initially, the n leftmost cells, for some finite n >= 0, hold the input, which is a string of symbols chosen from a subset of the tape symbols called the input symbols. The remaining infinity of cells each hold the blank, which is a special symbol that is not an input symbol.



A Turing machine can be formally defined as M = (Q, Σ, Γ, ∂, q_0, B, F):, where

Q, a finite set of states

Γ, is the finite set of allowable *tape symbols*

B, a symbol of Γ, is the *blank*

Σ, a subset of Γ not including B, is the set of input symbols

∂: Q × Γ → Q × Γ × {L, R} (∂ may, however, be undefined for some arguments)

q_0 in Q is the initial state

F ⊆ Q is the set of final states

■ **Example 1.** The design of a Turing Machine M to accept the language L = {0^n1^n, n >= 1} is given below. Initially, the tape of M contains 0^n1^n followed by an infinity of blanks. Repeatedly, M replaces the leftmost 0 by X, moves right to the leftmost 1, replacing it by Y, moves left to find the rightmost X, then moves one cell right to the leftmost 0 and repeats the cycle. If, however, when searching for a 1, M finds a blank instead, then M halts without accepting. If, after changing a 1 to a Y, M finds no more 0's, then M checks that no more 1's remain, accepting if there are none.

Let Q = { q_0, q_1, q_2, q_3, q_4 },

Σ = {0,1}, Γ = {0,1,X,Y,B} and F = {q_4}

∂ is defined with the following table:

	INPUT SYMBOL				
State	0	1	X	Y	B
q0	(q1,X,R)	—		(q3,Y,R)	—
q1	(q1,0,R)	(q2,Y,L)	—	(q1,Y,R)	—
q2	(q2,0,L)	—	(q0,X,R)	(q2,Y,L)	—
q3	—	—	—	(q3,Y,R)	(q4,B,R)
q4	—	—	—	—	—

The Turing Machine as a computer of integer functions

In addition to being a language acceptor, the Turing machine may be viewed as a computer of functions from integers to integers. The traditional approach is to represent integers in unary; the integer i >= 0 is represented by the string 0^i. If a function has more than one argument then the arguments may be placed on the tape separated by 1's.

For example, proper subtraction m – n is defined to be m – n for m >= n, and zero for m < n. The TM

M = ({q0,q1,...,q6}, {0,1}, {0,1,B}, ∂, q0, B, {})

defined below, if started with 0^m10^n on its tape, halts with 0^{m-n} on its tape. M repeatedly replaces its leading 0 by blank, then searches right for a 1 followed by a 0 and changes the 0 to a 1. Next, M moves left until it encounters a blank and then repeats the cycle. The repetition ends if

(i) Searching right for a 0, M encounters a blank. Then, the n 0's in 0^m10^n have all been changed to 1's, and n+1 of the m 0's have been changed to B. M replaces the n+1 1's by a 0 and n B's, leaving m-n 0's on its tape.

(ii) Beginning the cycle, M cannot find a 0 to change to a blank, because the first m 0's already have been changed. Then n >= m, so m – n = 0. M replaces all remaing 1's and 0's by B.

The function ∂ is described below.

∂(q0,0) = (q1,B,R) Begin. Replace the leading 0 by B.

∂(q1,0) = (q1,0,R) Search right looking for the first 1.

∂(q1,1) = (q2,1,R)

∂(q2,1) = (q2,1,R) Search right past 1's until encountering a 0. Change that 0 to 1.

∂(q2,0) = (q3,1,L)

∂(q3,0) = (q3,0,L) Move left to a blank. Enter state q0 to repeat the cycle.

∂(q3,1) = (q3,1,L)

∂(q3,B) = (q0,B,R) If in state q2 a B is encountered before a 0, we have situation i described above. Enter state q4 and move left, changing all 1's to B's until encountering a B. This B is changed back to a 0, state q6 is entered and M halts.

∂(q2,B) = (q4,B,L)

∂(q4,1) = (q4,B,L)

∂(q4,0) = (q4,0,L)

∂(q4,B) = (q6,0,R) If in state q0 a 1 is encountered instead of a 0, the first block of 0's has been exhausted, as in situation (ii) above. M enters state q5 to erase the rest of the tape, then enters q6 and halts.

∂(q0,1) = (q5,B,R)

∂(q5,0) = (q5,B,R)

∂(q5,1) = (q5,B,R)

∂(q5,B) = (q6,B,R)

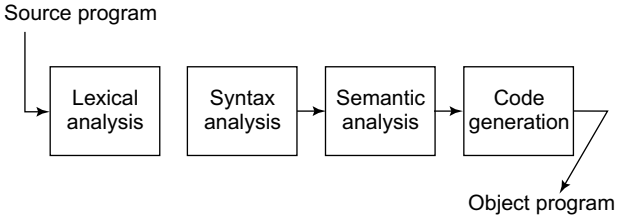
3.2 Compiler Design

Three phases of the compilation process:

(1) Lexical analysis: reads high-level source program and divides it into a stream of basic lexical 'tokens'.

(2) Syntax and semantic analysis phase: then combines these tokens into data structures reflecting the form of the source program in terms of the syntactic structures of the language.

(3) Code generator: converts these data structures into code for the target machine.



Task of a compiler:

1. the analysis of the source program (lexical syntax and semantic analysis)
2. the synthesis of the object program (a single code-generation phase)

3.2.1 Lexical Analysis

Reads the characters of the source program and recognises the tokens (lexemes) or basic syntactic components that they represent. It is able to distinguish and pass on, as single units, objects such as:

Numbers,

Punctuation symbols,

Operators,

Reserved keywords,

Identifiers and so on.

Effect: simplifies the syntax analyser → effectively reducing the size of the grammar the syntax analyser has to handle.

In a free-formal language, the lexical analyser ignores spaces, newlines, tabs and other layout characteristics, as well as comments.

e.g., for I := 1 to 10 do sum := sum + term[i]; (*sum array*)

will be transformed by the lexical analyser into the sequence of tokens:

```
for    i    :=    1    to    10    do    sum
  :=  sum  +   term  [i];
```

Pascal, maintains a list of reserved words so that they can be distinguished from identifiers and passed to the next phase of the compiler in the form of, for example, a short integer code (or, use a symbol table).
In a nutshell,

- A lexical analyser is a pattern matcher for character strings
- A lexical analyser is a "front-end" for the parser
- Identifies substrings of the source program that belong together – lexemes
  - Lexemes match a character pattern, which is associated with a lexical category called a token.
- The lexical analyser is usually a function that is called by the parser when it needs the next token

There are three approaches to building a lexical analyser:

1. Write a formal description of the tokens and use a software tool that constructs table-driven lexical analysers given such a description
2. Design a state diagram that describes the tokens and write a program that implements the state diagram
3. Design a state diagram that describes the tokens and hand-construct a table-driven implementation of the state diagram

The second approach is our subject of study: State diagram design

- A naïve state diagram would have a transition from every state on every character in the source language – such a diagram would be very large.
- In many cases, transitions can be combined to simplify the state diagram.
- When recognising an identifier, all uppercase and lowercase letters are equivalent – Use a character class that includes all letters.
- When recognising an integer literal, all digits are equivalent – use a digit class.
- Reserved words and identifiers can be recognised together (rather than having a part of the diagram for each reserved word).

  (Use a table lookup to determine whether a possible identifier is in fact a reserved word.)

Convenient utility subprograms:

1. **getChar** – gets the next character of input, puts it in nextChar, determines its class and puts the class in charClass
2. **addChar –** puts the character from nextChar into the place the lexeme is being accumulated, lexeme
3. **lookup** – determines whether the string in lexeme is a reserved word (returns a code)

Reasons to separate Lexical Analysis from Syntax Analysis:

1. **Simplicity:** Techniques for Lexical Analysis can be simpler than those required for Syntax Analysis. DFA vs. PDA. Separation also simplifies the Syntax Analyser.
2. **Efficiency:** Separation into different modules makes it easier to perform simplifications and optimisations unique to the different paradigms.
3. **Portability:** Due to input/output and character set variations, Lexical Analysers are not always machine independent.

Errors often detected in a Lexical Analyser:

1. Numeric literals that are too long.
2. Identifiers that are too long (often a warning is given)
3. Ill-formed numeric literals.
4. Input characters that are not in the source language

Input Buffering

Moving input data into local memory can increase the performance.

Double buffering

Use the sentinel, "@" to identify the end of a buffer. If the end of buffer is found, increment to the next buffer and re-fill the prior buffer. The sentinel enables only one check to be made, then a second check to know which buffer needs to be refilled. Otherwise it would be necessary to check for the end of either buffer for each character read. Also consider the event if the sentinel is an invalid character in the source code.

Buffering allows for easy look ahead of characters.

The look ahead will allow for the identification of a less than token before getting the next character from the buffer.

Example of a Lexical Analyser

Build a Lexical Analyser which identifies the following tokens:

1. digits
2. digits E [sign] digits
3. digits.digits [E [sign] digits]

"sign" refers to +, –

"E" refers to exponent by power 10

[ ] refers to 0 or 1 of contents

{ } refers to 0 or more of contents

ε = λ

The above tokens can be accepted by the following DFA:

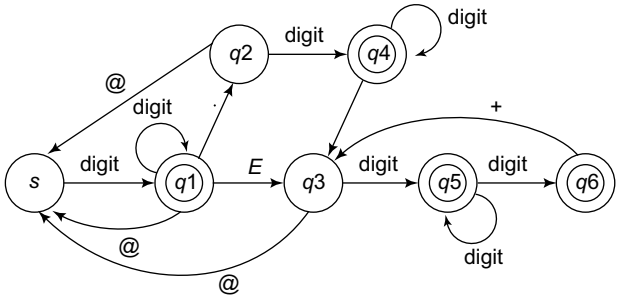


Table 3.1 illustrates the DFA state information

	S	q1	q2	q3	q4	q5	q6
Digit	q1	q1	q4	q5	q4	q5	q5
.	er	q2	er	er	S	S	er
E	er	q3	er	er	q3	S	er
+, –	er	S	er	q6	S	S	er
ά	er	S	er	er	S	S	er
token array	0	int	0	0	float	float	0

Figure 3.2 illustrates the relation between lexical analyser and syntax analyser.

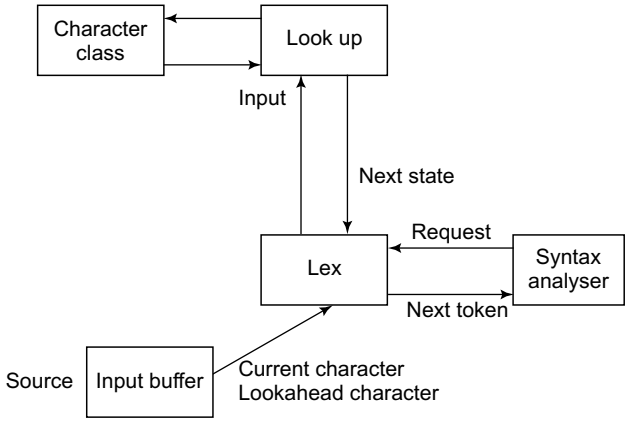


Figure 3.2

The String Table

The String Table is a data structure for unique lexemes. The String Table can be queried for the lexeme by a numeric code. Lexemes are generally inserted in the String Table by the Lexical Analyser.

The String Table can be used for:

1. Error messages
2. Memory map listings
3. Intermodule linkages

Index	String	Token
0	fptr	[ident]
1	number	[ident]
2	5.1	[f_lit]
3		

It is not recommended to use the String Table for reserved words. Use a separate structure to determine if an identifier is a reserved word, and then assign a token value accordingly. If it is not a reserved word, insert it in the String Table if necessary.

Do not confuse the String Table with the Symbol Table. The String Table is for unique spellings of lexemes, allowing index comparisons instead of string comparisons during syntax analysis and code generation. The Symbol Table is a dynamic data structure used differently than the String Table.

3.2.2 Syntax Analysis

The syntax analyser or parser has to determine how the tokens retuned by the lexical analyser should be grouped and structured according to the syntax rules of the language.

Output: representation of the syntactic structure often expressed in the form of a tree (the 'parse tree').

Usually, lexical analyser should be responsible for all the simple syntactic constructs, such as identifiers, reserved words and numbers, while the syntax analyser should deal with all the other structures.

Semantic Analysis

To determine the semantics or meaning of the source program (the translation phase) may cope with tasks involving declarations and scopes of identifiers, storage allocation, type checking, selection of appropriate polymorphic operators, addition of automatic type transfers, etc.

This phase is often followed be a process that takes the parse tree from the syntax analyser and produces a linear sequence of instructions equivalent to the original source program (instructions for a virtual machine).

Code Generation

(The final phase) to take the output from the semantic analyser or translator and output machine code or assembly language for the target hardware (machine's architecture is required to write good code generator).
+ code improvement or code optimisation.

Syntax Specification

Role: to define the set of valid programs.
Sets
e.g., a digit in Pascal could be defined as {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0}
or, $\{xy^n \mid n>0\}$ xy, xyy, xyyy, …, xy^n
i.e., a string that starts with a single x followed by any number (greater than zero) of y_s.

Backus-Naur Form (BNF)

A formal meta language that is frequently used in the definition of the syntax of programming languages (introduced in the definition of ALGOL 60).

A technique for representing rules that can be used to derive sentences of the language.

(If a finite set of these rules can be used to derive all sentences of a language, then this set of rules constitutes a formal definition of the syntax of the language).

■ **Example**

<sentence> → <subject> <predicate>
<subject> → <noun> | <pronoun>
<predicate> → <transitive verb> <object> | <intransitive verb>
<noun> → cats | dogs | sheep
<pronoun> → I | we| you | they
<transitive verb> → like | hate | eat
<object> → biscuits | grass | sunshine
<intransitive verb> → sleep | talk | run

Using complete set of rules, sentences can be generated by making random choices.
e.g. <sentence>
<subject> <predicate>
<noun> <predicate>
sheep <transitive verb> <object>
sheep eat <object>
sheep eat biscuits

Other, possible sentences in this language are:
I sleep,
Dogs hate grass,
We like sunshine

Note

No meaning of these sentences considered e.g. "cats eat sunshine" is syntactically correct, makes no good sense, no concern to the BNF rules.

Two distinct symbol types in BNF:

1. Symbols such as <sentence> <pronoun> and <intransitive verb> are called non-terminal symbols (i.e. appear on the left-hand of a BNF).
2. Symbols such as cats, dogs, I and grass are called terminal symbols, since they cannot be expanded further (the set of symbols of the language being defined).

Non-terminals are enclosed by angle brackets.

Other convention

- Non-terminals are either enclosed by angle brackets or are single upper-case letters.
- Terminals are represented as single lower-case letters, digits, special symbols (such as +, * or =), punctuation symbols or stings in bold type (such as begin).

Example related to another language

Set of rules: S → S + T | T
T → a | bt

S and T are non-terminals, whereas
a, b and + are terminals
(S is being defined recursively).

S
S + T (using S → S + T)
S + T + T (using S → S + T)
T + T + T (using S → T)
b + T + T (using T → b)
b + a + T (using T → a)
b + a + a (using T → a)

For example,

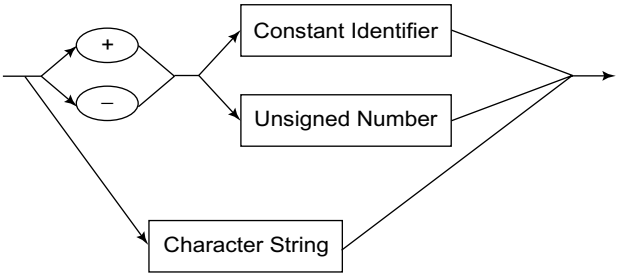
<expression> → <term> | <expression> + <term>
<term> → <primary> | <term> * <primary>
<primary> → a | b | c
a + b * c is generated as follows:
<expression>
<expression> + <term>
<term> + <term>
<primary> + <term>
a + <term>
a + <term> * <primary>
a + <primary> * <primary>
a + b * <primary>
a + b * c
or, a + b + c is generated as follows:

Syntax diagrams

Pictorial notation

A set of syntax diagrams, each defining a specific language construct.

e.g., the definition of a constant (in Pascal)



i.e., rectangular box – non-terminal symbol
terminal symbols, '+' & '–' enclosed in circles
(any path yields a syntactically correct structure)
Compact specification (e.g., Pascal in only a couple of sheets of paper).

EBNF (Extended Backus-Naur Form)

- used in the ISO Pascal Standard
- differs from BNF in several ways
- principal changes: the set of metasymbols (symbols used for special purposes in the metalanguage)

i.e.

- terminal symbols enclosed in double quotation marks
- a full stop is used to end each production

- equals sign is used to separate the non-terminal from its definition
- parenthesis for grouping

e.g.
AssignmentStatement = (Variable | FunctionalIdentifier) ":=" Expression

For repetition: [x] implies zero or more instance of x (i.e. x optional)

{x} implies zero or more instances of x

e.g. Identifier = Letter {Letter | Digit}

i.e. Simpler definition of syntax. Much Clearer.

The same in BNF could be:

<Identifier> ::= <letter> | <identifier> <letter>
| <identifier> <digit>

Grammars

Study of grammars started long before programming languages and was exposed primarily on the study of natural languages.

Then, found direct relevance in the formal study of programming languages.

The grammar – formally defined as a 4-tuple

G = (N, T, S, P)

N – the finite set of non-terminal symbols

T – the finite set of terminal symbols

S – the starting symbol (must be a member of N)

P – the set of productions (general form: $\alpha \rightarrow \beta$)

i.e., any occurrence of the string α in the string to be transformed can be replaced by the string β.
e.g., the set of BNF productions presented in example A above forms a part of the definition of the grammar of a language. The remainder of the definition is:

N = {sentence, subject, predicate, noun, pronoun, intransitive verb, transitive verb, object}

T = {cats, dogs, sheep, I, we, you, they, live, hate, eat, biscuits, grass, sunshine, sleep, talk, run}

S = sentence

Still, the definition of the grammar is not quite complete, the strings α and β must have some relationship to the sets N and T.

Suppose U is the set of all terminals and non-terminal symbols of the language; that is, U = N U T

U+ – the closure of U – non-empty strings

U* – the closure of U i.e. U+ U {ε} – empty string

Sentential form: any string that can be derived from the starting symbol.
Sentence: a sentential form that does not contain any non-terminal symbols (just terminals, no expansion).

We have seen how sentences can be generated very simply using a set of BNF productions.

The reverse, how the BNF rules were applied to generate a sentence is much harder. This process of determining the syntactic structure of a sentence is called parsing, or syntax analysis (major part of the compilation of H.l.L. programs).

Chomsky Classification

A grammar with general form
$\alpha \rightarrow \beta$ (no restrictions on the sentential forms α & β)
is called a Chomsky type 0 or a free grammar (too general)

Restricted form

$\alpha A\beta \rightarrow \alpha\gamma\beta$ where α, β and γ are members of U^*

γ is not null

A is a single non-terminal symbol

then the resulting grammar is of type 1, the context-sensitive grammars.

Especially, if $\alpha \rightarrow \beta$

Where $|\alpha| <= |\beta|$ and $|\alpha|$ denotes the length of the string α, then the grammar is context sensitive.

(A is transformed to γ only when it occurs in the context of being preceded by α and followed by β).

Type 2 or Context-free grammars if $A \rightarrow \gamma$ where A is a single non-terminal (since A can always be transformed into γ without any concern for its context.

(Immense importance in programming language design – corresponds directly to the BNF notation, where each production has a single non-terminal symbol on its left-hand side, and so any grammar that is expressed in BNF must be a context-free grammar).

e.g., Pascal and ALGOL 60 – context-free or type 2

Type 3 or finite, finite-state or regular grammars if all productions are of the form:

$A \rightarrow \alpha$ or $A \rightarrow \alpha B$ where A and B – non-terminal α is a terminal symbol (too restricted)

only for the design some of the structures used as components of most programming languages, e.g. identifiers, numbers, etc.

That is, hierarchically, it is shown in figure 3.3.

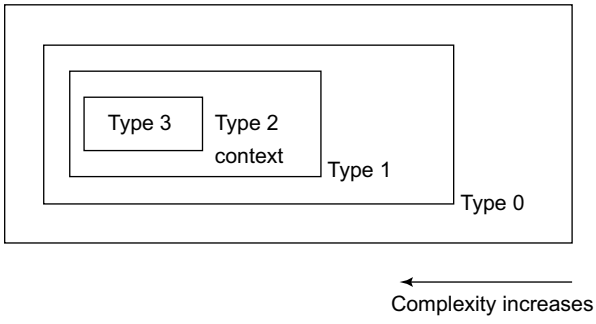


Figure 3.3

e.g., all type 3 languages are also type 1 languages

In processing from type 0 to type 3 grammars, language complexity and hence complexity of recognizers, parsers or compilers decreases.

Type 3 – easier – cause of finite-state automata

3.2.3 Semantics

Semantic rules specify the meanings or actions of all valid programs (much more difficult techniques for semantic specification than for syntax specification, not so well developed yet).

Compiler is concerned with two processes:

1. the analysis of the source program (concern of syntax)
2. the synthesis of the object program.

Syntax is largely concerned with the analysis phase.

Semantics is largely concerned with the synthesis phase (+ sometimes syntax)

Specification of semantics:

- operational approach
- denotational semantics
- axiomatic approach

Parsing

e.g., <expression> → <term> | <expression> + <term>
<term> → primary | <term> * <primary>
<primary> → a | b | c

to generate expressions such as a * b + c.

However, the compiler has to reserve this process, that is, perform a syntax analysis of the string, to determine if a string such as a * b + c is a valid expression and, if so, how it is structured in terms of the units <term> and <primary>.

The parse tree

Example (how the string a * b + c may be reduced)
Given the three productions defining <expression>, <term> and <primary>, the string a * b + c can be reduced as:

a * b + c	
<primary> * b + c	(<primary> → a)
<primary> * <primary> + c	(<primary> → b)
<primary> * <primary> + <primary>	(<primary> → c)
<term> * <primary> + <primary>	(<term> → <primary>)
<term> * <primary> + <term>	(<term> → <primary>)
<term> + <term>	(<term> → <term> * <primary>)
<expression> + <term>	(<expression> → <term>)
<expression>	(<expression> → <expression> + <term>)

However, if the productions are used differently,

a * b + c	
<primary> * b + c	(<primary> → a)
<primary> * <primary> + c	(<primary> → b)
<primary> * <primary> + <primary>	(<primary> → c)
<primary> * <term> + <primary>	(<term> → <primary>)
<primary> * <expression> + <primary>	(<expression> → <term>)
<primary> * <expression> + <term>	(<term> → <primary>)
<primary> * <expression>	(<expression> → <expression> + <term>)
<term> * <expression>	(<term> → <primary>)
<expression> * <expression>	(<expression> → <term>)

and then become stuck, implying the false deduction,
i.e. a * b + c not a sentence.
→ parsing process is not a trivial matter.
Syntactic structure of the string a * b + c is shown in figure 3.4.

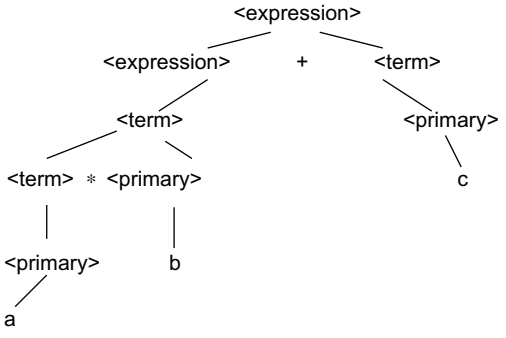


Figure 3.4

The tree combines the relevant information contained in the set of productions, together with the content of the original sentence, in a structure that is self-contained and which can be used by subsequent phases of the compiler.

Parsing strategies

Problem of parsing: Take the starting symbol of the language and start generate all possible sentences from it.

If the input matched one of these sentences, then all the input string is a valid sentence of the language (impractical approach since infinite possible sentences).

Two categories:

1. Top-down parsers: starting at the root (the starting symbol) and proceeding to the leaves.
2. Bottom-up parsers: start at the leaves and move up towards the root.

Top-down parsers – easy to write, actual code capable of being derived directly from the production rules, but cannot always applied as an approach.
→ bottom-up parsers, can handle a larger set of grammars.

Top-down parsing

The parsing process starts at the root of the parse tree; it first considers the starting symbol of the grammar.
The goal is to produce, from this starting symbol, the sequence of terminal symbols that have been presented as input to the parser.

Bottom-up parsing
Starts with the input string and repeatedly replaces strings on the right-hand sides of productions by the corresponding strings on the left-hand sides of productions, until, hopefully, just the starting symbol remains.

Necessary to determine which strings should be replaced and in what order the replacement should occur.
Process of derivation (leftmost, rightmost).

Leftmost derivation of a * b + c from	**Rightmost derivation**
expression	
<expression>	<expression>
<expression> + <term>	<expression> + <term>
<term> + <term>	<expression> + <primary>
<term> + <primary> + <term>	<expression> + c
<primary> + <primary> + <term>	<term> + c
a * <primary> + <term>	<term> * <primary> + c
a * b + <term>	<term> * b + c
a * b + <primary>	<primary> * b + c
a * b + c	a * b + c

Handle: the substring that is reduced, replaced by the left-hand side of the corresponding production.

■ **Example** Parse of a * b + c

Sentential form	**Handle**	**Production used**	**Reduced sentential form**
a * b + c	a	<primary> → a	<primary> * b + c
<primary> * b + c	<primary>	<term> → <primary>	<term> * b + c
<term> * b + c	b	<primary> → b	<term> * <primary> + c
<term> * <primary> + c	<term> * <primary>	<term> → <term> * <primary>	<term> + c
<term> + c	<term>	<expression>→ <term>	<expression> + c

Sentential form	Handle	Production used	Reduced sentential form
<expression> + c	c	<primary> → c	<expression> + <primary>
<expression> + <primary>	<primary>	<term> → <primary>	<expression> + <term>
<expression> + <term>	<expression> + <term>	<expression> → <expression> + <term>	<expression>
<expression>			

i.e., the canonical parse (canonical derivation)

In a nutshell,

- Goals of the parser when given an input program are:
  - Find all syntax errors; for each, produce an appropriate diagnostic message, and recover quickly
  - Produce the parse tree, or at least a trace of the parse tree, for the program
- Two categories of parsers are:
  1. Top-down parsers
  - Given a sentential form, xAα , the parser must choose the correct A-rule to get the next sentential form in the leftmost derivation, using only the first token produced by A
  - The most common top-down parsing algorithms:
    - Recursive descent - a coded implementation
    - LL parsers - table driven implementation
  2. Bottom-up parsers
  - Given a right sentential form, α, what substring of α is the right-hand side of the rule in the grammar that must be reduced to produce the previous sentential form in the right derivation.
  - The most common bottom-up parsing algorithms are in the LR family (LALR, SLR, canonical LR)
  3. The Complexity of Parsing.
  - Parsers that works for any unambiguous grammar are complex and inefficient ($O(n^3)$, where n is the length of the input).
  - Compilers use parsers that only work for a subset of all unambiguous grammars, but do it in linear time (O(n), where n is the length of the input).
- Parsers look only one token ahead in the input.

3.2.4 Intermediate Code

Need for Intermediate Code Generation:

(1) Suppose we have n-source languages and m-Target languages. Without Intermediate code we will change each source language into target language directly. So, for each source-target pair we will need a compiler. Hence, we need (n*m) compilers, one for each pair. If we use Intermediate code, we need n-compilers to convert each source language into intermediate code and m-compilers to convert intermediate code into m-target languages. Thus, we need only (n+m) compilers.

(2) Re-targetting is facilitated; a compiler for a different machine can be created by attaching a back-end(which generate Target Code) for the new machine to an existing Front-end (which generate Intermediate Code).

(3) A machine Independent Code-Optimiser can be applied to the Intermediate code.

(4) Intermediate code is simple enough to be easily converted to any target code.

(5) Complex enough to represent all the complex structure of high level language.

Code Generator

A code generator produces an object program when an input program is given. This is the final phase of compilation. It takes as input the intermediate representation (IR) produced by the front end of the compiler, along with relevant symbol table information, and produces as output a semantically equivalent target program.

Compilers that need to produce efficient target programs, include an optimisation phase prior to code generation. The optimiser maps the IR into IR from which more efficient code can be generated. In general, the code optimisation and code-generation phases of a compiler, often referred to as the *back* end, may make multiple passes over the IR before generating the target program.

An intermediate language should, ideally, have the following properties:

- It should be easy to translate from a high-level language to the intermediate language. This should be the case for a wide range of different source languages.
- It should be easy to translate from the intermediate language to machine code. This should be true for a wide range of different target architectures.
- The intermediate format should be suitable for optimisations.

While translating a source program into a functionally equivalent object code representation, a parser may first generate an intermediate representation. This makes retargeting of the code possible and allows some optimisations to be carried out that would otherwise not be possible. The following are commonly used intermediate representations:

1. Syntax tree
2. Postfix notation
3. Three-address code

Syntax Tree

The syntax tree is nothing more than a condensed form of the parse tree. The operator and keyword nodes of the parse tree (figure 3.4 (a)) are moved to their parent, and a chain of single productions is replaced by single link (figure 3.5 (b)). Syntax tree depicts the natural hierarchical structure of a source program.

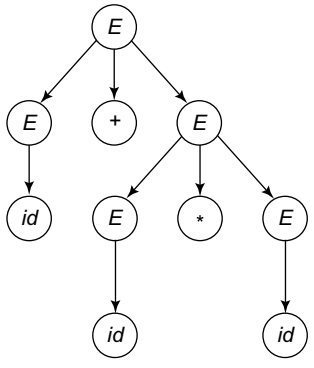


Figure 3.5 (a) Parse tree for id + id *id

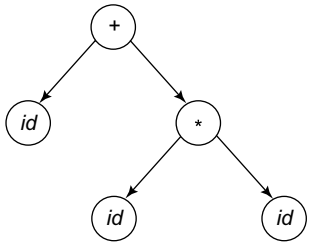
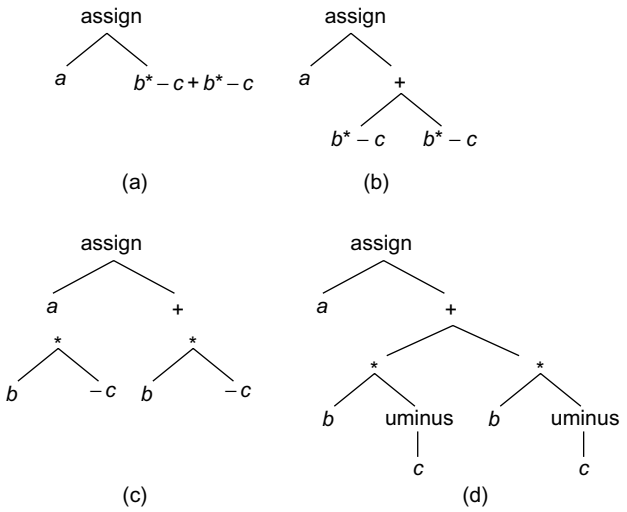


Figure 3.5 (b) Syntax tree for id + id *id

■ **Example 1.** Consider the following assignment statement

a := b * -c + b * -c;

The following figure explains the syntax tree for the given expression.



Postfix Notation

Its nature allows it to be evaluated with the use of a stack, Operands are pushed onto the stack; operators pop the right amount of operands from the stack, do the operation, then push the result back onto the stack. However, this notation is restricted to simple expressions such as in arithmetic where every rule conveys an operation. It cannot be used for the expression of most programming languages constructs.

■ **Example 2.**

a+b ⇒ ab+

a+b*c ⇒ abc*+

Three-Address Code

Three address code is a sequence of statements of the form $x = y$ op z. Since a statement involves no more than three references, it is called a "three-address statement," and a sequence of such statements is referred to as three-address code. For example, the three-address code for the expression $a + b * c + d$ is:

T1 = B*C

T2 = A+T1

T3 = T2+D

Three-address code (TAC) will be the intermediate representation. It is essentially a generic assembly language that falls in the lower-end of the mid-level IRs. Some variant of 2, 3 or 4 address code is fairly commonly used as an IR, since it maps well to most assembly languages.

A TAC instruction can have at most three operands. The operands could be two operands to a binary arithmetic operator and the third the result location, or an operand to compare to zero and a second location to branch to, and so on. For example, below on the left is an arithmetic expression and on the right, is a translation into TAC instructions:

a = b * c + b * d;

t1 = b * c;

t2 = b * d;

t3 = t1 +t2;

a = t3;

■ **Example 3.** An example c program and its IR code.

```
void main ()                    main;
{                                 BeginFune     ;
    int b;                        _t0 = 3;
    int a;                        b = _ t0;
```

```
    b = 3;                          _t1 = 12;
    a = 12;                         a = _t1;
    a = (b + 2) -(a*3)/6;           _t2 = 2;
}                                   _t3 = b + _ t2;
                                    _t4 = 3;
                                    _t5 = a* _t4;
                                    _t6 = 6;
                                    _t7 = _t5/ _t6;
                                    _t8 = _t3 - _t7;
                                    a = _t8;
                                    End Funo;
```

Implementation of Three Address Statements

TAC is an abstract form of intermediate code. In a compiler, these statements can be implemented as records with fields for the operator and the operands. It can be done in the form of Quadruples, Triples, and Indirect Triples

Quadruples

A quadruple is a record structure of four fields

- operator
- argument 1
- argument 2
- result

The contents of fields arg1, arg2 and result are pointers to the symbol table entries for the names represented by these fields, therefore, temporary names must be added in the symbol table.

x – 2 * y

(1)	load	t1	y	
(2)	loadi	t2	2	
(3)	mult	t3	t2	t1
(4)	load	t4	x	
(5)	sub	t5	t4	t3

Triples

x – 2 * y

(1)	load	y	
(2)	loadi	2	
(3)	mult	(1)	(2)
(4)	load	x	
(5)	sub	(4)	(3)

Indirect Triples

Another implementation of three-address code that has been considered is that of listing pointers to triples, rather than listing the triples themselves. This implementation is naturally called indirect triples. For example, let us use an array statement to list pointers to triples in the desired order.

x – 2 * y

	stmt	op	arg1	arg2
(1)	(100)	load	y	
(2)	(101)	loadi	2	
(3)	(102)	mult	(100)	(101)
(4)	(103)	load	x	
(5)	(104)	sub	(103)	(102)

Problems in Code Generation / Issues in the Design of a Code Generator:
The following are the assumptions regarding code generator:

1. The code generator is given the intermediate text in any one of its form.
2. But in specific algorithms, we deal with quadruples or in some case parse trees as the intermediate forms.
3. Necessary semantic checking is done.
4. The data areas and offsets have been determined for each name that information is available from the symbol table.

A code generator has three primary tasks: instruction selection, register allocation and assignment, and instruction ordering. Figure 3.6 illustrates position of code generator.

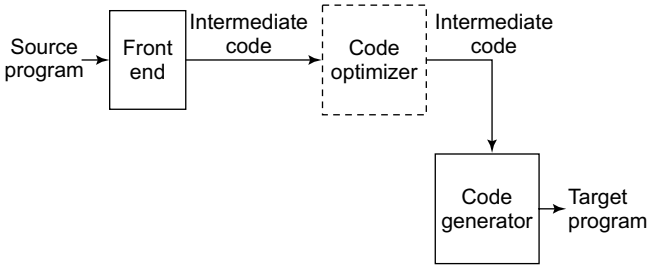


Figure 3.6 Position of Code Generator

Instruction selection involves choosing appropriate target-machine instructions to implement the IR statements. Register allocation and assignment involves deciding what values to keep in which registers. Instruction ordering involves deciding in what order to schedule the execution of instructions.

Many code generators partition IR instructions into *"basic blocks"*, which consist of sequences of instructions that are always executed together.

The Target Program
The instruction-set architecture of the target machine has a significant impact on the difficulty of constructing a good code generator that produces high-quality machine code.

An RISC machine typically has many registers, three-address instructions, simple addressing modes, and a relatively simple instruction-set architecture. In contrast, a CISC machine typically has few registers, two-address instructions, a variety of addressing modes, several register classes, variable-length instructions, and instructions with side effects.

In a stack-based machine, operations are done by pushing operands onto a stack and then performing the operations on the operands at the top of the stack. To achieve high performance the top of the stack is typically kept in registers. Stack-based machines almost disappeared because it was felt that the stack organisation was too limiting and required too many swap and copy operations.

Instruction Selection
The code generator must map the IR program into a code sequence that can be executed by the target machine. The complexity of performing this mapping is determined by factors such as:

- ✓ *The level of the IR.*
- ✓ *The nature of the instruction-set architecture.*
- ✓ *The desired quality of the generated code.*

If the IR is high level, the code generator may translate each IR statement into a sequence of machine instructions using code templates. Such statement-by-statement code generation, however, often produces poor code that needs further optimisation. If the IR reflects some of the low-level details of the underlying machine, then the code generator can use this information to generate more efficient code sequences.

The nature of the instruction set of the target machine has a strong effect on the difficulty of instruction selection. For example, the uniformity and completeness of the instruction set are important factors. If the target machine does not support each data type in a uniform manner, then each exception to the general rule requires special handling. On some machines, for example, floating-point operations are done using separate registers.

Instruction speeds and machine idioms are other important factors. If we do not care about the efficiency of the target program, instruction selection is straightforward. For each type of three-address statement, we can design a code skeleton that defines the target code to be generated for that construct. For example, every three-address statement of the form x = y + z, where x, y, and z are statically allocated, can be translated into the code sequence.

The quality of the generated code is usually determined by its speed and size. On most machines, a given IR program can be implemented by many different code sequences, with significant cost differences between the different implementations. A naive translation of the intermediate code may therefore lead to correct but unacceptably inefficient target code.

We need to know instruction costs in order to design good code sequences but, unfortunately, accurate cost information is often difficult to obtain. Deciding which machine-code sequence is best for a given three-address construct may also require knowledge about the context in which that construct appears.

Register Allocation

A key problem in code generation is deciding what values to hold in what registers. Registers are the fastest computational unit on the target machine. Values not held in registers need to reside in memory. Instructions involving register operands are invariably shorter and faster than those involving operands in memory, so efficient utilisation of registers is particularly important. The use of registers is often subdivided into two sub-problems:

1. Register allocation, during which we select the set of variables that will reside in registers at each point in the program.
2. Register assignment, during which we pick the specific register that a variable will reside in.

Finding an optimal assignment of registers to variables is difficult, even with single-register machines. Mathematically, the problem is NP-complete.

Evaluation Order

The order in which computations are performed can affect the efficiency of the target code. As we shall see, some computation orders require fewer registers to hold intermediate results than others. However, picking a best order in the general case is a difficult NP-complete problem. Initially, we shall avoid the problem by generating code for the three-address statements in the order in which they have been produced by the intermediate code generator.

Code Generation from DAG

A useful data structure for automatically analysing basic blocks is a *Directed Acyclic Graph (DAG)*. Constructing a DAG from 3-address statements is a good way of determining common sub-expressions within a block.

The advantage of generating code for a basic block from its DAG representation is that, we can easily rearrange the order of the final computation sequence of quadruples central to our discussion is the case where the DAG is a tree. For this case we can generate code that we can prove is optimal under such criteria as program length or the fewest number of temporaries used. This algorithm for optimal code generation from a tree is also useful when the intermediate code is a parse tree.

Rearranging the Order

Consider the following quadruples sequence whose DAG representation is shown below:

$T_1 := A + B$

$T_2 := C + D$

$T_3 := E - T_2$

$T_4 := T_1 - T_3$. –> $T_4 := (A+B) - (E-(C+D))$

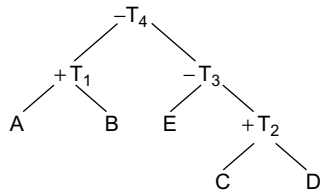


Figure 3.7 DAG for Quadruple Sequence

If we generate code for the quadruples by considering assuming two registers R_0 and R_1 are available, and only T_4 is live on exit, the code sequence is as follows:

```
MOV   A, R0
ADD   B, R0
MOV   C, R1
ADD   D, R1
MOV   R0, T1
MOV   E, R0
SUB   R1, R0
MOV   T1, R1
SUB   R0, R1
MOV   R1, T4
```

Figure 3.8 Code Sequence

On the other hand suppose we rearranged the order of the quadruples so that computation of T_1 occurs immediately before that of T_4 as:

$T_2 := C + D$

$T_3 := E - T_2$

$T_1 := A + B$

$T_4 := T_1 - T_3$, Now we have the code sequence as:

```
MOV   C, R0
ADD   D, R0
MOV   E, R1
SUB   R0, R1
MOV   A, R0
ADD   B, R0
SUB   R1, R0
MOV   R0, T4
```

Figure 3.9 Revised Code Sequence

Note: *By performing the computation according to second order, we have been able to save two instructions.*

3.2.5 Code Optimisation

- Source Level
- Assembly Level

There is no guarantee that always a program will be optimised.

Optimisation Criteria

In general, there are two fundamental criteria that decide which optimisations should be applied to a procedure – speed and space (data space and code space).

Various Optimisations

Figure 3.9 illustrates optimisations that can be carried out various levels. The order is important because in general, transformations are order-dependent. However, there is no "optimal order" for optimisations, and other alternatives may also be acceptable. Some transformations need to be repeated after other optimisations are applied.

The optimisations in box **A** are best performed on a high-level intermediate language (such as HIR). The optimisations in box **B** are best performed on a high or medium-level intermediate language (such as HIR or MIR) and early in the optimisation process. The optimisations done in **C1...C4** are best done on a medium-level or low-level intermediate language (MIR/LIR). **C2** and **C3** are two alternatives for redundancy elimination. The optimisations in box **D** are best done late in the optimisation process and on a low-level intermediate code (e.g. LIR) or on assembly or machine language. The optimisations in box E are done on the re-locatable load before it is loaded but after linking its components.

Constant Folding

Constant folding refers to evaluation at compile time of expressions whose operands are known to be constant. In its simplest form, constant folding involves determining that all the operands in an expression are constant-valued, performing the evaluation of the expression at compile time, and replacing the expression by its value.

Scalar Replacement of Aggregates

Scalar replacement of aggregates involves replacement of aggregates such as C structures and Pascal records with simple scalars of appropriate type.

Algebraic Simplification and Re-associations

Algebraic simplifications use algebraic properties of operators or particular operator-operand combinations to simplify expressions. *Re-assocation* refers to using specific algebraic properties—associativity, commutativity and distributivity, to divide expression to a constant or loop-invariant part and a variable part. Like constant folding, algebraic simplification and re-associations are best implemented as a subroutine, accessible to all phases of the optimiser.

Simplifications for integers:

$i + 0 = 0 + i = i - 0 = i$

$0 - i = -i$

$i * 1 = 1 * i = i / 1 = i$

$i * 0 = 0 * i = 0$

$-(-i) = i$

$i + (-j) = i - j$

Simplifications for shifts:

f shl 0 = f shr 0 = f shra 0 = f

f shl w = f shr w = f shra w = 0

Where w is the word length of the machine.

Some simplifications can be viewed as strength reductions, i.e., replacing an operator by one that is faster to compute, such as

$i \mu 2 = i * i$

$2 * i = i + i$

(where i is an integer value while μ as power function.).

Multiplications by small constants can frequently be done faster by sequence of shifts and adds. For example, i*7 can be computed by:

t ρ i shl 3

t ρ t – i

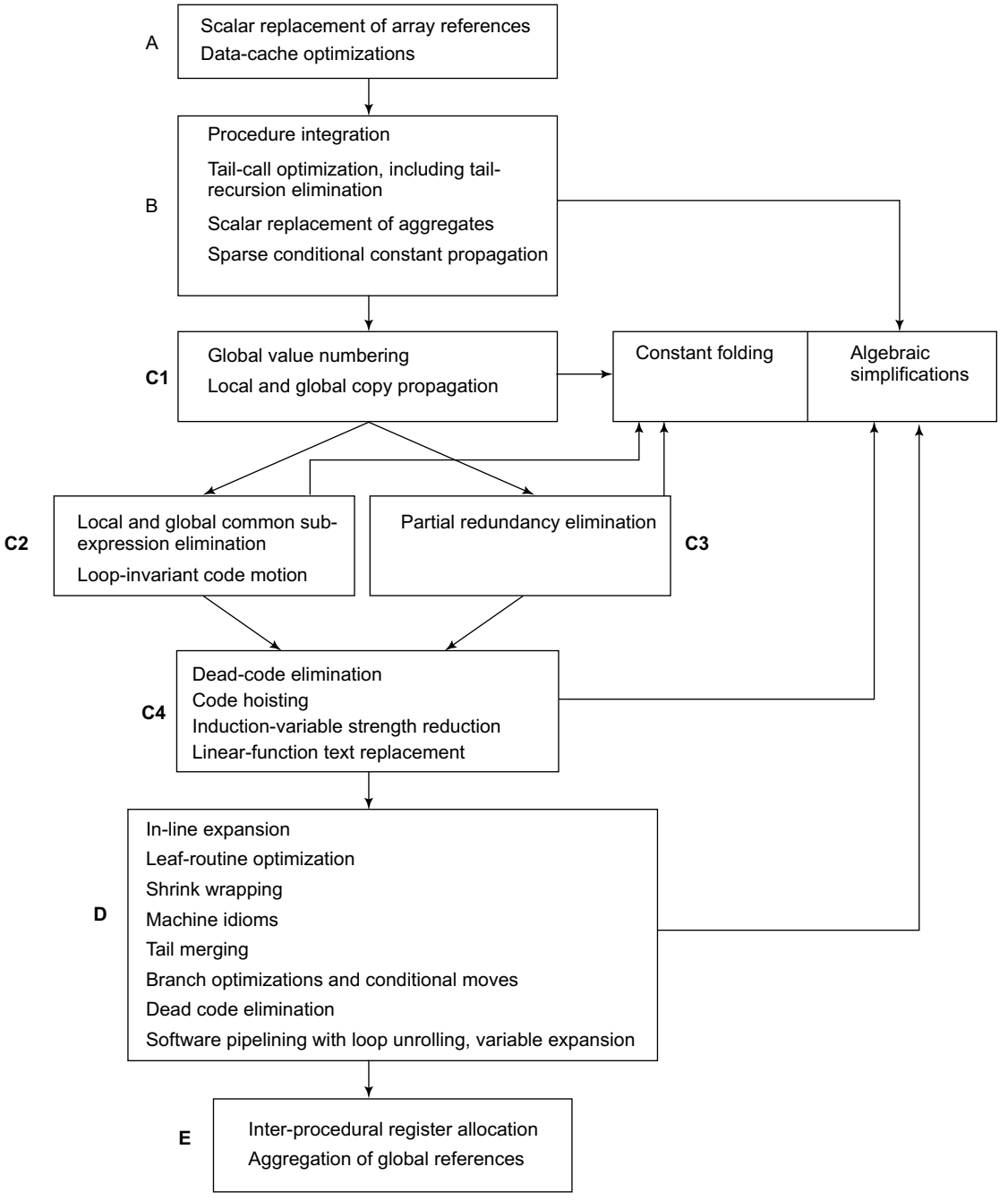


Figure 3.10

Here, we assume i value is shifted left by 3 bits and result is stored in t(ρ is assignment operator). We know that if a number is left shifted by n bits its value will be multiplied by 2^n. As we want i to be multiplied by 7, we shift the number 3 bits left and then, subtract i from t.

Another class of simplifications involves the use of commutativity and associativity. For example:

$(i - j) + (i - j) + (i - j) + (i - j) = 4 * i - 4 * j$

The problem here is that the simplified form (on the right) may cause an overflow, which would not occur for the original form (on the left).

Algebraic Simplifications of Addressing Expressions

Since overflow never makes a difference in addressing arithmetic, it is always safe to perform algebraic simplifications on addressing expressions. However, for addressing expressions the most useful transformation is reassociation. We use *canonicalisation,* namely transforming the expression to sum of products, and then use commutativity to collect the constant-valued and loop-invariant parts together. For example, consider the following Pascal Fragment:

```
var a: array[lo1..hi1,lo2..hi2] of eltype;
 i, j : integer;
. . .
for j:=lo2 to hi2 do begin
 a[i,j] := b + a[i,j]
end;
```

The address of a[i,j] is:

base_a + ((i – lo1) * (hi2 – lo2 + 1) + j – lo2) * w

Note

Do refer row major and column major order storage from data structures unit.

Where base_a is the address of the base of the array and w is the size in bytes of objects of type eltype. This complicated computation should be performed each time the loop is performed!.

After simplification and reassociation we get:

–(lo1*(hi2 – lo2 + 1) – lo2) * w + base_a + (hi2 – lo2+1)*i*w + j*w

Where -(lo1*(hi2 – lo2 + 1) – lo2) * w is a constant, that can be calculated in compile time, base_a + (hi2 – lo2+1)*i*w is loop-invariant, and so can be computed once before entering the loop, leaving only j*w to be computed and added in each iteration. This multiplication can also be strength reduced into addition. Please do refer row and column major order storage discussion in Unit 2 for clarifications about the above expressions. This simplification can be done by representing the expression in a tree, and applying a series of transformations recursively, until a canonical form is reached.

Local Optimisations

Most of the simple local optimisations performed in so-called basic block.

A basic block is a maximal sequence of instructions with:

- no labels (except the first instruction)
- no jumps(except the last instruction)
- each instruction is executed after all the preceding instructions.

■ **Example 1.** The following code will produce 2 blocks: (1–2) and (3–9).

```
1.  prod : = 1
2.  i: = 1
3.  t1 = 4*i
4.  t2: = a[t1]
5.  t3: = t2 *t1
6.  t4: = prod + t32
7.  t5 = i + 1
8.  i: = t5
9.  if i < = 20 goto 3
```

```
1.  prod : = 1
2.  i: = 1
```

```
3.  t1 = 4*i
4.  t2: = a[t1]
5.  t3: = t2 *t1
6.  t4: = prod + t3
7.  t5 = i + 1
8.  i: = t5
9.  if i < = 20 goto 3
```

Simple local optimisations should be performed in each block separately.

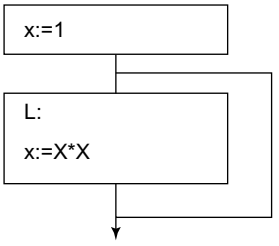
Control Flow Graph(CFG)

A control-flow graph is a directed graph with

- basic blocks as nodes

- an edge from block A to B if the execution can flow from the last instruction of A to the first instruction of B (the last instruction of A is jump Lb)

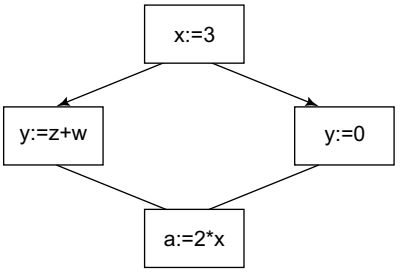
■ **Example 2.**



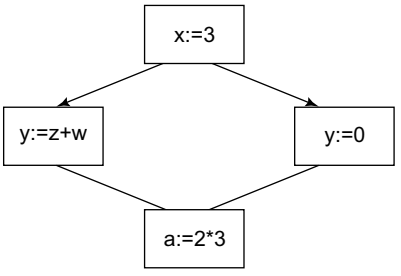
Global optimisation

Global optimisation is more complicated to perform. The entire program should be analysed and series of simple optimisations performed. However keeping in mind the control flow of the program an output of the simple optimisation can be different.

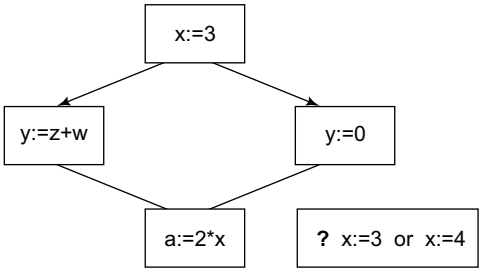
Let us analyse the following CFG



If constant propagation is performed then the result is:



However if the original program was:



Then constant propagation can not be used.
To use constant propagation (replace x with some value k) we must know that *on every path to the use of x the last assignment to x was x:=k.*

Checking all the paths can be difficult (more information about global optimisation can be found in the course book).

Peephole optimisation

The machine language is assembly that is later translated by assembler to the machine code.

Peephole optimisation is an effective technique for improving the assembly code.

(Peephole optimisation may be applied to either intermediate form code or to assembly code).

The main idea in this technique is that the compiler searches for a comparatively short machine instruction sequence that meeting particular context conditions and replaces the sequence with a more efficient one.

- The "peephole" is a short sequence (of usually contiguous) instructions.
- The optimiser replaces the sequence with another equivalent one (but faster).

■ **Example 3.**

1. mov r0,a
2. mov a,r0

instruction 2 can be deleted because whenever 2 is executed 1 will ensure that the value of a already in memory register r0.

Or

Jump to the next instruction

```
jmp     L1
L1:     mov     buffer, R0
```

Simply replaced with

```
L1:
```

Or

Duplicate transfer instruction

```
mov     a, r0
mov     a, r0
```

Simply replaced with

```
mov     a, r0
```

Or

Jump to another jump instruction

```
jz      L2
…
L2:     jmp     .L3
```

Simply replaced with

```
jz L3
```

Word-wide optimisation

If we have short 16-bit data, access two values at once and then work with 32-bit data. Also, called packed data processing.

1. Use casts and intrinsics

```
int dot_product(short *x, short *y, short z) //x,y point to short variables
{
int *w_x = (int *)x;  //cast to point to a word
int *w_y = (int *)y;
int sum1 = 0, sum2 = 0, i;
for (i = 0; i < z/2; i++){
sum1 += _mpy(w_x[i], w_y[i]);    // mpy 16LSB
sum2 += _mpyh(w_x[i], w_y[i]);   // mpy 16MSB
}
return (sum1 + sum2);
}
```

Equivalent to summing the even indexed numbers in the array (i=0,2,4) and summing the odd indexed numbers in the array and then summing the sums. Do remember that mpy is an assembly instruction in some processors which carries 16 bit multiplication.

2. Use _nassert intrinsic to specify that 16-bit short arrays are aligned on a 32-bit (word) boundary.

```
int dot_product(short *x, short *y, short z){
int sum = 0, i;
_nassert (((int)(x) & 0x3) == 0);
_nassert (((int)(y) & 0x3) == 0);
#pragma MUST_ITERATE(20, , 4);
//must be even number of times in order to split in half for optimization
for (i = 0; i < z; i++) sum += x[i] * y[i];
return sum;
}
```

_nassert tells the computer that the pointer x and y are on a word boundary (2 LSBs are 0). The compiler then knows that it can optimise the following loop with mpy and mpyh. _nassert is useful in that you do not need to rewrite code – simply add line.

Loop unrolling

- Small loops can be unrolled for higher performance at the expense of increased code size.
- When a loop is unrolled, the loop counter needs to be updated less often and fewer branches are executed.
- If the iteration number is small a loop can be fully unrolled and the loop overhead completely disappears.
- Loop unrolling is not supported by the compiler, should be done manually.

```
int countbit1 (uint n){
int bits = 0;
while (n !=0){
if (n & 1) bits++;
n >>=1;
}
return bits;
}
```

In ARM7 processor, checking a single bit takes 6 cycles, and the code size is only 9 instructions.

```
---------------------------------
int countbit1 (uint n){
int bits = 0;
while (n !=0) {
if (n & 1) bits++;
if (n & 2) bits++;
if (n &4) bits++;
if (n & 8) bits++;
n >>=4;        /* shift right by 4*/
}
return bits;
}
```

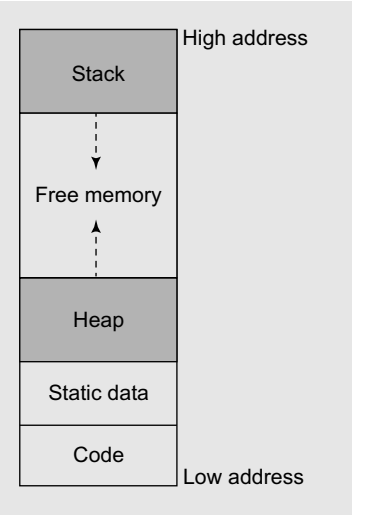
The above code checks 4 bits at a time, taking on average 3 cycles per bit. However, the code size is 15 instructions.

3.2.6 Run-Time Storage Organisation

The runtime environment is the structure of the target computers registers and memory that serves to manage memory and maintain information needed to guide a programs execution[1] process. In fact, almost every programming language uses one of the following three kinds of runtime environments.

[1] In most compiled languages, it is not possible to make changes to the code area during execution

- **Fully static Environment (Ex. In Fortran 77)**
- **Fully dynamic Environment (Ex in C, C++, Pascal, Ada)**
- **Stack Based Environment. (Ex in Lisp, other functional languages)**



An executable program generated by a compiler will have the following organisation in memory on a typical architecture.

This is the layout in memory of an executable program. Note that in a virtual memory architecture (which is the case for any modern operating system), some parts of the memory layout may in fact be located on disk blocks and they are retrieved in memory by demand (lazily).

The code area contains code of the program. It contains the entry points for the procedures are functions used in the program. All code addresses are computable at compile time; at least relative to some base.

There is one class of data that can be fixed in memory prior to execution and that comprises global and/or static data of the a program (Note in Fortran 77 all data items are global in nature). In C, external and static variables also comes under this category in addition to global variables. The declarations such **const of** C, Pascal and string literals, integer/long constants (such as 12121) etc., also occupies this data area[2].

The dynamically allocated data (i.e., the data created using malloc in C) as well as the static data without a fixed size (such as arrays of variable size) are created and kept in the heap. The heap grows from low to high addresses. When you call malloc in C to create a dynamically allocated structure, the program tries to find an empty place in the heap[3] with sufficient space to insert the new data; if it can not do that, it puts the data at the end of the heap and increases the heap size.

The focus of this section is the stack in the memory layout. It is called the *run-time stack*. The stack, in contrast to the heap, grows in the opposite direction (upside-down): from high to low addresses, which is a bit counterintuitive. The stack is not only used to push the return address when a function is called, but it is also used for allocating some of the local variables of a function during the function call, as well as for some bookkeeping.

Fully Static Runtime Environment

- Fully static runtime environment may be useful for the languages in which pointers or dynamic allocation is not possible in addition to no support for recursive function calls.
- Every procedure will have only one activation record which is allocated before execution.
- Variables are accessed directly via fixed addresses.
- Little book keeping overhead; i.e., at most return address may have to be stored in activation record.
- The calling sequence involves calculation of each argument address and storing into its appropriate parameter location and saving the return address and then a jump is made.

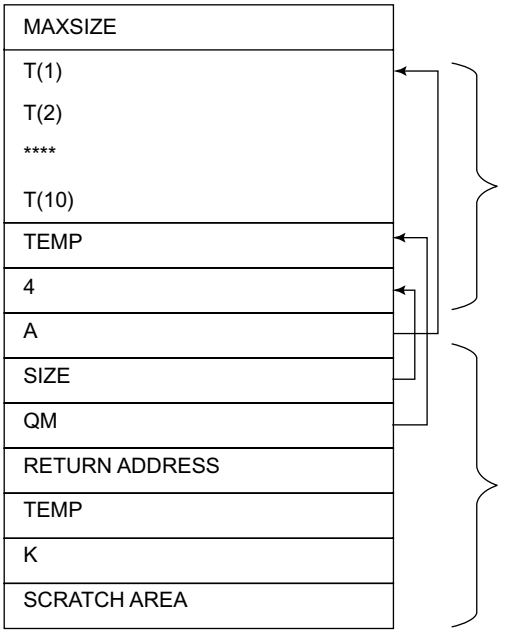
A Fortran Example

```
PROGRAM TEST
COMMON MAXSIZE
INTEGER MAXSIZE
REAL T(10), TEMP
MAXSIZE=10
READ *, T(1),T(2),T(3),T(4)
CALL SUM(T,4,TEMP)
PRINT *, TEMP
END
```

[2] Small compile time constants such 0 or 1 etc are directly included in the code.

[3] Heap is a linear memory area. It is unrelated to heap data strcuture of data structure.

```
      SUBROUTINE SUM(A,SIZE,QM)
      COMMON MAXSIZE
      INTEGER MAXSIZE, SIZE
      REAL A(SIZE), QM, TEMP
      INTEGER K
      TEMP=0.0
      DO 1 K=1,SIZE
      TEMP = TEMP + A(K)
    1 CONTINUE
      QM = TEMP
      RETURN
      END
```



ACTIVATION RECORD STRUCTURE FOR MAIN & SUBROUTINE

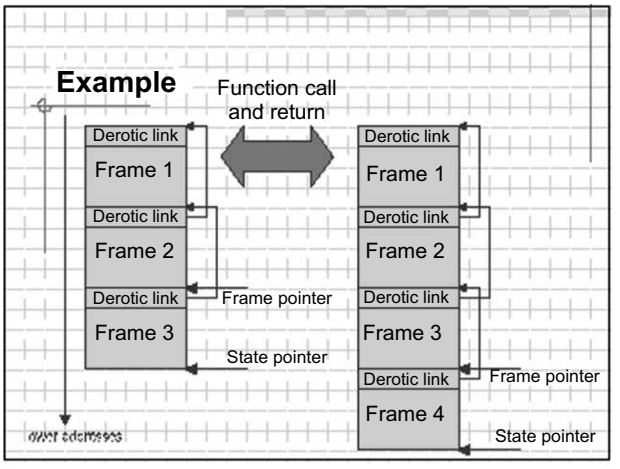
Stack Based Runtime Environments

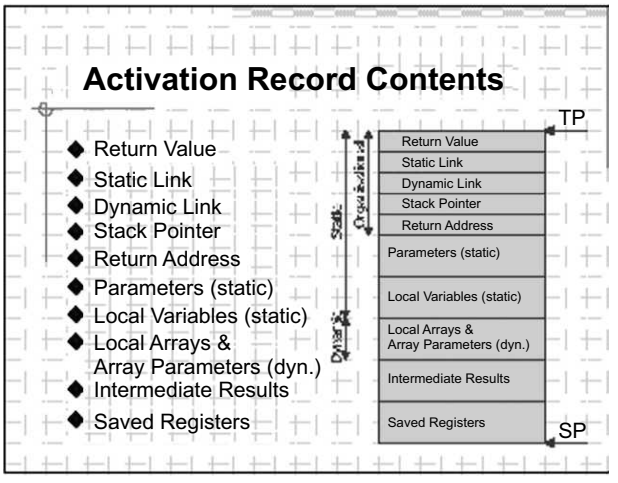
Here, activation records are allocated (push of the activation records) whenever a function call is made. The necessary memory is taken from stack portion of the program. When program execution returns from function the memory used by the activation record is deallocated (pop of the activation record). Thus, stack grows and shrinks with the chain of function calls (Each function may have several different activation records also).

Let us consider the lifetime of a function call. When you call a function you not only want to access its parameters, but you may also want to access the variables local to the function. Worse, in a nested scoped system where nested function definitions are allowed, you may want to access the local variables of an enclosing function. In addition, when a function calls another function, we must forget about the variables of the caller function and work with the variables of the callee function and when we return from the callee, we want to switch back to the caller variables. That is, function calls behave in a stack-like manner. Consider for example the following program:

```
procedure P ( c: integer )
 x: integer;
 procedure Q ( a, b: integer )
 i, j: integer;
 begin
 x := x+a+j;
 end;
begin
 Q(x,c);
end;
```

At run-time, P will execute the statement x := x+a+j in Q. The variable a in this statement comes as a parameter to Q,





while j is a local variable in Q and x is a local variable to P. How do we organise the runtime layout so that we will be able to access all these variables at run-time? The answer is to use the run-time stack.

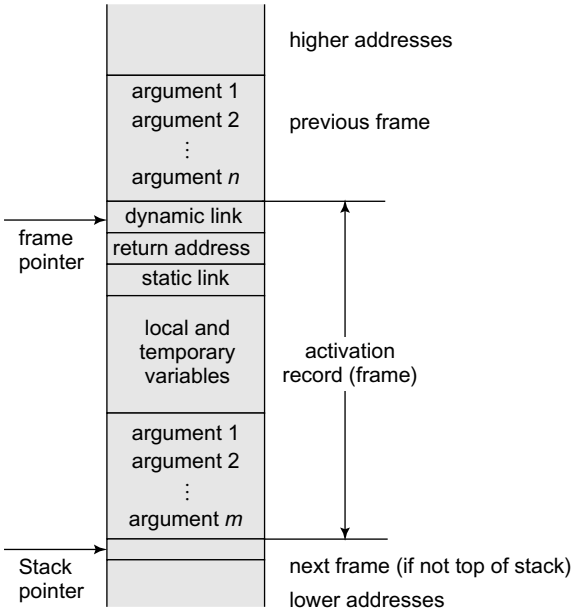
When we call a function f, we push a new *activation record*[4] (also called as a activation *frame*) on the run-time stack, which is particular to the function f. Each frame can occupy many **consecutive bytes in the stack and may not be of a fixed size**. When the callee function returns to the caller, the activation record of the callee is popped out.

For example, if the main program calls function P, P calls E, and E calls P, then at the time of the second call to P, there will be 4 frames in the stack in the following order: main, P, E, P

Note that a callee should not make any assumptions about who is the caller because there may be many different functions that call the same function. The frame layout in the stack should reflect this. When we execute a function, its frame is located on top of the stack. The function does not have any knowledge about what the previous frames contain. Two things one has to do when executing a function: the first is that we should be able to pop-out the current frame of the callee and restore the caller frame. This can be done with the help of a pointer in the current frame, called the *dynamic link*[5], that points to the previous frame (the caller's frame). Thus all frames are linked together in the stack using dynamic links. This is called the *dynamic chain*. When we pop the callee from the stack, we simply set the stack pointer to be the value of the dynamic link of the current frame. The second thing that we need to do, is if we allow nested functions, we need to be able to access the variables stored in previous activation records in the stack. This is done with the help of the *static link*. Frames linked together with static links form a *static chain*. The static link of a function f points to the latest frame in the stack of the function that lexically (statically) contains f. If f is not lexically contained in any other function, its static link is null. For example, in the previous program, if P called Q then the static link of Q will point to the latest frame of P in the stack (which happened to be the previous frame in our example). Note that we may have multiple frames of P in the stack; Q will point to the latest. Also notice that there is no way to call Q if there is no P frame in the stack, since Q is hidden outside P in the program.

- Program Counter (PC) whose value is the address of the next instruction to be executed.
- Stack Pointer (SP) whose value is top of the stack (top of stack, tos).
- Frame Pointer (FP) which points to the current activation record.
- Argument Pointer (AP) which points to the area of the activation record reserved for arguments.
- Instruction pointer (IP) or code pointer.
- Closure is combination of IP and access link (Environment Pointer).

A typical organisation of a frame is the following:



Before we create the frame for the callee, we need to allocate space for the callee's arguments. These arguments belong to the caller's frame, not to the callee's frame. There is a frame pointer (called FP) that points to the beginning of the frame.

[4] Also known as stack frame when it is in stack

[5] control link

The stack pointer points to the first available byte in the stack immediately after the end of the current frame (the most recent frame). There are many variations to this theme. Some compilers use *displays* to store the static chain instead of using static links in the stack. A display is a register block where each pointer points to a consecutive frame in the static chain of the current frame. This allows a very fast access to the variables of deeply nested functions. Another important variation is to pass arguments to functions using registers.

When a function (the caller) calls another function (the callee), it executes the following code before (the pre-call) and after (the post-call) the function call:

- *pre-call*[6]*:* allocate the callee frame on top of the stack; evaluate and store function parameters in registers or in the stack; store the return address to the caller in a register or in the stack.
- *post-call*[7]*:* copy the return value; deallocate (pop-out) the callee frame; restore parameters if they passed by reference.

In addition, each function has the following code in the beginning (prologue) and at the end (epilogue) of its execution code:

- *prologue:* store frame pointer in the stack or in a display; set the frame pointer to be the top of the stack; store static link in the stack or in the display; initialise local variables.
- *epilogue:* store the return value in the stack; restore frame pointer; return to the caller.

Depending on the language, activation records are allocated from either static area (like Fortran 77), or from stack area (like C or Pascal) or from heap area (like Lisp).

We can classify the variables in a program into four categories:

- statically allocated data that reside in the static data part of the program; these are the global variables.
- dynamically allocated data that reside in the heap; these are the data created by malloc in C.
- register allocated variables that reside in the CPU registers; these can be function arguments, function return values, or local variables.
- frame-resident variables that reside in the run-time stack; these can be function arguments, function return values, or local variables.

Every frame-resident variable (ie. a local variable) can be viewed as a pair of (level, offset). The variable level indicates the lexical level in which this variable is defined. For example, the variables inside the top-level functions (which is the case for all functions in C) have level=1. If a function is nested inside a top-level function, then its variables have level=2, etc. The offset indicates the relative distance from the beginning of the frame that we can find the value of a variable local to this frame. For example, to access the *n*th argument of the frame, we retrieve the stack value at the address FP+1, and to access the first argument, we retrieve the stack value at the address FP+n (assuming that each arguments occupies one word). When we want to access a variable whose level is different from the level of the current frame (ie. a non-local variable), we subtract the level of the variable from the current level to find out how many times we need to follow the static link (ie. how deep we need to go in the static chain) to find the frame for this variable. Then, after we locate the frame of this variable, we use its offset to retrieve its value from the stack. For example, to retrieve to value of x in the Q frame, we follow the static link once (since the level of x is 1 and the level of Q is 2) and we retrieve x from the frame of P.

Another thing to consider is what exactly do we pass as arguments to the callee. There are two common ways of passing arguments: by value and by reference. When we pass an argument by reference, we actually pass the address of this argument. This address may be an address in the stack, if the argument is a frame-resident variable. A third type of passing arguments, which is not used anymore but it was used in Algol, is call by name. In that case we create a *thunk* inside the caller's code that behaves like a function and contains the code that evaluates the argument passed by name. Every time the callee wants to access the call-by-name argument, it calls the thunk to compute the value for the argument.

Case Study: The MIPS Architecture

The following explanation describes a function call abstraction for the MIPS architecture. This may be slightly different from the one you will use for the project.

MIPS uses the register $sp as the stack pointer and the register $fp as the frame pointer. In the following MIPS code we use both a dynamic link and a static link embedded in the activation records.

[6] call sequence
[7] return sequence

Consider the previous program:

```
procedure P ( c: integer )
 x: integer;
 procedure Q ( a, b: integer )
 i, j: integer;
 begin
 x := x+a+j;
 end;
begin
 Q(x,c);
end;
```

The activation record for P (as P sees it) is shown in the first figure of 3.11.

The activation record for Q (as Q sees it) is shown in the second figure of 3.11. The third figure of 3.11 shows the structure of the run-time stack at the point where x := x+a+j is executed. This statement uses x, which is defined in P. We can not assume that Q called P, so we should not use the dynamic link to retrieve x; instead, we need to use the static link, which points to the most recent activation record of P. Thus, the value of variable x is computed by:

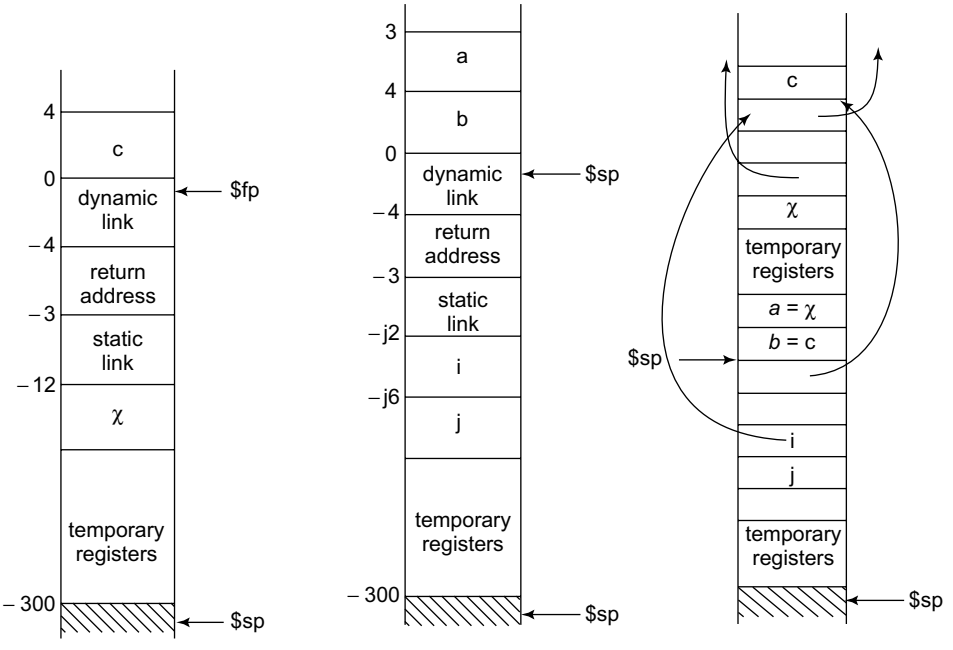


Figure 3.11

```
lw $t0, -8($fp)          # follow the static link of Q
lw $t1, -12($t0)         # x has offset=-12 inside P
```

Function/procedure arguments are pushed in the stack before the function call. If this is a function, then an empty placeholder (4 bytes) should be pushed in the stack before the function call; this will hold the result of the function.

Each procedure/function should begin with the following code (prologue):

```
sw $fp, ($sp) # push old frame pointer (dynamic link)
move $fp, $sp # frame pointer now points to the top of stack
subu $sp, $sp, 500 # allocate say 500 bytes in the stack
# (for frame size = 500)
sw $ra, -4($fp) # save return address in frame
sw $v0, -8($fp) # save static link in frame
```

(where $v0 is set by the caller - see below) and should end with the following code (epilogue):

```
lw $ra, -4($fp) # restore return address
move $sp, $fp # pop frame
lw $fp, ($fp) # restore old frame pointer (follow dynamic link)
jr $ra # return
```

For each procedure call, you need to push the arguments into the stack and set $v0 to be the right static link (very often it is equal to the static link of the current procedure; otherwise, you need to follow the static link a number of times). For example, the call Q(x,c) in P is translated into:

```
lw $t0, -12($fp)
sw $t0, ($sp) # push x
subu $sp, $sp, 4
lw $t0, 4($fp)
sw $t0, ($sp) # push c
subu $sp, $sp, 4
move $v0, $fp # load static link in $v0
jal Q # call procedure Q
addu $sp, $sp, 8 # pop stack
```

Note that there are two different cases for setting the static link before a procedure call. Lets say that caller_level and callee_level are the nesting levels of the caller and the callee procedures (recall that the nesting level of a top-level procedure is 0, while the nesting level of a nested procedure embedded inside another procedure with nesting level l, is $l + 1$). When the callee is lexically inside the caller's body, that is, when callee_level=caller_level+1, we have:

```
move $v0, $fp
```

The call Q(x, c) in P is such a case because the nesting levels of P and Q are 0 and 1, respectively. Otherwise, we follow the static link of the caller $d + 1$ times, where d=caller_level-callee_level (the difference between the nesting level of the caller from that of the callee). For d=0, that is, when both caller and callee are at the same level, we have

```
lw $v0, -8($fp)
```

For d=2 we have

```
lw $t1, -8($fp)
lw $t1, -8($t1)
lw $v0, -8($t1)
```

These cases are shown in Figure 3.12.

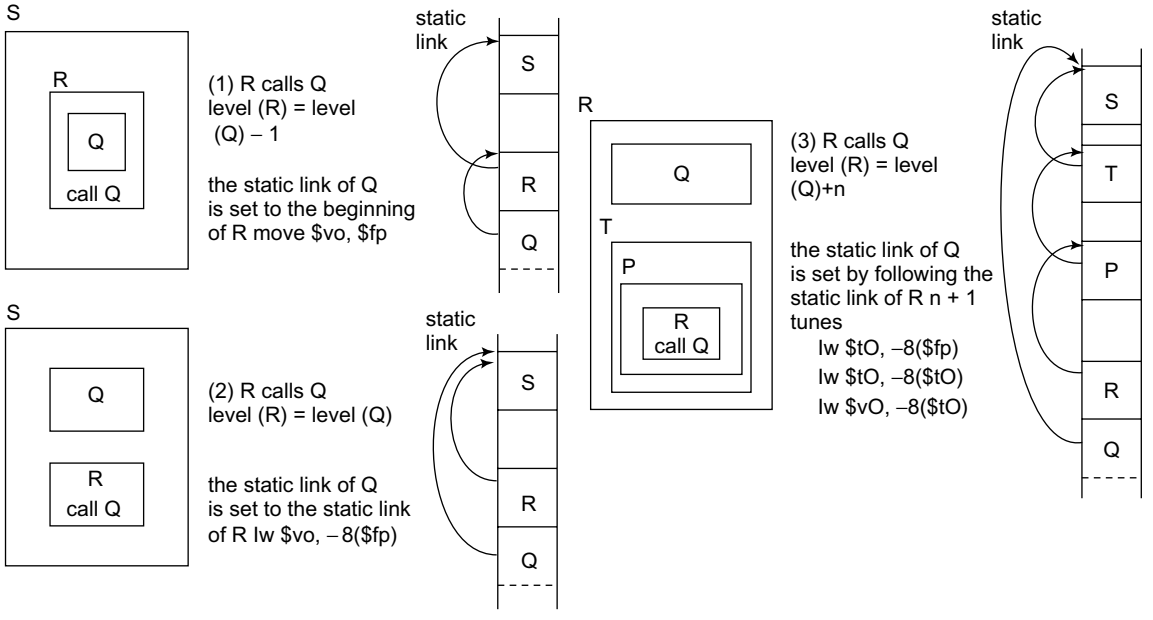


Figure 3.12

Note also that, if you have a call to a function, you need to allocate 4 more bytes in the stack to hold the result.

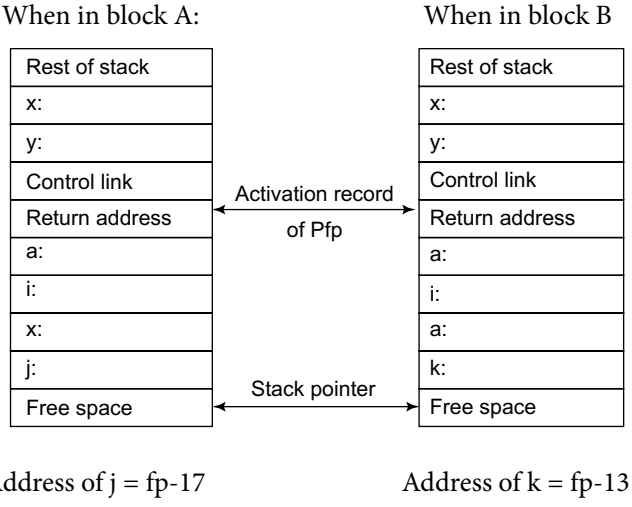
■ **Example**

Consider the following C code fragment.

```
Void p(int x, double y)
{
int I;
…..
A:
{
double x;
int j;
…..
}
…..
B:
{
char *a;
int k;
…
}
…
}
```

Draw run time stack when we enter into block A. Calculate address of j with respect to its fp. Similarly find out the address of k with respect to fp. Assume the data sizes as: integer = 2 bytes, char = 1 byte, double = 8 bytes, address =4 bytes.

■ **Answer:**

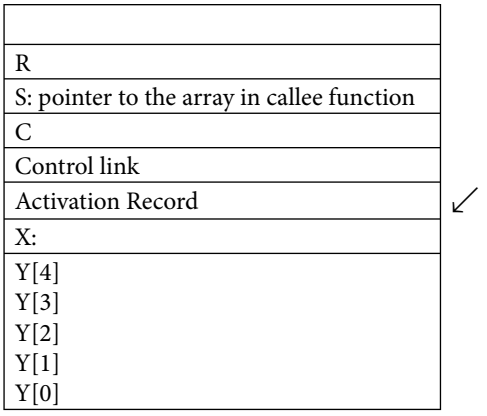


■ **Example 1.** Consider the following procedure in C syntac:

```
void f(char c, char s[10], double r){
int *x;
int y[5];
}
```

Using the standard C parameter passing conventions, and assuming the data sizes as: integer = 2 bytes, char = 1 byte, double = 8 bytes, address =4 bytes, determine the offsets from the fp of the following, using activation record structure which does not have access link.

Assuming all parameters are passed in C standard convention. That is, the array (address) is sent to the function. Others are sent in passing by value style.



Addresses:

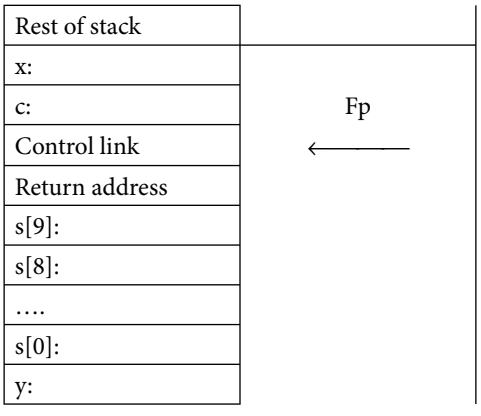
Variable	address	
C	fp+4	
S	fp+5	
S [I]	@5(fp) + 2*I	here @ means indirection
R	fp+9	
X	fp -4	
Y	fp-18	

■ **Example 2.**

Consider the following C program segment.

```
void f(int x, char c){
 int s[10];
double y;
}
```

Using the standard C parameter passing conventions, and assuming the data sizes as: integer = 2 bytes, char = 1 byte, double = 8 bytes, address =4 bytes, determine the offsets from the fp of the following, using activation record structure which does not have access link.



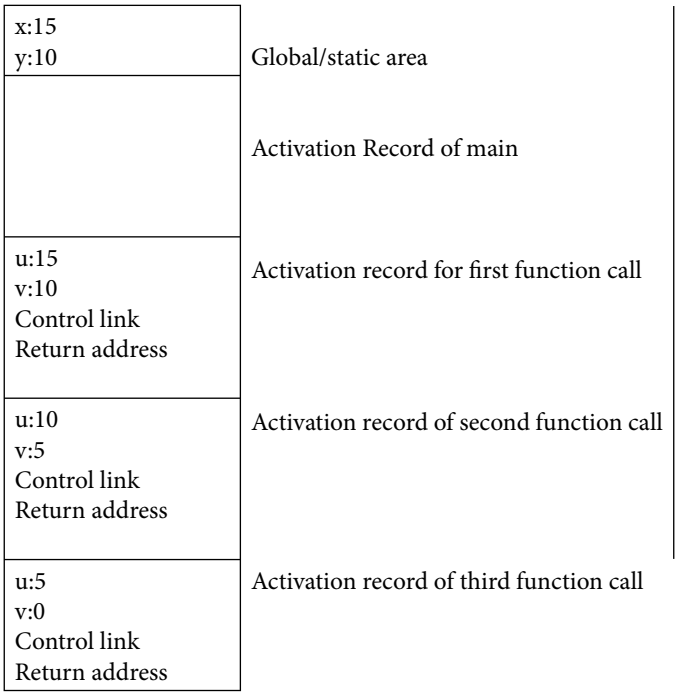
Name	Offset from fp
x	+5
c	+4
s	–24
y	–32

i'th element of s can be found as: –24+i*2+fp

■ **Example 3.**

Draw stack frame structure for the following recursive function.

```
int x,y;
int gcd(int u, int v) {
if(v==0) return u;
else
return(gcd(v,u%v));
}
main(){
 x=15; y=10;
printf("%d\n", gcd(x,y) );
}
```



The calling function may create stack frame and push the arguments. Where as the other part is created by called function.

A language where a reference to a local variable if returned from a procedure in stack based environment it leads to dangling reference as by that time activation record is de-allocated from the stack.

Because of dangling references, in C local functions are illegal.

Pass-by-value-Result requires several modifications to the basic structure of the runtime stack and the calling sequence. That is, first activation record can not be freed till local values are copied back. Also, it should have a mechanism to re-compute their addresses after returning from a function call.

If a static variable is declared in a function then the necessary memory is not allocated in that functions activation record. Rather, it is allocated in the global/static area of the program such that it is available across different calls of f. If we have another global/static variable with the same name also there will not be any confusion as symbol table maintains the scope of the static variables.

Activation tree is the one which represents function calls hierarchy.

If frame size is constant then we can even use SP as reference for locating all variables in stack frame.

Parameters passed in registers will also have an address in activation record.

Outgoing arguments of frame n and return value of frame n+1 accessed within frame n with positive offset relative to fp. But incoming arguments of frame n+1 and return value accessed within frame n+1 with negative offset relative to fp.

Parameter passing on to stack increases needless memory traffic. This can be improved by the following means:

- Pass parameters in registers
- Pass first k (usually 4 or 6) arguments in registers and remaining on stack.
- Avoid additional memory traffic for storing current content of regisetrs used for parameter passing.
- If procedures are leaf nodes try to continue the execution even without creating stack frames.
- By using register windows (RISC architectures uses)
- Inter Procedural register allocation
- Use registers used by dead variables.

Global variable are accessed directly (or by offset from base pointer) where as local variables of a function are accessed with respect to fp.

While dealing with variable length data (such as a function call **printf("Hello How are you\n")** such as unconstrained array (also called as open arrays) of Ada. That is array size is unpredictable during compilation time. This can be taken care by an extra level of indirection. That is, pointer is allocated during the compile time and actual memory is allocated from **tos** and its addressed is stored here. Of course, during function return such a memory has to be de-allocated such that there will not be any dangling problems.

Calling a function with variable no. of arguments (such as printf function call which accepts any number of argments) is another important aspect should be given emphasis. This is dealt by C language compilers by pushing arguments in the reverse order on to runtime stack. Then, the first parameter is always located at a fixed offset (+4) from fp. Another option is to use argument pointer (used in VAX architectures).

Access link is an extra piece of book keeping information which is used in languages which supports local procedures which points to the activation record of defining environment rather than calling environment. This is also called as static link though it is not compile time quantity. If a function P embodies two sub functions say Q, R in it then P may not necessarily have any access link.

Access chaining is implemented by repeatedly fetching access links to access a variable. This is an inefficient method. Thus, display data structure is used which maintains access links outside the stack and is indexed by nesting level.

Some programming language supports functions as arguments to functions. Here, a pair of pointers known as closure is used which is combination of instruction pointer and environment pointer.

The code that pushes the arguments is generated by the compiler when it processes the function call, but the offsets are figured when the compiler processes the function declaration. Since the call and declaration can be in different files, there is no way that the compiler can check for consistency unless you use a function prototype in a common header file. Otherwise we get a nasty bug known as **phase error.**

Tail recursion

While writing recursive functions if we employ tail recursion the number of activation records the function call is created will be reduced.

The following examples explains the factorial calculation:

Direct recursion version

```
int fact(int n){
if n(<0) return 0;
else if(n==0) return 1;
else if(n==1) return 1;
else return n*fact(n-1);
}
```

Tail recursion version

```
int fact(int n, int a){ /* a value of 1 is passed*/
if n(<0) return 0;
else if(n==0) return 1;
else if(n==1) return a;
else return fact(n-1, n*a);
}
```

If we call the above function with n value as 1 then in the first version in total 4 activation records may be created. Whereas in the second one with one activation record the recursive call is executed.

Fully Dynamic Runtime Environment

Functional languages such as Lisp, ML, etc uses this style of call stack management. Sailently, here activation records are de-allocated only when all references to them have disappeared, and this requires that activation records to dynamically freed at arbitrary times during execution. Here, excellent memory manager (garbage collector) is needed.

The data structure that handles such a management is heap and thus this is also called as heap management.

3.3 Solved Questions

1. If an NFA M accepts the empty string (i.e., ϵ), does M's start state have to be an accepting state? Why or why not?

Answer: No. It does not have to be an accepting state. Because M is an NFA, we can accept the empty string by having a non-accepting start state which has an ϵ-transition (or a sequence of such ϵ-transitions) to an accepting state.

2. Is every finite language regular? Why or why not?

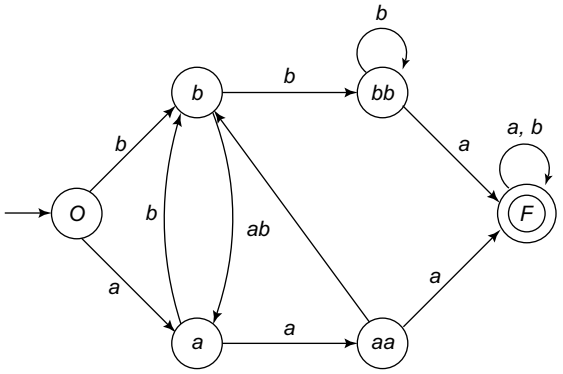
Answer: Yes. Every finite language is regular. You can build an NFA for it by building a linear DFA that recognises each string individually. Then join them all to a common start state using ϵ-transitions.

3. Suppose an NFA M recognises a language L then its complement M' recognises L'. Is this statement valid for all languages?

Answer: No. Valid for DFA only.

4. Construct an DFA over alphabet Σ={a,b} which recognises strings having substrings aaa or bba.

Answer:



5. Let Σ and ⌈ be alphabets. Suppose that h is a function from Σ* to ⌈*. Define what it means for h to be a homomorphism.

Answer: A mapping h is a homomorphism if h(xy) = h(x)h(y) for any strings x and y. Or, equivalently, h is a homomorphism if $h(c_1c_2\ldots c_n) = h(c_1)h(c_2) \ldots h(c_n)$ for any sequence of characters $c_1c_2 \ldots c_n$.

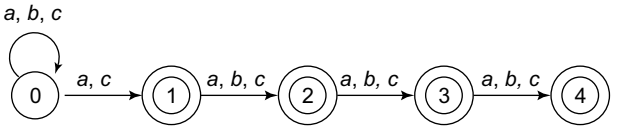
Or, we can say it in other words: the output of h on a string is the concatenation of its outputs on the individual characters making up the string.

6. Give a regular expression for the language L containing all strings in a*b* whose length is a multiple of three. E.g. L contains aaaabb but does not contain ababab or aaabb.

Answer: (aaa)*(bbb)* U(aaa)*aab(bbb)*U (aaa)*abb (bbb)*.

7. Let Σ = {a, b, c}. Give an NFA for the language L containing all strings in Σ* which have an a or a c in the last four positions. E.g. bbabbb and abbbcb are both in L, but acabbbb is not. Notice that strings of length four or less are in L exactly when they contain an a or a c.

Answer:



8. Is the language $\{ww^Rw/w \epsilon \{a; b\}\}$ a context-free language?

Answer: No. A context-free language can only generate matched pairs, not matched triples.

9. If L is a non-regular language over Σ*, and h is a homomorphism, then h(L) must also be non-regular. How valid is this statement?

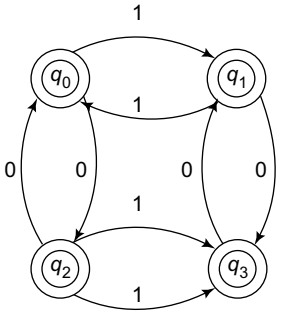
Answer: No. Suppose that h mapped all characters to the empty string. Then h(L) would be regular no matter what L is.

10. Suppose all the words in language L are no more than 1024 characters long. Then L must be regular. Is this statement valid?

Answer: Yes. There is only a finite set of strings with ⇐ 1024 characters. So L is finite and therefore regular.

11. Draw a transition diagram of an FA which accepts all strings of 0's and 1's in which both the number of 0's and 1's are even.

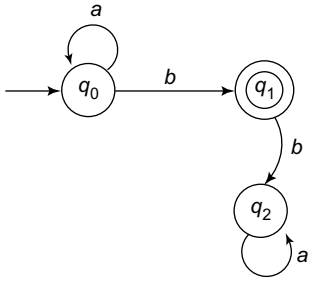
Answer:



12. Construct a DFA that accepts all strings consisting of arbitrary number of a's followed by a single 'b'.

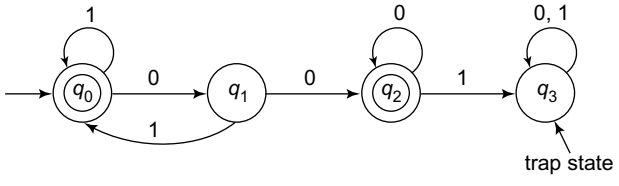
$L=\{a^n b/n>=0\}$

Answer:



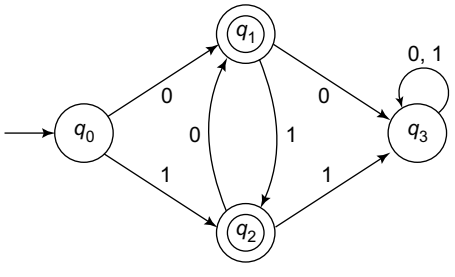
13. Construct a DFA that accepts all strings defined on {0,1}, except those containing substring 001.

Answer:



14. Design an NFA that accepts all strings defined on {0,1}, except those containing two consecutive zero's or one's .

Answer:



15. Explain what is meant by ϵ-closure. Also, find out whether the string "ab" is acceptable according to the given NFA.

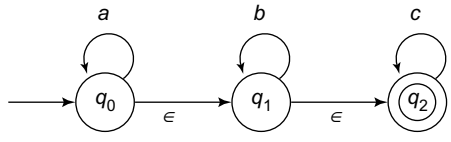
Answer:

∈ - closure of a particular state is a set of all those states of automata which can be reached from that state on a path labeled by '∈'.

Rules:

- q0 is added to ∈ - closure of q0
- If q1 is in ∈ - closure of q0 and there is an edge labeled '∈' from q1 to q2, then q2 is also added to ∈ -closure of q0 if q2 is not already there.
- If 'T' is a set of states then ∈ - closure (T) is the union of ∈ - closure of single states.

For the figure given below, check whether input 'ab' is accepted or not



∈ - closure $(q_0) = \{q_0, q_1, q_2\}$

$\delta^*(q_0, \in) = \in$ - closure $(q_0) = \{q_0, q_1, q_2\}$

$\delta^*(q_0, a) = \in$ - closure $(\delta(\delta^*(q_0, \in)), a)$

$= \in$ - closure $(\delta(\{q_0, q_1, q_2\}), a)$

$= \in$ - closure $(\delta(q_0,a), \delta(q_1,a), \delta(q_2,a))$

$= \in$ - closure $(\{q_0\}, \phi, \phi)$

$= \in$ - closure $(\{q_0\})$

$= \{q_0, q_1, q_2\}$

$\delta^*(q_0, ab) = \in$ - closure $(\delta(\delta^*(q_0, a)), b)$

$= \in$ - closure $(\delta(q_0,b), \delta(q_1,b), \delta(q_2,b))$

$= \in$ - closure $(\phi, \{q_1\}, \phi)$

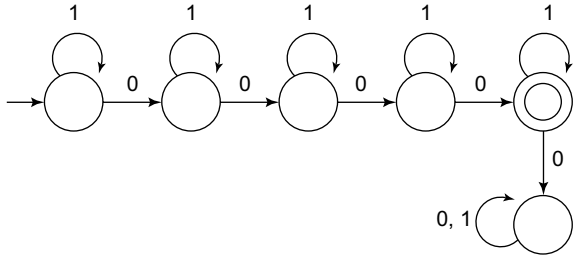
$= \in$ - closure (q_1)

$= \{q_1, q_2\}$

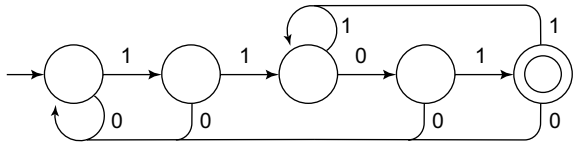
q_2 is the final state , hence 'ab' is accepted by the machine.

16. Some useful DFAs

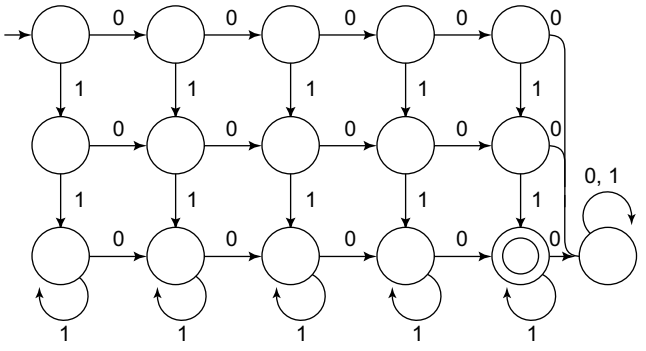
a. All strings that contain exactly 4 0s.



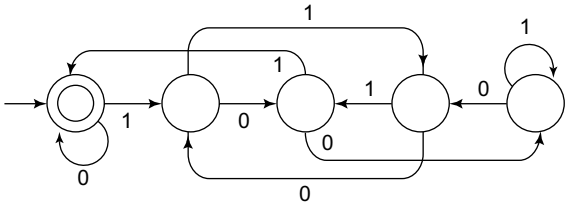
b. All strings ending in 1101.



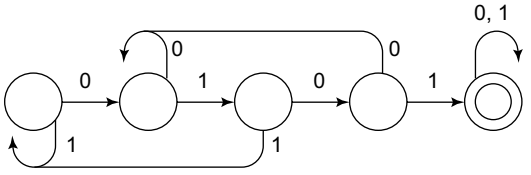
c. All strings containing exactly 4 0s and at least 2 1s.



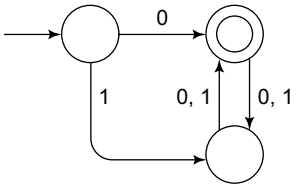
d. All strings whose binary interpretation is divisible by 5.



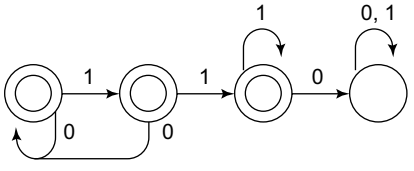
e. All strings that contain the substring 0101.



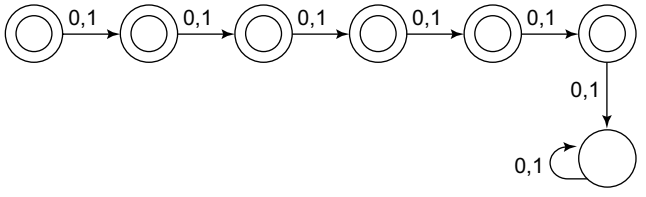
f. All strings that start with 0 and has odd length or start with 1 and has even length.



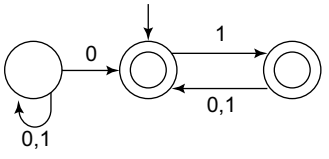
g. All strings that don't contain the substring 110.



h. All strings of length at most 5.

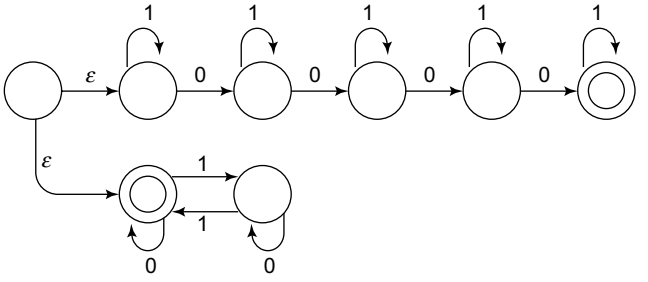


i. All strings where every odd position is a 1.

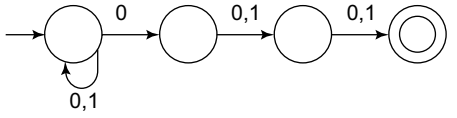


17. Some useful NFAs

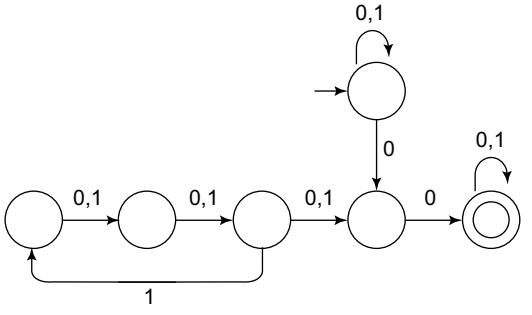
a. All strings containing exactly 4 0s or an even number of 1s. (8 states)



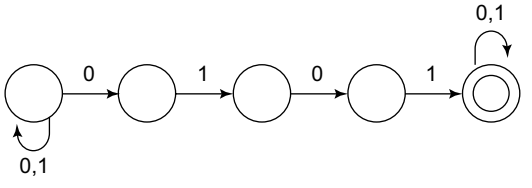
b. All strings such that the third symbol from the right end is a 0. (4 states)



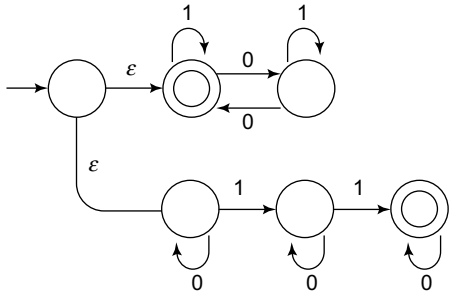
c. All strings such that some two zeros are separated by a string whose length is 4i for some i>=0. (6 states)



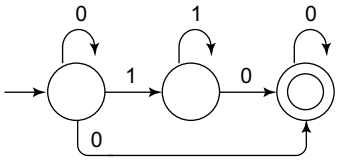
d. All strings that contain the substring 0101. (5 states)



e. All strings that contains an even number of 0s or exactly two 1s. (6 states)



f. The language 0*1*0*0 (3 states)



18. L1: The set of strings where each string w has an equal number of zeros and ones; and any prefix of w has at least as many zeros as ones. L2: The set of strings defined inductively as follows: if w is in the set then 0w1 is also in the set; if u and v are in the set then so is uv; and the empty string is in the set. Prove that every string in L2 is contained in L1.

Answer:

We can analyse L2 inductively to see that it maintains the property of L1 for each case:

1. The empty set. This is a member of L1, since it satisfies the properties directly.
2. Consider the string 0w1. Assuming that w is in L1, we maintain the equal number of 0s and 1s because we add one of each. We also maintain the prefix condition, since the 0 is added before the 1.
3. Consider the concatenated string uv. Assuming that u and v are both in L1, simply concatenating them together will maintain the equal number of 0s and 1s. The prefix condition is slightly more difficult. We consider the following prefixes:
   a. PREFIX(u). Since u is in L1, this must be in L1.
   b. u. Again, since u is in L1, this must be in L1.
   c. uPREFIX(v). Since u has an equal number of 0s and 1s, and v is in L1, this must maintain the prefix property.

19. L1: The set of strings where each string w has an equal number of zeros and ones; and any prefix of w has at least as many zeros as ones. L2: The set of strings defined inductively as follows: if w is in the set then 0w1 is also in the set; if u and v are in the set then so is uv; and the empty string is in the set. Prove that every string in L1 is contained in L2.

Answer: The proof is by induction on the length of strings in L1:

1. The base case is the empty string. This is in L2 by definition.
2. For the inductive step, suppose that all strings in L1 of length <= n are in L2. Let w be a string in L1 of lenght n+1 and suppose it is of the form $A_1A_2...A_{n+1}$, where A_i is either 0 or 1. Let j be the first index with the property that $A_1A_2...A_j$ has the same number of zeros and ones. There are two cases to analyse.
   a. $j < n+1$. Then not only does $u = A_1A_2...A_j$ have the same number of zeros and ones, any prefix of u will have at least as many zeros as ones since it is also a prefix of w. So u is in L1. Let $v = A_{j+1}A_{j+2}...A_{n+1}$. Then v must have the same number of zeros and ones since both u and w=uv do. Also, any prefix x of v cannot have more ones than zeros in it since then ux would be a prefix of w that had more ones than zeros. Therefore, v must be in L1. Since, both u and v are of length $\Leftarrow$ n, by the induction hypothesis they are in L2. Therefore, w = uv must be in L2, by the definition of L2.
   b. $j = n+1$. Then w = 0u1 for some string u, and u has the same number of zeros and ones, since w does. Also, no prefix x of u can have more ones than zeros, since then 0x would either have more ones than zeros which is impossible by hypothesis, or 0x would have the same number of ones as zeros, which is also impossible by since $j = n+1$. Therefore we can conclude that u is in L1, and since it is of length $\Leftarrow$n it is in L2 by the induction hypothesis.

This completes the inductive step, and therefore L1 is contained in L2.

20. Prove that if L1 is regular and L2 is regular then so is L1-L2 (the set of all strings in L1 but not in L2).

Answer: L1-L2 is the same as the intersection of L1 and the complement of L2. Since the set of regular languages is closed under each of these operations, L1-L2 must be regular.

21. If L is regular then prove Prefix(L) is regular where Prefix(L) is the set of all strings which are a proper prefix of a string in L.

Answer: We can construct a DFA to decide Prefix(L) by taking the DFA for L and marking all states from which an accept state is reachable as accept states. So, Prefix(L) must be regular.

22. Prove that Regular Sets are closed under MIN. MIN(R), where R is a regular set, is the set of all strings w in R where every proper prefix of w is not in R. (Note that this is not simply the complement of PREFIX).

Answer: We can construct a DFA to decide MIN(R) by taking the DFA for R and redirecting all outgoing arrows from all the accept states to a dead state. So, MIN(R) must be regular.

23. Prove that Regular Sets are NOT closed under infinite union. (A counterexample suffices).

Answer: Consider the sets {0}, {01}, {0011}, etc. Each one is regular because it only contains one string. But the infinite union is the set $\{0^i1^i \mid i>=0\}$ which we know is not regular. So the infinite union cannot be closed for regular languages.

24. Prove that regular sets are not closed under infinite intersection?

Answer: We know that

$\{0^i1^i \mid i>=0\} = \{0\} \cup \{01\} \cup \{0011\} \cup ...,$

Taking complements and applying DeMorgan's law gives us

$\{0^i1^i \mid i>=0\}^c = \{0\}^c \wedge \{01\}^c \wedge \{0011\}^c \wedge ...,$

Where we are using U to denote union and ^ to denote intersection. Recall the complement of a regular language is regular, and hence the complement of a not-regular language is not regular. So we can conclude that the left hand side of the equation is not-regular, and each term in the intersection is regular. Therefore infinite intersection does not preserve regularity.

25. Prove that the language $\{1^k0^i1^i0^j1^j0^k/i,j,k>0\}$is CFL.

Answer: It is CFL. This can be generated using the following rules.

$S \rightarrow 1A0$

$A \rightarrow 1A0 \mid B$

$B \rightarrow CC$

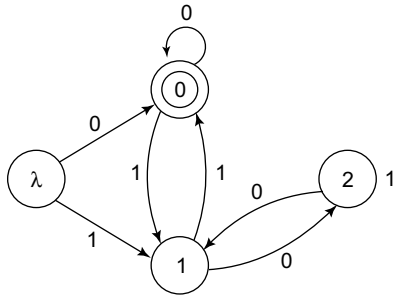
$C \rightarrow 0D1$

$D \rightarrow 0D1 \mid \epsilon$

26. Is the set of binary numbers that are divisible by 3 regular?

Answer: Yes. We can recognize by using the following FA. Thus, it is regular. However, it is very difficult to have regular expression for this.

While drawing FA, the following discussion is very useful. If we know that the input so far was a number that was divisible by 3, then we know that after reading one more 0, the number is again divisible by 3. (This is so because reading that 0 amounted to multiplying the previous number by 2.) What if a 1 had been read instead? This would have amounted to multiplying the previous number by 2 and adding 1. The resulting number, when divided by 3, would yield a remainder of 1. This translates into a piece of a finite-state machine shown in figure.



27. For each of the languages below, indicate the smallest complexity class that contains it. (i.e. Regular, Deterministic Context Free, Turing Machines (Recursive). Assume an alphabet of {0,1} unless otherwise specified. You do not need to prove your answers.

Answer:

a. **DCFL**. $\{0^n1^m0^p1^q \mid n+m = p+q \text{ and } n,m,p,q > 0\}$

b. **DCFL**. $\{0^n1^m0^m1^n \mid n,m > 0\}$

c. **regular** . $\{0^n1^m0^p1^q \mid n,m,p,q > 0\}$

d. **CFL**. The set of strings over alphabet {0,1,2} with an equal number of 0s and 2s or an equal number of 0s and 1s.

e. **regular**. $\{0^m \mid m = 2k+1 \text{ where } k>0\}$

f. **regular**. The set of strings with 3n 0s and 4m 1s for m,n > 0.

g. **DCFL**. The set of strings with at least ten times as many 0s as 1s.

h. **regular**. The set of strings that are either odd length or contain 5 consecutive 1s.

i. **TM**. $\{0^m10^{m!} \mid m>0\}$

j. **DCFL**. The set of stings over alphabet {0,1,2} where the number of 1s equals the number of 2s and every 0 is followed immediately by at least one 1.

28. Is it recursively enumerable or not?

For each of the following languages, state whether the language is or is not recursively enumerable and whether the complement of the language is or is not recursively enumerable. Give some justification for your answers.

a. The language of all TM's that accept nothing.

b. The language of all TM's that accept everything.

c. The language of all TM's that accept Regular sets.

d. The language of all PDA's that accept everything.

e. The language of all CFG's that are ambiguous.

Answer:

a. **no yes** The language of all TMs that accept nothing.

We need to try an infinite number of strings in a TM to determine that it accepts nothing, but we

only need to find a single string that it accepts to show that it accepts something.

b. **no no** The language of all TMs that accept everything.

We need to try an infinite number of string in a TM to determine whether it accepts everything, and we may or may not need to if it does not.

c. **no no** The language of all TMs that accept Regular languages.

We can enumerate all possible regular languages, but testing every regular language against every TM would take forever. If fact, even testing a TM against a single infinite regular language would take forever.

d. **no yes** The language of all PDAs that accept everything.

We would have to try every string before declaring that a PDA accepts everything, but since membership in a CFG (equivalent to a PDA) is decidable, we determine that a PDA does not accept anything once it rejects anything.

e. **yes no** The language of all CFGs that are ambiguous.

We can determine that a CFG is ambiguous by finding a single string which has an ambiguous derivation, but we cannot determine if a CFG is unambiguous unless we try everything string in it.

29. Put the following grammar into Chomsky Normal Form. Show all work.

S →A | AB0 | A1A

A →A0 |ϵ

B → B1 | BC

C → CB | CA | 1B

Answer: Converting to Chomsky Normal Form

Removing all e rules first. The resultant rules are:

S → e | A | AB0 | A1A | B0 | A1 | 1A

A → A0 | 0

B → B1 | BC

C → CB | CA | 1B

Removing unit rules, we have arrived at the following rules.

S → e | **A0** | **0** | AB0 | A1A | B0 | A1 | 1A

A → A0 | 0

B → B1 | BC

C → CB | CA | 1B

Convert remaining rules into proper form

S → e | A0 | 0 | AS_1 | B0 | A1 | 1A

A → A0 | 0

B → B1 | BC

C → CB | CA | 1B

S1 → B0 | 1A

S → e | A$\mathbf{N_0}$ | AS_1 | B$\mathbf{N_0}$ | A$\mathbf{N_1}$ | $\mathbf{N_1}$A

A → A$\mathbf{N_0}$ | **0**

B → B$\mathbf{N_1}$ | BC

C → CB | CA | $\mathbf{N_1}$B

S_1 → B$\mathbf{N_0}$ | $\mathbf{N_1}$A

N0 → 0

N1 → 1

30. Convert the following grammar into an equivalent one with no unit productions and no useless symbols.

S → A | CB

A → C | D

B → 1B | 1

C → 0C | 0

D → 2D | 2

Converts to

S → **0C** | **0** | **2D** | **2** | CB

A → C | D

B → 1B | 1

C → 0C | 0

D → 2D | 2

A is now useless and can be removed.

31. Explain why the grammar below is ambiguous.

S → 0A | 1B

A → 0AA | 1S | 1

B → 1BB | 0S | 0

The grammar is ambiguous because we can find strings which have multiple derivations:

S	S
0A	0A
0 0AA	0 0AA
00 1S 1	00 1 1S
001 1B 1	0011 0A
0011 0 1	00110 1

32. Construct context free grammar to accept the following languages.

a. {w | w starts and ends with the same symbol}

Answer:

S → 0A0 | 1A1

A → 0A | 1A | e

b. {w | |w| is odd}

Answer:

S → 0A | 1

A → 0S | 1S | e

c. {w | |w| is odd and its middle symbol is 0}

Answer:

S → 0 | 0S0 | 0S1 | 1S0 | 1S1

d.{0^n1^n | n>0} U {0^n1^{2n} | n>0}

Answer:

S → 0A1 | 0B11

A → 0A1 | e

B → 0B11 | e

e. {$0^i1^j2^k$ | i!=j or j!=k}

Answer:

S → AC | BC | DE | DF

A → 0 | 0A | 0A1

B → 1 | B1 | 0B1

C → 2 | 2C

D → 0 | 0D

E → 1 | 1E | 1E2

F → 2 | F2 | 1F2

f. Binary strings with twice as many 1s as 0s.

Answer:

S → e | 0S1S1S | 1S0S1S | 1S1S0S

33. Identify whether the following languages are regular or not?

a. {www | w is {0,1}*}

Answer: Not. Assume that the language is regular. Let p be the pumping length, and choose s to be the string $0^p10^p10^p1$. Now we try to break it up into s=xyz. Since |xy| <= p and |y|>0, y can only contain 0's. When we pump the string even just once we get $xy^2z = 0^{p+|y|}10^p10^p1$, and this is not of the form www, since |y| > 0. This contradicts the pumping lemma, so the language is not regular.

b. {0^m1^n | m is not equal to n}

Answer: No. We know that

{0^n1^n | n >= 0} = {0^m1^n | m,n >= 0}^{0*1*}^{c},

where we are using ^ to denote intersection and ^{c} to denote complement. The proof is by contradiction. If {0^m1^n | m is not equal to n} really were regular then {0^n1^n | n >= 0} would also be regular because 0*1* is regular and because of the closure properties of regular sets. Therefore it can't be regular.

There is a direct way to prove it as well: If p is the pumping length and we take the string $s = 0^p1^{p+p!}$, then no matter what the decomposition s = xyz is the string $xy^{1+p!/|y|}z$ will equal $0^{p+p!}1^{p+p!}$ which is not in the language.

c. {$0^m1^n0^m$ | m,n >= 0}

Answer: No. Assume that the language is regular. Let p be the pumping length, and choose s to be the string 0^p10^p. Now we try to break it up into s=xyz. Since |xy| <= p, y can only have zeros in it. Now $xy^jz = 0^{p+ (j-1) |y|}10^p$, and since |y|>0 the number of 0's on the left and right sides of xy^jz will not be the same for any j>1 so xy^jz will not be in the language, contradicting the pumping lemma. Therefore {$0^m1^n0^m$ | m,n >= 0} is not regular.

d. Regular. The set of strings that have an even number of double zeros in them.

This can be decided by a DFA.

e. The set of all strings of the form xwx^R where x and w are strings over the alphabet {0,1}.

Answer: Regular. This description is somewhat misleading. Since w can represent any string and x can be 0 or 1, this is the same as all strings which begin and end with the same character. This is easily seen to be a regular language.

f. The set of all strings over the alphabet {0} whose length is n! for some n > 0.

Answer: No. Assume that the set is regular. Let p be the pumping length. Without loss of generality we can assume that p is at least 2. (We can always increase the pumping length. We only do this because some of our calculations donot work for p = 1) Then, according to the pumping lemma, we can break the string $s=0^{p!}$ in to s=xyz where y has positive length and |xy| <= p. Then $s=xy^2z = 0^{p! + |y|}$ must also be in the language, so p!+|y| must also be a factorial. But (p+1)!-p! = (p)p! > p >= |y| so it follows that p! + |y| < (p+1)!, and therefore p! + |y| is not a factorial. This is a contradiction so the language cannot be regular.

g. The set of strings over the alphabet {0} of the form 0^n where n is not prime.

Answer: No. To prove this language is not regular, we instead examine the complement because the set of regular languages is closed under complement. Assume that the set is regular. Let p be the pumping lenght of the language. Then, according to the pumping lemma, we break the string $s=0^p$ into s=xyz where y has positive length. Then, $s=xy^iz = 0^{p + (i-1) |y|}$ must also be in the set for any i. In particular let i = p+1. Then $xy^{p+1}z=0^{p+p|y|}$ must be in the set so p + p|y| = p(1 +|y|) must be prime. Thus we have a contradiction and the set cannot be regular.

34. Prove that the language L={$0^{n!}$/n>=0} is not regular.

Answer: Assume that this language is regular and that the pumping constant is *n*. Then choose the string *w* = $0n^!$. Then the pumping lemma tells us that we can write this as *xyz* with the three conditions satisfied. The condition that *y* is not the empty string and |*xy*| ≤ *n*, implies that *y* must be 0*j* where *j* ∈ {1, 2, .., *n*}. The pumping lemmas says that then xy^2z should be in *L*. But $xy^2z = an^{!+}j$. But if we add 1 to *n* to *n*! we do not get another number which is a factorial. Thus, the language is not regular.

35. Prove that the language L^1={$a^ib^jc^k$/j<i,j<k} is not context free.

Answer: Assume that language is CFL and take pumping length of p. Consider a string s=$a^{p+1}b^pc^{p+1}$. Applying pumping lemma, that taking s as uvxyz such that |vxy|<=p and|vy|>=1. As, |vxy|<=p, vy cannot have a's, b's and c's. Also, it cannot have equal number of a's or b's or c's. May be there can be two other possibilities.

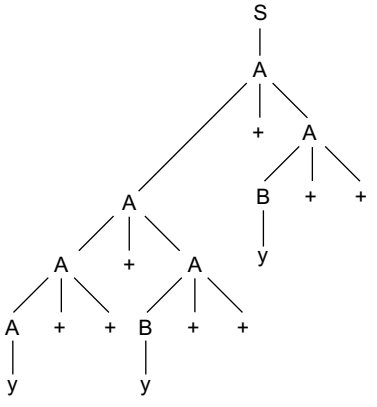
Case 1 : If *vy* has more *a*'s or more *c*'s than *b*'s, then choose *s′* = *uxz* and observe that *s′* has at most as many *a*'s or at most as many *c*'s as *b*'s (because more *a*'s or more *c*'s than *b*'s were removed).

Case 2 : If *vy* has more *b*'s or more *a*'s than *c*'s, then choose $s' = uv^2xy^2z$ and observe that *s′* has at most as many *b*'s as *a*'s or *c*'s (because more *b*'s were added tgab *a*'s or *c*'s).

In both cases s′ ∉ L_1, contradicting the pumping lemma. We conclude that L_1 is not context free.

36. Compiler Construction

37. For the following grammar, Draw the parser tree for the input "y + + + y + + + y + +"



S → A

A → A+A| B++

B → y

38. S → abcS| Sabc| h

Describe the language constructed by the grammar.

Answer: Zero or more instances of abc including only one h. h appears only at the beginning, at the end or between c and a.

Is it ambiguous? Why?

YES, there are two parse trees for the sentence — abchabc

If it is ambiguous, write an unambiguous context-free grammar that describes the same language.

S → L h L

L → L N | L

N → abc | ε

39. Write an unambiguous context-free grammar that generates all strings of a and b.

Answer: List → List char | char

char → a | b

40. Describe the language denoted by the regular expression: ((ε|0) l*)*

Answer: All binary numbers

41. Write a regular expression that derive strings of a and b. These strings contain at least two b's.

Answer:

(a|b)* b (a|b)* b (a|b)*

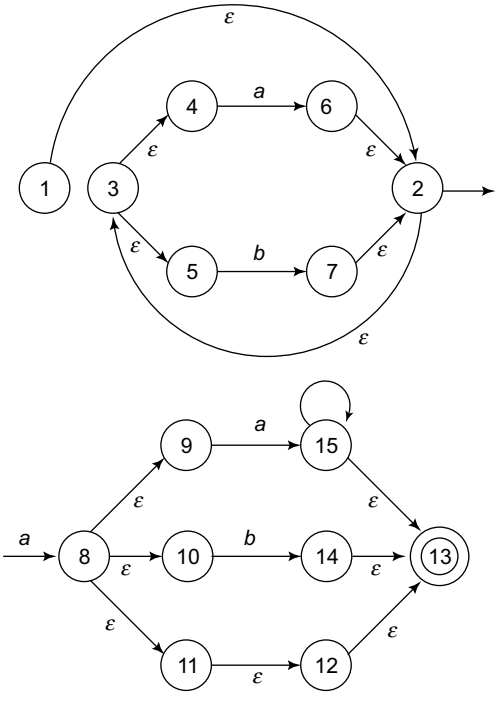
(a* b a* b a*)*(a|b)*

42 Write a regular expression that derive strings of a and b. These strings contain at most two b's.

Answer:

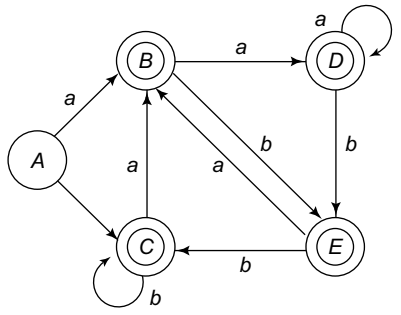
a* (ε |b) a* (ε |b) a*

43. Construct the DFA from the following NFA



Answer:

A	{1, 2, 3, 4, 5}	a	b
B	{2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13}	D	E
C	{2, 3, 4, 5, 7}	B	C
D	{2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 15}	D	E
E	{2, 3, 4, 5, 7, 13, 14}	B	C



44. Write an unambiguous context-free grammar for the language of strings like a = b = c

Note: equality operator is a right-associative operator

Answer:

Right → Letter = Right | Letter

Letter → a | b| … | z

45. Write an unambiguous context-free grammar for the language of right-associative lists of identifiers separated by commas.

Answer:

List → id, List | id

46. Write a regular expression that derive strings of a and b. These strings should have odd number of b's.

Answer:

a* b (a* b a* b)* a*

47. Write a regular expression that derive strings of a and b. These strings contain at least two b's.

Answer:

(a|b)* b (a|b)* b (a|b)*

48. Write a regular definition for Pascal unsigned numbers which are strings such as 342, 43.08, 56.56E4, 76.8E-20 or 23E33.

Answer:

Digit → 0|1|…|9

Num → Digit+ (. Digit+)? (E (+|–)? Digit+)?

49. Write a regular definition for the following language: all strings of letters that contain the five vowels in order.

Vowels letters: a, e, i, o, u, A, E, I, O, U.

Answer:

Letter → {All letter capital and small except vowel letters}

S → Letter* (a|A) Letter* (e|E) Letter* (i|I) Letter* (o|O) Letter* (u|U)

50. Is the following grammar ambiguous? Why?

S → a S b S | b S a S | ϵ

Answer:

Yes, a b a b has 2 parse trees, S → a S b S → a b S a S b S → a b a b

S → a S b S → a S b a S b S → a b a b

51. Is the following grammar ambiguous?

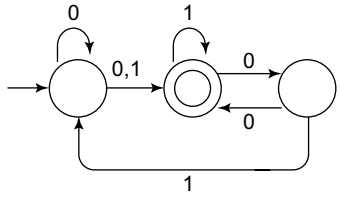
stmt → if expr then stmt | matched_stmt

matched_stmt → if expr then matched_stmt else stmt | other

Answer: Yes. The following is derived more than one way.

if expr then matched_stmt else if expr then stmt

52. Consider the following NFA over the alphabet {0,1}:



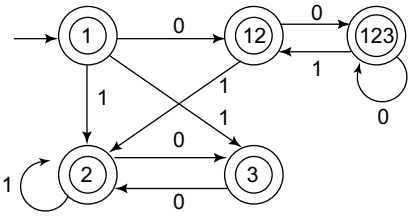
a. Convert this NFA to a minimal DFA.

b. Write a regular expression for the set that machine accepts.

c. Write a linear grammar where each right side is of the form aB or a. ("a" a terminal and "B" a nonterminal) to generate the set.

Answer:

a. Equivalent NFA to the given DFA is:



b. Regular expression for the set that machine accepts is given as:

[0+(0+1)(1+00)*01]*(0+1)(1+00)*

c. Linear grammar where each right side is of the form aB or a.

A → 0A | 0B | 1B

B → 1B | 0C | e

C → 0B | 1A

53. Which of the following languages are regular? If the language is regular, present a finite automaton or regular expression. If not, give a proof using the pumping lemma.

(a) The set of all 0-1 strings in which the total number of zeros to the right of each 1 is even.

Answer: Regular as we can represent them using regular expression 0*(1*00)*1*

(b) The set of all 0-1 strings that contain more 1s than 0s.

Answer: Not regular. Use pumping lemma on string 0^m1^{m+1}, where m is the pumping constant.

(c) The set of all 0-1 strings of the form 0^m1^n where m is odd and n is even.

Answer: Regular as we can represent the same using regular expression 0(00)*(11)*.

(d) The set of all 0-1 strings in which the number of occurrences of "000" and of "111" are the same. (Note that the string "1110000111" contains two occurrences of each.)

Answer: Not regular. Using pumping lemma, one can prove.

(e) Prove L={a^{n^3}/n>=1} is not regular?

Answer: Using pumping lemma, we can prove this. Acceptable strings in this language are: a, a^8, a^{27}, a^{64} etc. Consider the string aaaaaaaa and pumping width as 2. and x=a, y=a and z=aaaaaa. We have xy length is <=2. Now, consider the string xy^jz for j value of 5. The resulting string is aaaaaaaaaa which is not in the language. Thus, it is not regular.

(f) Prove that the language L={$a^mb^nc^k$/m,n,k>=0, m ≠ n,m≠k,n≠k}is not regular.

Answer: Use pumping lemma and prove.

54. What is the language that is represented by the following rules?

S→ aSb|aSbb|ϵ

Answer: {a^nb^m/m/2<=n<=m}

55. What is the language that is represented by the following rules?

S → aSa|bSb|aa|bb

Answer: {ww^R/|w|>=1},where R indicates reverse.

56. What is the language that is represented by the following production rules?

S →aaSb|aS|ϵ

Answer: { $a^{i+j}b^j$|i>=j>=0}

57. Let L_1 and L_2 be languages in the respective language class, and let R be a regular language, and x be a given word over alphabet Σ. The following table categorises following decision problems as (D) decidable, (U) undecidable.

Language class/ Problem	regular	context-free	recursive	r.e.
$L_1 \cup L_2 = \Sigma^*$?	D	U	U	U
$x \in L_1$?	D	D	D	U
$R \subseteq L_1$?	D	U	U	U
$L_1 - R = \phi$?	D	D	U	U
$\exists y \in L_1, \lvert y\rvert \leq 5$?	D	D	D	U

($|y|$ **denotes the length of** y.)

58. If L is regular then L′={x/ ax ϵ L or bxϵ L} is also regular.

Answer: As L is regular, then L_1'={x:ax ϵ L} and L_2' ={x:xx ϵ L} are regular. As union of two regular languages are regular, L′ which is union of L_1' and L_2' is also regular.

59. If L is regular, prove that $L' = \{xx- \mid x \in L\}$ is not regular. Here $x-$ is x without its last symbol, e.g. $(bab)- = ba$, $b- = \epsilon$. (We let $\epsilon - = \epsilon$.)

Answer: Consider a language that is represented by the regular expression a*b. Then L› is the language {a^nba^n}. Let us use pumping lemma to check whether L′ is regular or not. Assume that the pumping length is n and the string w= a^nba^n is in the language. For any partition of w=xyz such that |xy|<=n and|y|>1, xy^2z will be of the form a^mba^n where m>n. That is, xy^2z will not be in L′. So, L′ is not regular.

60. If L is regular, then $L' = \{x \mid xy \in L$ for some string $y\}$ is regular.

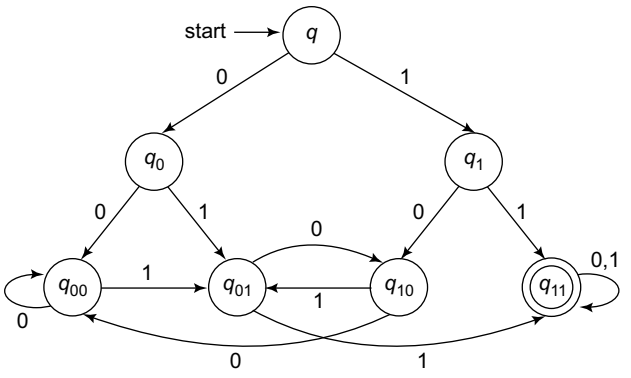
Answer: Yes. We can simply prove by adding some more states to the DFA.

61. Prove using pumping lemma whether the following language L is CFG or not.

{w $|\exists\ i, j \geq 0,\ w = a^i b^j c^i d^j$}

Answer: L is not context-free: Let p be a proposed pumping lemma constant, and let $w = apbpcpdp \in L$. If $w = uvxyz$ with $|vxy| \leq p$ and $|vy| \geq 0$, then assume without loss of generality that v only contains an a or a b (or both) (as the cases of v containing c or d is analogous, as is the case of y containing a particular symbol). If v contains an a, then y does not contain a c. On the other hand, if v contains a b, then y does not contain a d. In either case, uv^2xy^2z L, as either the number of a's increases while the number of c's does not, or the number of b's increases while the number of d's does not.

62. Minimise the following DFA.

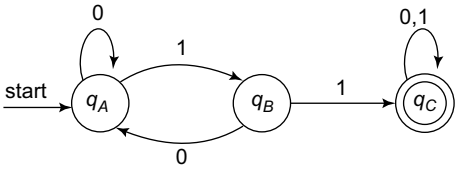


Answer:

To obtain the minimised DFA, we mark all pairs of distinguishable states, First, every state (but q_{11}) is distinguishable from q_{11} because it is an accepting state. Then, on transition 1, we note that q_1 and q_{01} go to q_{11}, while q, q_0, q_{00} and q_{10}, go to a state distinguishable from q_{11}. So these are all distinguishable pairs. At this paint we have following pairs of distinguishable states.

q_0						
q_1	x	x				
q_{00}			x			
q_{01}	x	x		x		
q_{10}			x		x	
q_{11}	x	x	x	x	x	x
	q	q_0	q_1	q_{00}	q_{01}	q_{10}

No, more pairs of states can be distinguished. The indistinguishable states split into three classes: Class A consisting of states {q, q_0, q_{00}, q_{10}}; class B, consisting of (q_1, q_{01}}, and class C, consisting of q_{11}. This yields the following minimized DFA.



63. What are the possible error recovery actions in lexical analysis:

Answer:

a. Deleting an extraneous character
b. Inserting a missing character
c. Replacing an incorrect character by a correct character
d. Transposing two adjacent characters

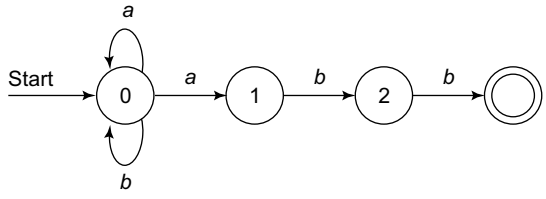
64. Write the algorithm for simulating a DFA.

Answer:

```
s := s0;
c := nextchar
while c ≠ eof do
    s := move(s,c)
    c := nextchar
end
if s is in F then
    return "yes"
    else return "no";
```

65. Write the transition graph for an NFA that recognises the language (a|b)*abb.

Answer :



66. Name some variety of intermediate forms.

Answer:

- Postfix notation or polish notation.
- Syntax tree
- Three address code
- Quadruple
- Triple

67. What are the techniques behind code optimisation phase?

Answer:

a. Constant folding
b. Loop constant code motion
c. Induction variable elimination
d. Common sub expression elimination
e. Strength reduction
f. Mathematical identities

68. Write the syntax for three-address code statement, and mention its properties.

Answer :

Syntax: A= B op C

- Three-address instruction has at most one operator in addition to the assignment symbol. The compiler has to decide the order in which operations are to be done.
- The compiler must generate a temporary name to hold the value computed by each instruction.
- Some three-address instructions can have less than three operands.

69. What is linear analysis?

Answer :

Linear analysis is one in which the stream of characters making up the source program is read from left to right and grouped into tokens that are sequences of characters having a collective meaning. This is also called as lexical analysis or scanning.

70. What is a symbol table?

Answer :

A symbol table is a data structure containing a record for each identifier, with fields for the attributes of the identifier. The data structure allows us to find the record for each identifier quickly and to store or retrieve data from that record quickly. Whenever an identifier is detected by a lexical analyser, it is entered into the symbol table. The attributes of an identifier cannot be determined by the lexical analyser.

71. What is the back-end phases of a compiler?

Answer :

The back end of compiler includes those portions that depend on the target machine and generally those portions do not depend on the source language, just the intermediate language. These include

- Code optimisation
- Code generation, along with error handling and symbol-table operations.

72. List the various error recovery strategies for a lexical analysis.

Answer :

- Panic mode recovery
- Deleting an extraneous character
- Inserting a missing character
- Replacing an incorrect character by a correct character
- Transposing two adjacent characters

73. What is an operator precedence parser?

Answer :

A grammar is said to be an operator precedence if it possess the following properties:

1. No production on the right side is ε.
2. There should not be an any production rule possessing two adjacent non terminals at the right hand side.

Advantages

This type of parsing is simple to implement.

Disadvantages

1. The operator like minus has two different precedence (unary and binary). Hence it is hard to handle tokens like minus sign.
2. This kind of parsing is applicable to only small class of grammar.

74. What are the properties of LR parser?

Answer :

1. LR parsers can be constructed to recognise most of the programming languages for which the context free grammar can be written.
2. The class of grammar that can be parsed by LR parser is a superset of class of grammars that can be parsed using predictive parsers.
3. LR parsers work using non backtracking shift reduce technique yet it is efficient one.

Types of LR parsers

- SLR parser- simple LR parser
- LALR parser- lookahead LR parser
- Canonical LR parser

75. What are the problems with top down parsing?

Answer :

- Backtracking
- Left recursion
- Left factoring
- Ambiguity

76. Explain FIRST and FOLLOW algorithm.

Answer :

FIRST

1. If X is terminal, then FIRST(X) IS {X}.
2. If $X \rightarrow \varepsilon$ is a production, then add ε to FIRST(X).
3. If X is non terminal and $X \rightarrow Y1,Y2..Yk$ is a production, then place a in FIRST(X) if for some i , a is in FIRST(Yi) , and ε is in all of FIRST(Y1),… FIRST(Yi-1);

FOLLOW

1. Place $ in FOLLOW(S),where S is the start symbol and $ is the input right endmarker.
2. If there is a production $A \rightarrow \alpha B\beta$, then everything in FIRST(β) except for ε is placed in FOLLOW(B).
3. If there is a production $A \rightarrow \alpha B$, or a production $A \rightarrow \alpha B\beta$ where FIRST(β) contains ε , then everything in FOLLOW(A) is in FOLLOW(B).

77. What is handle? Explain about handle pruning.

Answer :

A handle of a string is a substring that matches the right side of a production, and whose reduction to the nonterminal on the left side of the production represents one step along the reverse of a rightmost derivation.

A handle of a right – sentential form γ is a production of A→β and a position of γ where the string β may be found and replaced by A to produce the previous right-sentential form in a rightmost derivation of γ. That is , if S ⇒ αAw ⇒ $\alpha\beta$w, then A→β are in the position following α is a handle of $\alpha\beta$w.

A rightmost derivation in reverse can be obtained by handle pruning.

If w is a sentence of the grammar at hand, then w = γn, where γn is the nth right-sentential form of some as yet unknown rightmost derivation

S = γ0 ⇒ γ1…⇒ γn–1 ⇒ γn = w

78. What is meant by viable prefixes?

Answer :

The set of prefixes of right sentential forms that can appear on the stack of a shift-reduce parser are called viable prefixes. An equivalent definition of a viable prefix is that it is a prefix of a right sentential form that does not continue past the right end of the rightmost handle of that sentential form.

79. What is phrase level error recovery?

Answer :

Phrase level error recovery is implemented by filling in the blank entries in the predictive parsing table with pointers to error routines. These routines may change, insert, or delete symbols on the input and issue appropriate error messages. They may also pop from the stack.

80. What are the various types of intermediate code representation?

Answer:

- Syntax tree
- Posix
- Three address code

81. Explain about backpatching along with the common functions used in it.

Answer :

Backpatching is the activity of filling up unspecified information of labels using appropriate semantic actions in during the code generation process. In the semantic actions the functions used are mklist(i),merge_list(p1,p2) and backpatch(p,i)

Functions that are used in backpatching are:

- mklist(i) creates the new list. The index i is passed as an argument to this function where I is an index to the array of quadruple.
- merge_list(p1,p2) this function concatenates two lists pointed by p1 and p2. It returns the pointer to the concatenated list.
- backpatch(p,i) inserts i as target label for the statement pointed by pointer p.

82. What are the various methods of implementing three address statements?

Answer :

- Quadruple : a structure with atmost four fields such as operator(OP),arg1,arg2,result.
- Triples : the use of temporary variables is avoided by referring the pointers in the symbol table.
- Indirect triples : the listing of triples has been done and listing pointers are used instead of using statements.

83. What is a flow graph?

Answer :

A flow graph is a directed graph in which the flow control information is added to the basic blocks.

- The nodes to the flow graph are represented by basic blocks.
- The block whose leader is the first statement is called initial block.
- There is a directed edge from block B1 to block B2 if B2 immediately follows B1 in the given sequence. We can say that B1 is a predecessor of B2.

84. What is a DAG? Mention its applications.

Answer :

Directed acyclic graph(DAG) is a useful data structure for implementing transformations on basic blocks. DAG is used in

- Determining the common sub-expressions.
- Determining which names are used inside the block and computed outside the block.
- Determining which statements of the block could have their computed value outside the block
- Simplifying the list of quadruples by eliminating the common su-expressions and not performing the assignment of the form x := y unless and until it is a must.

85. Define peephole optimisation along with its characteristics.

Answer :

Peephole optimisation is a simple and effective technique for locally improving target code. This technique is applied to improve the performance of the target program by examining the short sequence of target instructions and replacing these instructions by shorter or faster sequence.

- Redundant instruction elimination
- Flow of control optimisation
- Algebraic simplification
- Use of machine idioms

86. What are the basic goals of code movement?

Answer :

To reduce the size of the code i.e. to obtain the space complexity.

To reduce the frequency of execution of code i.e. to obtain the time complexity.

87. What is code motion?

Answer :

Code motion is an optimisation technique in which amount of code in a loop is decreased. This transformation is applicable to the expression that yields the same result independent of the number of times the loop is executed. Such an expression is placed before the loop.

88. What are the steps involved in Non Recursive predictive parsing?

Answer :

- Input buffer is filled with input string with ∃ as the right end marker.
- Stack is initially pushed with ∃
- Construction of Parsing Table T
- Parsing by parsing routine

89. What is LL(1) grammar?

Answer :

A grammar 'G' whose parsing table has no multiply defined entries, can be called as LL(1) grammar.

90. What is a Shift - Reduce Parser?

Answer :

It is a bottom up parser. The parsing is done from the leaves to the root. The parse tree is constructed from the bottom to top, for an input string.

91. What are the demerits of SLR?

Answer :

- It will not produce uniquely defined parsing action tables for all grammars.
  - Shift-Reduce conflict.

92. Why LR parsing is good and attractive?

Answer :

- LR parsers can be constructed for all programming language constructs for which CFG can be written.
- LR parsing is Non-backtracking Shift-Reduce parsing.
- Grammars parsed using LR parsers are super set of the class of grammar.
- LR parser can detect syntactic error as soon as possible, when left-to-right
- Scan of the input.

93. What are the demerits of LALR parser.

Answer :

- Merger will produce reduce / reduce conflict.
- On erroneous input, LALR parser may proceed to do some reductions
- After the LR parser has declared an error, but LALR parser never shift a symbol after the LR parser declares an error.

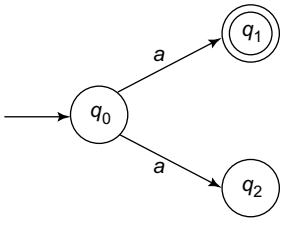
94. What is Data flow engine?

Answer :

Much of the information needed to perform good code optimisation involves "data flow analysis," the gathering of information about how values are transmitted from one part of a program to each other part. This is handled by the data flow engines.

OBJECTIVE TYPE QUESTIONS

1. The language recognised by the DFA which is formed by complementing the states (accepting or non-accepting) of the nodes of the following DFA



A. $\in$ B. $\in^*$

C. a D. {a, ϵ}

2. Equivalent to the regular expression

0* (a ∪ b) ∪ 0b*ucabb.

A. ϵ (a ∪ b) ∪ ϕ ∪ B. a ∪ bb

C. a ∪ b ∪ b D. None

3. The language $\{a^n b^n / 3 \geq n \geq 1\}$ is not regular. (Y/N)

Answer: N. It is regular. We can have a DFA to accept such a language.

4. Minimum number of nodes in a DFA that recognizes strings over {a,b} with length mod 3 = 0.

A. 4 B. N

C. 3 D. 2

5. Number of nodes in an FS machine that recognises odd length strings over S that contains any number of alphabets

A. 2 B. 3

C. 4 D. None

6. $\{ai|i$ is prime$\}$ is not context free. (Y/N)
7. $\{(anb)n|n >= 1\}$ is context free.(Y/N)
8. $\{(anb)m|m, n >= 1\}$ is context free. (Y/N)
9. If $L1$ is context free and $L2$ is regular, then $L1/L2$ is context free. (Note that $L1/L2 = \{x \mid 3\ y \in L2, xy \in L1\}$) (Y/N)
10. If $L1/L2$ and $L1$ are context free, then $L2$ must be recursive. (Y/N)
11. If $L1$ and $L1 \cap L2$ are context free, then $L2$ must be context free. (Y/N)
12. If $L1$ is context free and $L2$ is regular, then $L1 - L2$ is context free. (Y/N)
13. If $L1$ is regular and $L2$ is context free, then $L1 - L2$ is context free. (Y/N)
14. If $L1$ is regular and $L2$ is context-free, then $L1 \setminus L2$ must be a CFL. (Y/N)
15. If $L1$ and $L2$ are CFLs, then $L1 \cap L2$ must be a CFL. (Y/N)
16. If L is context free, then LR $(=\{x^R|x \in L\})$ is also context free. (Y/N)
17. Nondeterministic and deterministic versions of PDAs are equivalent. (Y/N)
18. If a language L does not satisfy the conditions stated in the pumping lemma for CFLs, then L is not context-free. (Y/N)
19. Every infinite set of strings over a single letter alphabet Σ $(=\{a\})$ contains an infinite context free subset. (Y/N)
20. Every infinite context-free set contains an infinite regular subset. (Y/N)
21. A language can be accepted by a nondeterministic pushdown automation if it can be generated by a context-free grammar. (Y/N)
22. Right-linear grammar are special cases of context-free grammar. (Y/N)
23. If both L and $\overline{L}$ are context-free, then L must be regular. (Y/N)
24. There is a language L which is context-free but not regular such that $\overline{\overline{L}}$ is also context-free. (Y/N)
25. $\{xxxx|x \in \{0, 1\}^*\}$ can be accepted by a deterministic 2-counter machine. (Y/N)
26. Given a TM M whose tape head can move left, right, or stay stationary, the problem of determining whether M ever executes a stationary move is un-decidable. (A stationary move is a transition without moving the tape head.) (Y/N)
27. Given a TM M, the problem of determining '$L(M) = ;$?' is un-decidable.(Y/N)
28. Given two languages $L1$ and $L2$, if $L1 \cdot m\ \overline{L}2$, then $\overline{L}1 \cdot m\ L2$. ($\cdot m$ denotes many-one reduction.) (Y/N)
29. If $L1$ and $L2$ are r.e., so is $L1L2$. (Y/N)
30. If $L1$ and $L2$ are recursive, so is $L1 - L2$. (Y/N)
31. The language $\{< M, x >|$ TM M does not accept input $x\}$ is r.e. ($< M, x >$ denotes the encoding of the pair M, x.) (Y/N)
32. There exists a language L such that L is context free but $\overline{L}$ is not recursive. (Y/N)
33. Given a PDA M, the problem of determining whether M accepts an infinite language is decidable. (Y/N)
34. Given two PDA $M1$ and $M2$, the problem of determining whether $L(M1) \setminus L(M2) =$ is decidable. (Y/N)
35. Given two regular languages $L1$ and $L2$, the problem 'Is $L2 - L1 = ;$?' is un-decidable. (Y/N)
36. Let $L1$ be regular and $L2$ recursively enumerable. Then $L1 \setminus L2$ is always recursive. (Y/N)
37. The union of infinitely many recursive languages is an r.e. language. (Y/N)
38. Given an input x and a multi-tape DTM M, the problem of determining whether M ever reads x's rightmost symbol is decidable. (Y/N)
39. The family of languages accepted by deterministic TMs is closed under complement. (Y/N)
40. Given a TM M and an input w, the problem of determining whether M (on input w) enters some state more than 100 times is decidable. (Y/N)
41. Every infinite subset of an infinite non-regular language is non-regular. (Y/N)
42. If $L1$ and $L2$ are context-free languages, then $L1 \setminus L2$ must be a recursive language. (Y/N)
43. If $L1$ and $L3$ are r.e. languages and $L1 - L2 = L3$, then $L2$ must be an r.e. language. (Y/N)
44. Given a recursive language L, the problem of determining whether $L = ;$ is decidable. (Y/N)
45. $\{< M > |L(M)$ is regular, M is a TM$\}$ is recursive. ($< M >$ denotes the encoding of TM M.) (Y/N)
46. $\{L \mid L = \Sigma^*$ or $L = \Phi\}$ is a trivial property of r.e. sets. (Y/N)
47. Given a TM M and a symbol $x\ 2$ §, it is decidable whether M (starting on a blank tape) ever writes x on its tape. (Y/N)
48. Every primitive recursive function is a total function. (Y/N)
49. Ackermann's function is a total recursive function. (Y/N)
50. Nondeterministic 1-counter machines are less powerful than deterministic 2-counter machines. (Y/N)

51. Given a context-free language $L1$ and a recursive language $L2$, it is un-decidable whether $L1$ C $L2$. (Y/N)

52. If L and LR (the reversal of L) are both in r.e., then L must be recursive. (Y/N)

53. Given a recursive set L and a regular set R, it is decidable whether L C R. (Y/N)

54. Given a recursive set L and a regular set R, it is decidable whether R C L. (Y/N)

55. Given a nondeterministic finite automaton M it is decidable whether the language accepted by M is finite or not. (Y/N)

56. Given a left-linear grammar G, it is decidable whether $L(G) = \Sigma^*$.(Y/N)

57. Recursive languages are closed under Kleene star (i.e., if L is recursive, so is L^*).(Y/N)

58. Recursively enumerable languages are closed under Kleene star. (Y/N)

59. The function $f(n) = 2f^{(n-1)}$, $n>= 1; f(0) = 1$ is primitive recursive. (Y/N)

60. For every language L *subset of* 0^*, L is always r.e. (Y/N)

61. Every total function $f : N \rightarrow N$ is a recursive function. (f is total if $f(x)$ is defined for every $x \epsilon N$.) (Y/N)

62. With respect to a given input, checking whether a C program terminates or not is decidable. (Y/N)

63. The language $\{a^n b^m c^n d^m \mid m, n >= 1\}$ can be accepted by a deterministic TM in polynomial time (i.e., in P). (Y/N)

64. $\{(a^i b^i)j \mid i, j \epsilon N\}$ is in P. (Y/N)

65. The class of NP languages is closed under intersection. (Y/N)

66. The class of NP languages is closed under concatenation. (Y/N)

67. Given a context-free grammar G and a word x, the problem 'Is $x \epsilon L(G)$?' is NP complete. (Y/N)

68. If $\{ww^R \mid w \epsilon \Sigma^*\}$ is solvable in polynomial time, then $P = NP$. (w^R denotes the reversal of word w.)(Y/N)

69. If $L2$ *subset* $L1$, and $L2$ is NP-hard, then $L1$ must be NP-hard as well. (Y/N)

70. A PDA with two stacks can recognise language $\{0^n 1^n 2^n / n>=0\}$. (Y/N)

71. For any language L, there are infinitely many different grammars G such that $L(G) = L$. (Y/N)

72. If L is a CFL and R is a regular language, then $R - L$ is a CFL.(Y/N)

73. If some word w in $L(G)$ has two different derivations, then G is ambiguous.(Y/N)

74. If L is not context-free, then LR is not context free either (where R is the reversal operator).(Y/N)

75. $L = \{0^n 1^m 0^m / n + m = 3 \ mod \ 5\}$ is context-free but not regular. (Y/N)

76. $L=\{a^i b^j c^k \mid 0 < i < j < k\}$ is context-free.(Y/N)

77. $L = \{0^n 1^m 0^n \mid n < 12 < m\}$ is regular. (Y/N)

78. If L is context-free and R and S are regular, then $MAJORITY(L,R,S)=\{w \mid w$ is in at least two of $R, L, S\}$ is also context-free. (Y/N)

Answer : Modify the PDA ML that accepts L to simultaneously simulate R and S. Accept if at least two accept.

79. The minimum spanning tree problem is NP-complete. (Y/N)

80. If some NP-complete language is solvable in polynomial time, then the PCP problem becomes solvable (i.e., recursive). (Y/N)

81. Given a TM M, 'M never moves its head left on the blank tape' is a nontrivial property of r.e. sets. (Y/N)

82. Given a TM M and an input x, it is decidable whether M never reads a blank symbol during the course of its computation on input x. (Y/N)

83. Given a TM M and an input x, it is decidable whether M ever visits a given state more than 10 times. (Y/N)

84. Every primitive recursive function is a total function. (Y/N)

85. Ackermann's function is a partial recursive function. (Y/N)

86. Deterministic PDA are less powerful than nondeterministic PDA. (Y/N)

87. If L and $\overline{L}$ are both in r.e., then L must be recursive. (Y/N)

88. Given an r.e. set L and a regular set R, it is decidable whether L *subset* R. (Y/N)

89. Given an r.e. set L and a regular set R, it is decidable whether R *subset* L. (Y/N)

90. Given a PDA M it is decidable whether the language accepted by M is finite or not. (Y/N)

91. If some NP-complete language is solvable in polynomial time, then NP=co-NP. (Y/N)

92. X–Every infinite r.e. set contains an infinite context-free subset. (Y/N)

93. The halting problem is NP-hard. (Y/N)

94. If P=NP, then $DTIME(n^2) = DTIME(2^n)$. (Note: DTIME denotes deterministic time.) (Y/N)

95. The PCP language (the language associated with the Post correspondence problem) is in r.e. (Y/N)

96. Given a CFG G in Chomsky Normal Form, it is decidable whether $L(G)$ Σ^*.(Y/N)

97. Every r.e. language can be accepted by a deterministic 2-counter machine. (Y/N)

98. There exists a language L *subset* 0^* which is not in r.e. (Y/N)

99. There exists a total function $f : N \to N$ which cannot be computed by any Turing machine. (f is total if $f(x)$ is defined for every $x \in N$.) (Y/N)

100. The language whose production rules are given as:

$S \to XY \mid W$

$X \to aXb \mid \epsilon$

$Y \to cY \mid \epsilon$

$W \to aWc \mid Z$

$Z \to bZ \mid \epsilon$

A. $\{a^ib^jc^k/i,j,k>=0\}$

B. $\{a^ib^jc^k/i,j,k>=1\}$

C. $\{a^ib^jc^k/i,j,k>=0, i=j \text{ or } i=k\}$

D. $\{a^ib^jc^k/i,j,k>=0, i=j \text{ or } i!k\}$

101. Prove that the language $L=\{w|\ni i,j>=0, w=a^ib^jc^jd^i\}$ is context free. (Y/N)

Answer: The above language can be generated using the following production rules. If we observe the rules, all the left hand sides are having single non-terminals. Thus, it is context free language.

$S \to aSd|T$

$T \to cTd|\epsilon$

102. Let R be a regular language and L be a context-free language, prove that $R||L$ is also context-free. (|| denotes the shuffle operator.) (Y/N)

103. The language $\{0^n0^{2n}0^{3n}/n>=0\}$ is

A. O^* B. $\{0\}^*$

C. $\{000^*\}$ D. $\{000000^*\}$

104. $L \leq_m \{0^n1^n/n>=0\}$ then

A. L is CFG B. L is regular

C. L is recursive D. L is context sensitive

ANSWER KEY

1. D	**2.** A	**3.** N	**4.** C
5. A	**6.** Y	**7.** N	**8.** N
9. Y	**10.** N	**11.** N	**12.** Y
13. N	**14.** Y	**15.** Y	**16.** Y
17. N	**18.** Y	**19.** N	**20.** N
21. Y	**22.** Y	**23.** N	**24.** Y
25. Y	**26.** Y	**27.** Y	**28.** Y
29. Y	**30.** Y	**31.** N	**32.** N
33. Y	**34.** N	**35.** N	**36.** N
37. N	**38.** N	**39.** N	**40.** Y
41. N	**42.** Y	**43.** N	**44.** N
45. N	**46.** N	**47.** N	**48.** N
49. Y	**50.** Y	**51.** N	**52.** N
53. N	**54.** N	**55.** Y	**56.** Y
57. Y	**58.** Y	**59.** Y	**60.** N
61. N	**62.** N	**63.** Y	**64.** Y
65. Y	**66.** Y	**67.** N	**68.** Y
69. N	**70.** Y	**71.** Y	**72.** N
73. N	**74.** Y	**75.** Y	**76.** N
77. Y	**78.** Y	**79.** N	**80.** N
81. N	**82.** Y	**83.** Y	**84.** Y
85. Y	**86.** Y	**87.** Y	**88.** N
89. N	**90.** Y	**91.** Y	**92.** N
93. Y	**94.** N	**95.** Y	**96.** N
97. Y	**98.** Y	**99.** Y	**100.** D
101. Y	**102.** Y	**103.** D	**104.** C

Previous Years' GATE Questions

1. Consider the languages $L_1=\{\Phi\}$ and $L_2=\{a\}$. Which one of the following represents $L_1L_2^* \cup L_1^*$

(GATE 2013)

A. $\{\epsilon\}$ B. Φ C. a^* D. $\{\epsilon, a\}$

2. Which of the following statements is/are FALSE?

(GATE 2013)

1. For every non-deterministic Turing machine there exists an equivalent deterministic Turing machine.
2. Turing recognisable languages are closed under union and complementation.
3. Turing recognisable languages are closed under intersection and complementation.
4. Turing recognisable languages are closed under union and intersection.

A. 1 and 4 only B. 1 and 3 only

C. 3 only D. 4 only

3. Which of the following statements is/are TRUE?

(GATE 2013)

1. The problem of determining whether there exists a cycle in an undirected graph is in P.
2. The problem of determining whether there exists a cycle in an undirected graph is in NP.

3. If a problem A is NP-complete, there exists a non-deterministic polynomial time algorithm to solve A.

A. 1,2 and 3only B. 1 and 2 only
C. 2 and 3 only D. 1 and 3 only

4. Consider the following languages **(GATE 2013)**

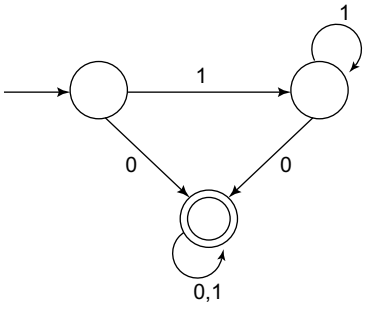
$L_1=\{0^p1^q0^r/p,q,r>=0\}$

$L_2=\{0^p1^q0^r/p,q,r>=0, p\neq r\}$

A. L_2 is context-free
B. $L_1 \bigcap L_2$ is context-free
C. Complement of L_2 is recursive
D. Complement of L_1 is context-free but not regular.

5. Consider the following DFA A given below:

(GATE 2013)



Which of the following are FALSE?

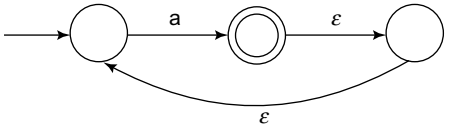
1. Complement of L(A) is context-free
2. L(A)=L((11*0+)(0+)*0*1*)
3. For the language accepted by A, A is the minimal DFA
4. A accepts all strings over {0,1} of length at least 2

A. 1 and 3 only B. 2 and 4 only
C. 2 and 3 only D. 3 and 4 only

6. What is the complement of the language accepted by the NFA shown below?

Assume Σ = {a} and ε is the empty string.

(GATE 2012)



A. ∅ B. {ε}
C. a* D. {a, ε}

As we want complement of the language, first and last states (nodes) becomes acceptable nodes. Thus, an empty string itself leads to final state.

7. Which of the following problems are decidable?

1. Does a given program ever produce an output?
2. If *L* is a context-free language, then, is also context-free?
3. If *L* is a regular language, then, is *L* also regular?
4. If *L* is a recursive language, then, is also recursive?

A. 1, 2, 3, 4 B. 1, 2
C. 2, 3, 4 D. 3, 4

8. Given the language *L* = {*ab, aa, baa*}, which of the following strings are in *L**? **(GATE 2012)**

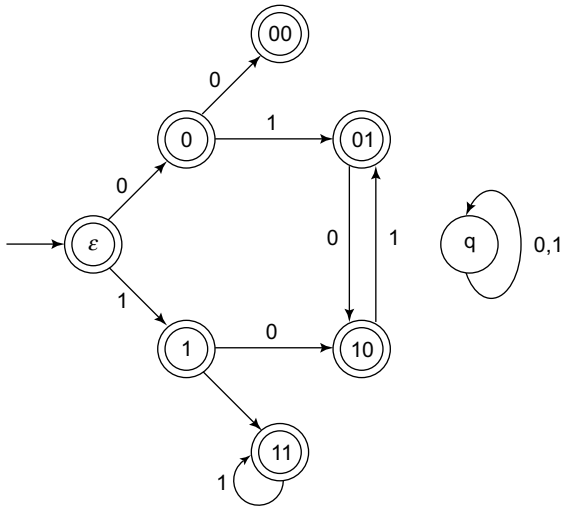
1. *abaabaaabaa* 2. *aaaabaaaa*
3. *baaaaabaaaab* 4. *baaaaabaa*

A. 1, 2 and 3 B. 2, 3 and 4
C. 1, 2 and 4 D. 1, 3 and 4

Answer: string baaaaabaaaab of third option cannot be generated by concatenating language elements.

9. Consider the set of strings on {0,1} in which, *every substring of 3 symbols* has at most *two* zeros. For example, 001110 and 011001 are in the language, but 100010 is not. All strings of length less than 3 are also in the language. A partially completed DFA that accepts this language is shown below.

(GATE 2012)



The missing arcs in the DFA are

A.

	00	01	10	11	q
00	1	0			
01					1
10	0				
11			0		

B.

	00	01	10	11	q
00		0			1
01		1			
10				0	
11		0			

C.

	00	01	10	11	q
00		1			0
01		1			
10			0		
11		0			

D.

	00	01	10	11	q
00		1			0
01				1	
10	0				
11			0		

For the grammar below, a partial LL(1) parsing table is also presented along with the grammar. Entries that need to be filled are indicated as E1, E2, and E3. ε is the empty string, $ indicates end of input, and, | separates alternate right hand sides of productions.

$S \rightarrow a\ A\ b\ B \mid b\ A\ a\ B \mid \varepsilon$

$A \rightarrow S$

$B \rightarrow S$

	a	b	$
S	E1	E2	$S \rightarrow \varepsilon$
A	$A \rightarrow S$	$A \rightarrow S$	error
B	$B \rightarrow S$	$B \rightarrow S$	E3

10. The FIRST and FOLLOW sets for the non-terminals A and B are **(GATE 2012)**

A. FIRST (A) = {a, b, ε} = FIRST (B)
FOLLOW (A) = {a, b}
FOLLOW (B) = {a, b, $}

B. FIRST (A) = {a, b, $}
FIRST (B) = {a, b, ε}
FOLLOW (A) = {a, b}
FOLLOW (B) = {$}

C. FIRST (A) = {a, b, ε} = FIRST (B)
FOLLOW (A) = {a, b}
FOLLOW (B) = ∅

D. FIRST (A) = {a, b} = FIRST (B)
FOLLOW (A) = {a, b}
FOLLOW (B) = {a, b}

11. The appropriate entries for E1, E2, and E3 are **(GATE 2012)**

A. E1 : $S \rightarrow aAbB, A \rightarrow S$
E2 : $S \rightarrow bAaB, B \rightarrow S$
E3 : $B \rightarrow S$

B. E1 : $S \rightarrow aAbB, S \rightarrow \varepsilon$
E2 : $S \rightarrow bAaB, S \rightarrow \varepsilon$
E3 : $S \rightarrow \varepsilon$

C. E1 : $S \rightarrow aAbB, S \rightarrow \varepsilon$
E2 : $S \rightarrow bAaB, B \rightarrow \varepsilon$
E3 : $B \rightarrow S$

D. E1 : $A \rightarrow S, S \rightarrow \varepsilon$
E2 : $B \rightarrow S, S \rightarrow \varepsilon$
E3 : $B \rightarrow S$

12. The lexical analysis for a modern computer language such as Java needs the power of which one of the following machine models in a necessary and sufficient sense? **(GATE 2011)**

A. Finite state automata

B. Deterministic pushdown automata

C. Non-Deterministic pushdown automata

D. Turing machine

13. Let P be a regular language and Q be a context free language such that Q subset of P.

(For example, let P be the language represented by the regular expression p^*q^* and Q be $\{ P^nQ^n/n \in N \}$). Then which of the following is ALWAYS regular?

(GATE 2011)

A. $P \cap Q$ B. $P - Q$ C. $\Sigma^* - P$ D. $\Sigma^* - Q$

Answer: $\Sigma^* - P$ is the complement of P so it is always regular, since regular languages are closed under complementation.

14. In a compiler, keywords of a language are recognised during **(GATE 2011)**

A. Parsing of the program

B. The code generation

C. The lexical analysis of the program

D. Dataflow analysis

15. Which of the following pairs have different expressive power? **(GATE 2011)**

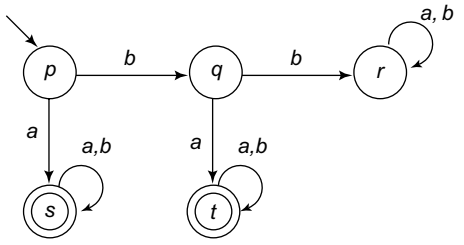
A. Deterministic finite automata (DFA) and Non-deterministic finite automata (NFA)

B. Deterministic push down automata (DPDA) and Non-deterministic push down automata (NPDA)

C. Deterministic single-tape Turing machine and Non-deterministic single tape Turing machine

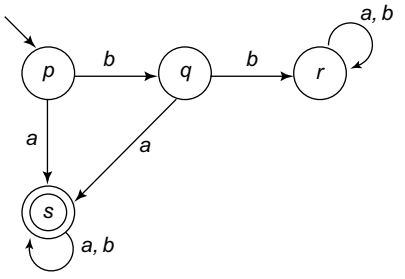
D. Single-tape Turing machine and multi-tape Turing machine

16. A deterministic finite automation (DFA)D with alphabet $\Sigma = \{a,b\}$ is given below **(GATE 2011)**

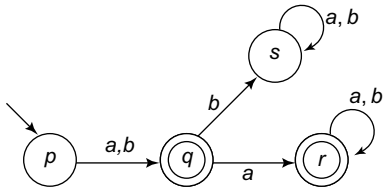


Which of the following finite state machines is a valid minimal DFA which accepts the same language as D?

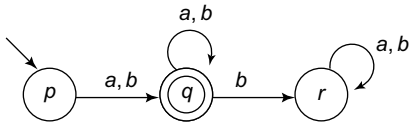
A.



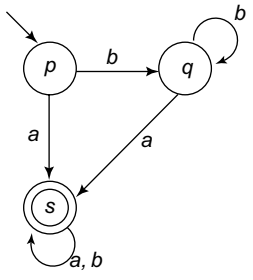
B.



C.



D.



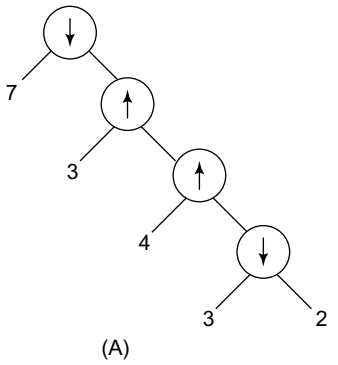
Options B and C will accept the string b

Option – D will accept the string "bba"

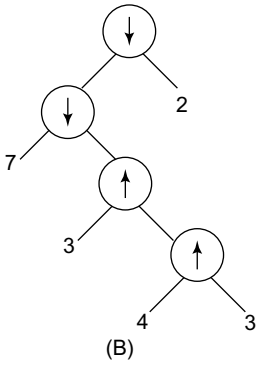
Both are invalid strings.

So the minimised DFA is option A

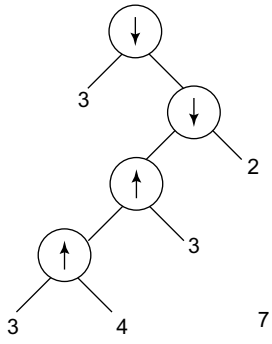
17. Consider two binary operators '↑' and '↓' with the precedence of operator ↓ being lower than that of the operator ↑. Operator ↑ is right associative while operator ↓, is left associative. Which one of the following represents the parse tree for expression (7 ↓ 3 ↑ 4 ↑ 3 ↓ 2)? **(GATE 2011)**



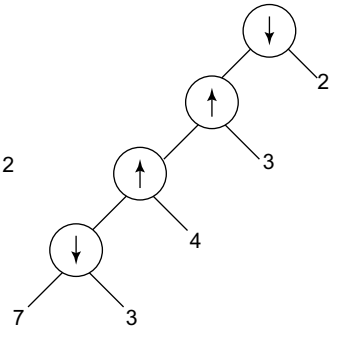
(A)



(B)



(C)



(D)

18. Consider the languages L1, L2 and L3 as given below

L1= { $0^p 1^q$ |p,q ϵ N}

L2= {$0^p 1^q$ |p,q N and p =q }and

L3 = {$0^p 1^q 0^r$ |p,q,r ϵN and p =q= r}

Which of the following statements is NOT TRUE?

(GATE 2011)

A. Push Down Automata (PDA) can be used to recognize L1 and L2

B. L1 is a regular language

C. All the three languages are context free

D. Turing machines can be used to recognize all the languages

L1: regular language

L2: context free language

L3: context sensitive language

19. Which data structure in a compiler is used for managing information about variables and their attributes?

(GATE 2010)

A. Abstract syntax tree

B. Symbol table

C. Semantic stack

D. Parse table

20. Which languages necessarily need heap allocation in the runtime environment? **(GATE 2010)**

A. Those that support recursion
B. Those that use dynamic scoping
C. Those that allow dynamic data structures
D. Those that use global variables

21. Let L1 be a recursive language. Let L2 and L3 be languages that are recursively enumerable but not recursive. Which of the following statements is not necessarily true? **(GATE 2012)**

A. L2 – L1 is recursively enumerable
B. L1 – L3 is recursively enumerable
C. L2 ∩ L1 is recursively enumerable
D. L2 ∪ L1 is recursively enumerable

22. The grammar S → aSa|bS|c is **(GATE 2010)**

A. LL(1) but not LR(1)
B. LR(1) but not LR(1)
C. Both LL(1) and LR(1)
D. Neither LL(1) nor LR(1)

23. Let L = {w ∈ (0 + 1)*|w has even number of 1s}, i.e. L is the set of all bit stringswith even number of 1s. Which one of the regular expressions below represents L? **(GATE 2010)**

A. (0 *10 *1) *
B. 0 * (10 *10 *) *
C. 0 * (10 *1*) * 0 *
D. 0 *1(10 *1) *10 *

24. Consider the languages L1 = $\{0^i1^j \mid i \neq j\}$. L2 = $\{0^i1^j \mid i = j\}$. L3 = $\{0^i1^j \mid i = 2j + 1\}$, L4 = $\{0^i1^j \mid i \neq 2j\}$. Which one of the following statements is true? **(GATE 2010)**

A. Only L2 is context free
B. Only L2 and L3 are context free
C. Only L1 and L2 are context free
D. All are context free

25. Let w be any string of length n in {0, 1}*. Let L be the set of all substrings of w. What is the minimum number of states in a non-deterministic finite automaton that accepts L? **(GATE 2010)**

A. n–1 B. n C. n+1 D. 2^n–1

26. S → aSa|bSb|a|b

The language generated by the above grammar over the alphabet{a,b} is the set of **(GATE 2009)**

A. All palindrome
B. All odd length palindromes
C. Strings that begin and end with same symbol
D. All even length palindromes

27. Which one of the following languages over the alphabet {0,1} is described by the regular expression: (0+1)*0(0+1)*0(0+1)*? **(GATE 2009)**

A. The set of all strings containing the substring 00
B. The set of all strings containing at most two 0's
C. The set of all string containing at least two 0's
D. The set of all strings that begin and end with either 0 or 1

28. Which one of the following is FALSE? **(GATE 2009)**

A. There is a unique minimal DFA for every regular language
B. Every NFA can be converted to an equivalent PDA
C. Complement of every context-free language is recursive
D. Every nondeterministic PDA can be converted to equivalent deterministic PDA

29. Let L = L1⋂L2, where L1 and L2 are languages as defined below: **(GATE 2009)**

L1={$a^mb^mca^nb^n$/m,n>=0}

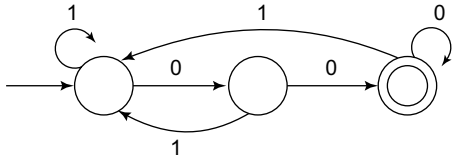
L2={$a^ib^jc^k$/i,j,k>=0}

Then L is

A. Not recursive B. Regular
C. Context-free but not regular
D. Recursively enumerable but not context free

This follows directly from the property of CFGs, where we see that CFGs are not closed under intersection, but produces CFGs when intersected with RLs.

30.



The above DFA accepts the set of strings over{0,1} that

A. Begins either with 0 or 1
B. End with 0
C. End with 00
D. Contain the substring 00

31. Which of the following statements are TRUE? **(GATE 2009)**

I. There exists parsing algorithms for some programming languages whose complexities are less that $\Theta(n^3)$
II. A programming language which allows recursion can be implemented with static storage allocation.
III. No L-attributed definition can be evaluated in the framework of bottom-up parsing
IV. Code improving transformations can be performed at both source language and intermediate code level.

A. I and II B. I and IV
C. III and IV D. I,III, and IV

32. Which of the following is true for the language $\{a^p\}$ p is a prime ? **(GATE 2008)**

A. It is not accepted by a Turing machine
B. It is regular but not context-free
C. It is context-free but not regular
D. It is neither regular nor context-free, but accepted by a Turing machine

33. Which of the following are decidable? **(GATE 2008)**

I. Whether the intersection of two regular languages is infinite
II. Whether a given context-free language is regular
III. Whether two push-down automata accept the same language
IV. Whether a given grammar is context-free

A. I and II B. I and IV
C. II and III D. II and IV

Explanation: I is true because intersection of two regular languages is always regular. So we can build DFA for intersection language and one can check for the finiteness in polynomial time(checking for loop .....). IV is also true because given a grammar whether it is context free or not can be easily identified by writing some simple string recognition program (context-free grammar (CFG) is a grammar in which every production rule is of the form $V \rightarrow w$ where V is a single nonterminal symbol, and w is a string of terminals and/or nonterminal's (possibly empty) can be easily identified.

34. Which of the following describes a handle (as applicable to LR-parsing) appropriately? **(GATE 2008)**

A. It is the position in a sentential form where the next shift or reduce operation will occur
B. It is non-terminal whose production will be used for reduction in the next step
C. It is a production that may be used for reduction in a future step along with a position in the sentential form where the next shift or reduce operation will occur
D. It is the production p that will be used for reduction in the next step along with a position in the sentential form where the right hand side of the production may be found

35. Some code optimisations are carried out on the intermediate code because **(GATE 2008)**

A. They enhance the portability of the compiler to other target processors
B. Program analysis is more accurate on intermediate code than on machine code
C. The information from dataflow analysis cannot otherwise be used for optimisation
D. The information from the front end cannot otherwise be used for optimisation

36. If L and $\overline{L}$ are recursively enumerable then L is **(GATE 2008)**

A. Regular B. Context-free
C. Context-sensitive D. Recursive

37. Which of the following statements is false? **(GATE 2008)**

A. Every NFA can be converted to an equivalent DFA
B. Every non-deterministic Turing machine can be converted to an equivalent deterministic Turing machine
C. Every regular language is also a context-free language
D. Every subset of a recursively enumerable set is recursive

38. Given below are two finite state automata (→ indicates the start state and F indicates a final state) **(GATE 2008)**

Y:

	a	b
→1	1	2
2(F)	2	1

Z:

	a	b
→1	2	2
2(F)	1	1

Which of the following represents the product automaton Z×Y?

A.

	a	b
→P	S	R
Q	R	S
R(F)	Q	R
S	Q	P

B.

	a	b
→P	S	Q
Q	R	S
R(F)	Q	P
S	P	Q

C.

	a	b
→P	Q	S
Q	R	S
R(F)	Q	P
S	Q	P

D.

	a	b
→P	S	Q
Q	S	R
R(F)	Q	P
S	Q	P

39. Which of the following statements are true? **(GATE 2008)**

I. Every left-recursive grammar can be converted to a right-recursive grammar and viceversa

II. All ε-productions can be removed from any context-free grammar by suitable transformations

III. The language generated by a context-free grammar all of whose productions are of the form X $\rightarrow$ w or X $\rightarrow$ wY (where, w is a string of terminals and Y is a non-terminal), is always regular

IV. The derivation trees of strings generated by a context-free grammar in Chomsky Normal Form are always binary trees

A. I, II, III and IV

B. II, III and IV only

C. I, III and IV only

D. I, II and IV only

40. Match the following **(GATE 2008)**

E.	Checking that identifiers are declared before their use	P.	$L = \{a^n b^m c^n d^m \mid n \geq 1, m \geq 1\}$
F.	Number of formal parameters in the declaration of a function agrees with the number of actual parameters in use of that function.	Q.	$X \rightarrow XbX \mid XcX \mid dXf \mid g$
G.	Arithmetic expressions with matched pairs of parentheses	R.	$L = \{wcw \mid w \in (a\|b)^*\}$
H.	Palindromes	S.	$X \rightarrow bXb \mid cXc \mid \varepsilon$

A. E – P, F – R, G – Q, H – S

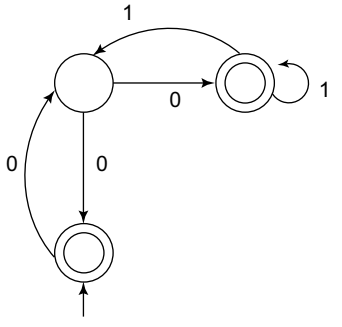
B. E – P, F – R, G – S, H – Q

C. E – R, F – P, G – Q, H – S

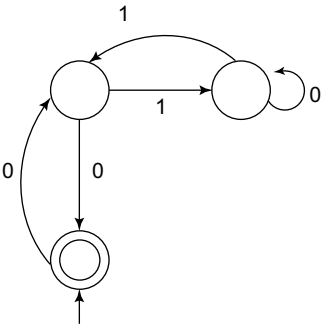
D. E – P, F – R, G – S, H – Q

41. Match the following NFAs with the regular expressions they correspond to **(GATE 2008)**

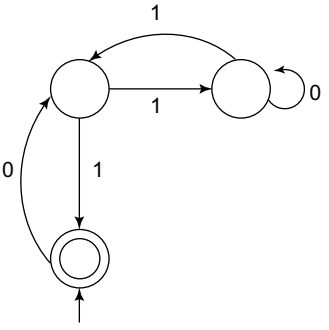
P.



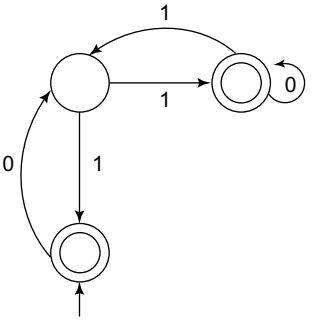
Q.



R.



S.



1. ε + 0 (01 *1+00)*01*
2. ε + 0 (10 *1+00)*0
3. ε + 0 (10 *1+10)*1
4. ε + 0 (10 *1+10)*10*

(A) P – 2, Q – 1, R – 3, S – 4

(B) P ↕–1, Q – 3, R – 2, S – 4

(C) P – 1, Q – 2, R – 3, S – 4

(D) P – 3, Q – 2, R – 1, S – 4

42. Which of the following are regular sets? **(GATE 2008)**

I. $\{ a^n b^{2m} \}\ n >= 0, m >= 0\}$

II. $\{ a^n b^m \}\ n = 2m\}$

III. $\{ a^n b^m \}\ n \neq m\}$

IV. {xcy /x, y, ϵ {a,b} *}

A. I and IV only B. I and III only

C. I only D. IV only

43. Which of the following are true? **(GATE 2008)**

I. A programming language which does not permit global variables of any kind and has no nesting of procedures/functions, but permits recursion can be implemented with static storage allocation

II. Multi-level access link (or display) arrangement is needed to arrange activation records only if the programming language being implemented has nesting of procedures/functions

III. Recursion in programming languages cannot be implemented with dynamic storage allocation

IV. Nesting procedures/functions and recursion require a dynamic heap allocation scheme and cannot be implemented with a stack-based allocation scheme for activation records

V. Programming languages which permit a function to return a function as its result cannot be implemented with a stack-based storage allocation scheme for activation records

A. II and V only B. I, III and IV only
C. I, II and V only D. II, III and V only

44. An LALR(1) parser for a grammar G can have shift-reduce (S-R) conflicts if and only if **(GATE 2008)**

A. The SLR(1) parser for G has S-R conflicts
B. The LR(1) parser for G has S-R conflicts
C. The LR(0) parser for G has S-R conflicts
D. The LALR(1) parser for G has reduce-reduce conflicts

45. Which of the following problems is undecidable? **(GATE 2007)**

A. Membership problem of CFLs
B. Ambiguity problem of CFGs
C. Finiteness problem for FSAs
D. Equivalence problem for FSAs

Answer: B. For the membership problem we have the CYK algorithm or any parsing algorithms of CFGs which uses backup. The finiteness problem of FSAs is to merely ensure that there is no cycle in the state diagram of the finite state machine. For any two regular sets L1 and L2 we have L1=L2 iff L1ΠL2' U L2ΠL1' is empty. Since the regular sets are closed under union and intersection the equivalence problem is decidable.

A standard undecidability result is the determination of the ambiguity of CFGs.

46. Which of the following is TRUE? **(GATE 2007)**

A. Every subset of a regular set is regular.
B. Every finite subset of a non-regular set is regular
C. The union of two non-regular sets is not regular
D. Infinite union of finite set is regular

Every finite set is trivially a regular set. Choice A cannot be correct as any formal language is a subset of Σ^* which is a regular set. Choice C cannot be correct as the union of two nonregular cfls, a cfl and its complement is necessarily regular e.g. take all the palindromes over some alphabet. A formal language can be looked upon as the infinite union of singleton sets consisting of one string in the language, so D cannot be correct.

47. Which of the following is a top-down parser? **(GATE 2007)**

A. Recursive descent parser
B. Operator precedence parser
C. An LR(k) parser
D. An LALR(k) parser

A recursive descent parser is nothing but an LL(1) parser.

48. A minimum state deterministic finite automaton accepting the language L={w|w ε{0,1}*}, number of 0s and 1s are divisible by 3 and 5, respectively has **(GATE 2007)**

A. 15 states B. 11 states
C. 10 states D. 9 states

49. The language L = { 0^i21^j |i,j>=0} over the alphabet {0, 1, 2} is **(GATE 2007)**

A. Not recursive
B. Is recursive and is a deterministic CFL
C. Is a regular language
D. Is not a deterministic CFL but a CFL

Answer : One can easily design a deterministic push down automata which scans the input left to right, when a 0 is encountered it stacks it, and moves to the right. When a 2 is encountered it switches from a stacking to a pop state and so long as a 1 comes in the input it pops a 0. When the input is exhausted and the stack is empty it goes to a final state.

50. Which of the following languages is regular? **(GATE 2007)**

A. {ww^R| w in $\{0,1\}^+$}
B. {wwRx| w,x in $\{0,1\}^+$|}
C. {wxwR| w,x in {0,1}+}
D. {xwwR| w in $\{0,1\}^+$}

Answer : This is merely the language where the strings are either 0w0, 1w1 where w is any string in {0,1}*.

The remaining are standard palindrome languages:

For (A) intersect with 0*110* and apply the pumping lemma.

For (B) intersect with 0*110*1 and apply the pumping lemma

For (D) intersect with 10*110* and apply the pumping lemma

51. Consider the grammar with the nonterminals N = {S, C, S1}, terminals T= {a, b, I, t, e}, with S as the start symbol, and the following set of rules:

$S \rightarrow iCtSS1 \mid a$

$S1 \rightarrow eS \mid \varepsilon$

$C \rightarrow b$

The grammar is not LL(1) because: **(GATE 2007)**

A. it is left recursive B. it is right recursive

C. it is ambiguous D. it is not context-free

Explanation: The grammar is not left recursive, so (A) is not correct. The grammar is very much a cfg, as only a single nonterminal appears on the lhs of every rule, so (D) is not correct. The grammar is right recursive, as S→+----S----, but this has nothing to do with the property of being LL(1),

52. Consider the following two statements:

(GATE 2007)

P: Every regular grammar is LL(1)

Q: Every regular set has a LR(1) grammar

Which of the following is TRUE?

A. Both P and Q are true

B. P is true and Q is false

C. P is false and Q is true

D. Both P and Q are false

Explanation: Consider the left recursive right linear grammar generating a+, S→Sa|a

This cannot be LL(1) so P is false. Every regular set is necessarily a dcfl and so a LR(1) grammar exists for it, so Q is true.

53. Consider the following grammar

$S \rightarrow S^*E$

$S \rightarrow E$

$E \rightarrow F+E$

$E \rightarrow F$

$F \rightarrow Id$

Consider the following LR(0) items corresponding to the grammar rule above

i. $S \rightarrow S^*.E$ ii. $E \rightarrow F.+E$

iii. $E \rightarrow F+.E$

Given the items above, which two of them will appear in the same set in the canonical sets-of-items for the grammer?

A. (i) and (ii) B. (ii) and (iii)

C. (i) and (iii) D. None of the above

54. Let $L1=\{0^{n+m}1^n0^m/n,m>=0\}$, $L2=\{0^{n+m}1^{n+m}0^m/n,m>=0\}$, and $L3=\{0^{n+m}1^{n+m}0n+^m/n,m>=0\}$. Which of these languages are not context free?

A. L1 only B. L3 only

C. L1 and L2 D. L2 and L3

55. If s is a string over (0+1)* then let n0(s) denote number of0'sin s and n1(s) the number of 1s in s. Which one of the following languages is not regular>?

A. L={s ϵ (0+1)*/n0(s) is a 3-digit prime}

B. L={s ϵ (0+1)*/for every prefix S' of s/no(s')-n1(s') ⇐ 2}

C. L = {s ϵ (0 + 1)*/|n0(s)–n1(s)| ⇐ 4}

D. L={s ϵ (0+1)*/no(s) mod 7=n1(s) mod 5 =0}

56. For S ϵ(0+1)* let d(s) denote decimal value of s (e.g. d(101)=5). Let L={s ϵ(0+1)*/d(s)mod 5=2 and d(s) mod 7 ≠4}.

Which one of the following statements is true?

A. L is recursively enumerable, but not recursive.

B. L is recursive, but not context-free

C. L is context-free, but not regular

D. L is regular

57. Consider the following statement about the context free grammar.

G={S→SS, S→ab, S→ba, S→ϵ}

I. G is ambiguous

II. G produces all strings with equal number of a's and b's.

III. G can be accepted by a deterministic PDA

Which combination below expresses all the true statements.

A. I only B. I and III only

C. II and III only D. I, II and III

58. Let L1 be a regular language, L2 be a deterministic context free language and L3 a recursively enumerable, but not recursive, language. Which one of the following statements is false?

a. L1 ⋂ L2 is a deterministic CFL

b. L3 ⋂ L1is recursive

c. L1 ⋃ L2is context free

d. L1 ⋂ L2 ⋂ L3 is recursively enumerable

59. Consider the regular language L=(111+11111)*. The minimum number of states in any DFA accepting this language.

A. 3 B. 5 C. 8 D. 9

60. Consider the following grammar:

S→FR

R→*S|ϵ

F→id

In the predictive parser table, M, of the grammar the entries M[S,id] and M[R,$] respectively.

A. {S→FR} and {R→ϵ} B. {S→FR} and {}

C. {S→FR} and {R→*S} D. {F→id} and {R→ϵ}

61. Consider the following translation scheme

S→ER

R→*E{print('*');}|ϵ

E→F+E{print('+');}|F

F→(S)|id{print(id,value);}

Here id is a token that represents an integer and id.value represents the corresponding integer value. For an input '2+3*4', this translation

A. 2*3+4 B. 2*+34|

C. 23*4+ D. 234+*

62. Which one of the following grammars generates the language $L=\{a^ib^j/i \neq j\}$?

A. $S \rightarrow AC|CB$
$C \rightarrow aCb|a|b$
$A \rightarrow a\ A| \in$
$B \rightarrow B\ b| \in$

B. $S \rightarrow aS|Sb|a|b$

C. $S \rightarrow AC|CB$
$C \rightarrow aCb| \in$
$A \rightarrow a\ A| \in$
$B \rightarrow B\ b| \in$

D. $S \rightarrow AC|CB$
$C \rightarrow aCb| \in$
$A \rightarrow a\ A|\ a$
$B \rightarrow B\ b|\ b$

63. In the correct grammar above, what is the length of the derivation (number of steps starring from S) to generate the string a^lb^m with $l \neq m$.

A. Max(l,m)+2 B. L+m+2

C. L+m+3 D. Max(l,m)+3

ANSWER KEY

1. A	**2.** A	**3.** A	**4.** D
5. D	**6.** B	**7.** D	**8.** C
9. D	**10.** A	**11.** C	**12.** A
13. C	**14.** C	**15.** B	**16.** A
17. B	**18.** C	**19.** B	**20.** C
21. B	**22.** C	**23.** B	**24.** D
25. C	**26.** B	**27.** B	**28.** D
29. C	**30.** C	**31.** B	**32.** D
33. B	**34.** D	**35.** A	**36.** D
37. D	**38.** A	**39.** A	**40.** C
41. C	**42.** A	**43.** D	**44.** B
45. B	**46.** B	**47.** A	**48.** A
49. B	**50.** C	**51.** C	**52.** C
53. C	**54.** D	**55.** B	**56.** B
57. D	**58.** D	**59.** D	**60.** A
61. D	**62.** D	**63.** A	

CHAPTER 4 FOUR

Operating Systems

4.1 Process and Threads

A program under execution is called as a process. Normally, a Process Control Block (PCB) is created when a process is initiated which contains process identification number, starting address of page table, security related information, open file table entries per process, details regarding CPU time consumption, etc.

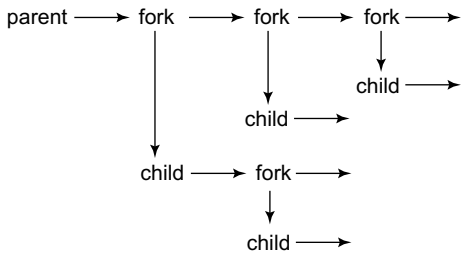
In today's virtual memory-based operating systems the following things may take place in order when we run a program.

1. User clicks an application or enters its name at the command prompt.
2. A special process structure with text, data, stack, heap, etc. is created in swap partition. It is called as virtual address space.
3. A PCB is created and added to the ReadyQueue.
4. Then scheduler decides when the process to be executed.

■ **Example** In Unix/Linux, we can use fork() to create a new processes. The fork() system call returns child's PID to parent process and 0 to child process. In a lay man's term, we can say that the statements after fork() will be executed both in the child and parent processes. Thus, if we happened to have a series of fork() calls (say n class) one after another we will have total 2^n processes to be generated by the program. They all can be said as process tree as child, parent, grandparent relationship we may see among them.

The following program explains about the process tree. Here, fork() is called 3 times thus 2^3 processes will be created in total including main process. Thus, in total we may see 8 times Hello message to be printed on the screen.

```
#include<sdtio.h>
#include<unistd.h>
int main(){
fork();
fork();
fork();
printf("Hello\n");
return 0;
}
```



■ **Example** The following program creates a process chain. Also check the process ID's of the processes created by running the following program. The fork() system call return zero to the child process and PID of the child to the parent process. Thus, we are calling another fork() if the returned value is zero, which will be true only in child process. Thus, first a child will be created and for that another child will be created and so on. This is called as **process chain**.

```
#include<unistd.h>
#include<sdtio.h>
int main(){
  if (fork() == 0){
    printf("Pid: %d\tpid: %d\n",getpid(),getppid());
        if(fork()==0){
      printf("Pid: %d\tpid: %d\n",getpid(),getppid());
         if (fork()==0)
      printf("Pid: %d\tpid: %d\n",getpid(),getppid());
      }
  }
  return 0;
}
```

The above program creates in total four processes such that every parent has one child exactly.

4.1.1 Thread

A ***thread*** is a sequential execution stream, and it is also the smallest scheduling unit of concurrency to run on a processor. The beauty of the thread is that each thread can be programmed as if it owns the entire CPU (e.g., you can use an infinite loop within a thread without halting the entire system). In other words, a thread contains the states of its own program counter, register values, and execution stacks. Therefore, threads provide the illusion of having an infinite number of CPUs, even on a single-processor machine.

Threads simplify programming significantly, and Microsoft Word is an example. As you are typing in Word, there is a thread dedicated for checking grammar, a thread for checking spelling, a thread for reformatting the text, and many other threads for various purposes. Since the thread for grammar checking can be programmed independently from the thread for spelling check, the difficulty for programming a large application like Word is greatly simplified.

How the Main Thread is Special?

However, for compatibility reasons, the **main()** routine has to retain some characteristics it had prior to the advent of POSIX threads. In particular:

1. We do not need to create explicitly the main thread with **pthread_create().** This thread is created on our behalf at process start-up by the system libraries.
2. The prototype for the **main()** function is **int main(void)** or **int main(int,char* argv[]).** This differs from the standard prototype of a thread's start routine: **void* start_routine(void*).**
3. If we return from **main(),** the *entire process terminates*. That is, all threads vanish, the process ceases to exist and the process exit status can be retrieved by the parent process.

Threads and Dispatching Loop

Inside each thread, there is a ***thread control block***. The thread control block maintains the execution states of the thread, the status of the thread (e.g., running or sleeping), and scheduling information of the thread (e.g., priority).

Threads are run from a ***dispatching loop***.

```
LOOP
Run thread
    Save states (into the thread control block)
    Choose a new thread to run
    Load states from a different thread (from the thread control block)
```

To run a thread, just load its states (registers, program counter, stack pointer) into the CPU, and do a jump. The process of saving the states of one thread and restoring states of another thread is often called a ***context switch***. The decision of which thread to run next involves ***scheduling***.

How Does the Dispatcher Regain Control?

The dispatcher gets control back from the running thread in two ways:

1. ***Internal events*** (sleeping beauty—go to sleep and hope Prince Charming will wake you):
   (a) A thread is waiting for I/O.
   (b) A thread is waiting for some other thread.
   (c) Yield—a thread gives up CPU voluntarily.
2. ***External events***:
   (a) Interrupts—a completed disk request wakes up the dispatcher, so the dispatcher can choose another thread to run).
   (b) Timer—it's like an alarm clock.

What States Should a Thread Save?

A thread should save anything that the next thread may trash before a context switch: program counter, registers, changes in execution stack. Each thread should be treated as an independent stream of execution.

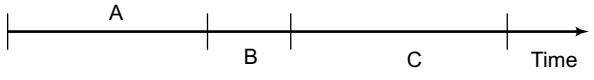
A context switch can also occur during an interrupt. During an interrupt, hardware causes the CPU to stop what it's doing, and to run the interrupt handler. The handler saves the states of the interrupted thread, runs the handler code, and restores the states of the interrupted thread.

How Does the Dispatcher Choose the Next Thread?

The dispatcher keeps a list of threads that are ready to run.
If no threads are ready to run—the dispatcher just loops.
If one thread is ready to run— life is easy.

If more than one thread are ready to run, we can choose the next thread according to different scheduling policies. Some examples are FIFO (first in, first out), LIFO (last in, first out), and priority-based policies.

The dispatcher also has the control of how to share the CPU among multiple threads. Suppose that we have thread A, B, and C. At one extreme, a dispatcher can run one thread to completion before running the other.



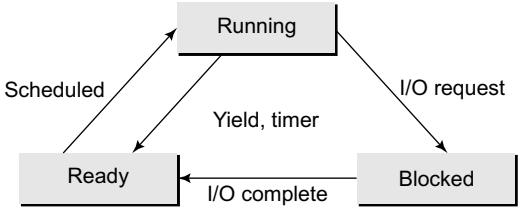
Alternatively, a dispatcher can use the timer to timeshare the CPU among three threads.

Per-Thread States

Each thread can be in one of three states:

1. Running—has the CPU
2. Blocked—waiting for I/O or another thread

3. Ready to run—on the ready list, waiting for the CPU



4.1.2 Process and Threads

A Thread is said to be light weight process as the overhead involved is less when a context switching takes place between two threads.

It is practically observed that context switching overhead is less for thread based systems then process based systems.

We cannot have in today's operating system, a program running without creating a process.

The virtual address space for a program is created only when a process is created for that program. On systems which are said to be having single process and single thread, we will not be able to find any real difference between the thread and that process. Only if we have multiple threads under a single process, may we find conceivable difference in program running time compared to an equivalent program which contains a main process and a set of child processes.

Thread based solutions are preferable if the available independent tasks are less computationally intensive otherwise process based concurrency is preferable.

■ **Example** The following program calls fork() after creating a thread in the main. As fork is called after fork, only thread of main process will be running. Thus, we get Hello message **once**.

```
#include<stdio.h>
#include<pthread.h>
void * HI (void *x) {
printf("Hello \n");
}
int main () {
pthread_t tid;
pthread_create (&tid, NULL, HI, NULL);
fork();
pthread_join (tid, NULL);
     return 0;
}
```

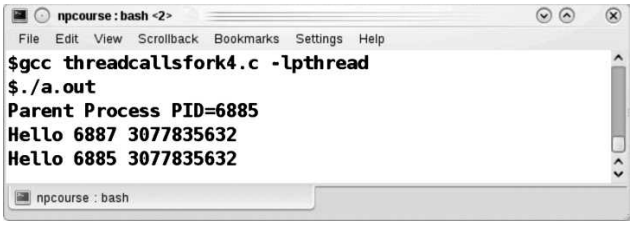
■ **Example** In the following program, we are calling pthread_create() after fork system call. Thus, it will be called in both the processes. That is, new thread will be created in both the processes. Thus, we get the message Hello two times.

```
#include<stdio.h>
#include<pthread.h>
void * HI (void *x) {
printf("Hello \n");
}
int main () {
pthread_t tid;
fork();
pthread_create (&tid, NULL, HI, NULL);
pthread_join (tid, NULL);
return 0;
}
```

■ **Example** We wanted to find out whether we can fork from a thread function or not. We know already that a thread can call another thread. Here, we are trying to see whether a thread can call fork or not. If so, what happens?

```
#include<stdio.h>
#include<pthread.h>
void * HI (void *x) {
fork();
printf("Hello %d %u\n", getpid (), pthread_ self());
}
int main () {
pthread_t tid;
printf("Parent Process PID=%d/n", getpid ());
pthread_create (&tid, NULL, HI, NULL);
pthread_join (tid, NULL);
return 0;
}
```

Snap shot of the above program is given as:



In our program, we have printed the process PID before creating a thread. Inside the thread function, we have called fork system call. After that, we have executed a printf function which prints a message Hello, PID of the process and TID of the thread. We have got these details two times which suggests that when a fork call is made, a new process is created and in that new thread is also created. This can be corroborated from the PID values. However, if we see TID values we may find both are same. Why?

We have fork handlers also like exit and signal handlers. For this purpose, we have to use pthread_atfork() method whose syntax is given below.

int pthread_atfork(void (**prepare***) (void), void (****parent***) (void),**
void (**child***) (void));**

The **pthread_atfork()** function declares fork handlers to be called prior to and following fork, within the thread that called **fork()**. The order of calls to **pthread_atfork()** is significant.

Before **fork()** processing begins, the *prepare* fork handler is called. The *prepare* handler is not called if its address is *NULL*.

The *parent* fork handler is called after **fork()** processing finishes in the parent process, and the *child* fork handler is called after **fork()** processing finishes in the child process. If the address of *parent* or *child* is *NULL*, then its handler is not called.

The *prepare* fork handler is called in LIFO (last-in first-out) order, whereas the *parent* and *child* fork handlers are called in FIFO (first-in first-out) order. This calling order allows applications to preserve locking order.

■ **Example** The following program demonstrates the use of pthread_atfork() function. We have created a thread by calling pthread_create() before fork. For pthread_fork() functions we have supplied thread functions. As thread is created first, its function HI will be executed first. Then, when fork is called the functions xyz, pqr and ijk are executed.

```
#include<stdio.h>
#include<pthread.h>
void  xyz (void) {
printf("Hi\n");
```

```
}
void pqr (void) {
printf ("Hey\n");
}
void ijk (void) {
printf ("Hei\n");
}
void * HI (void * x) {
printf ("Hello\n");
}
int main () {
pthread_t tid;
pthread_create (&tid, NULL, HI, NULL);
pthread_atfork (xyz, pqr, ijk);
fork();
pthread_join (tid, NULL);
return 0;
}
```

Snapshot of the above program is given as:

■ **Example** The following program demonstrates how to use pthread_at fork function. As discussed above, functions xyz, pqr and ijk are executed. For the sake of understanding, we have printed PID of the process also. We may find that xyz and pqr are executed in the main process while ijk is executed in child process.

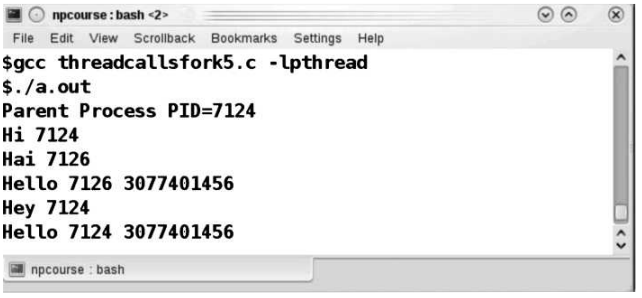
```
#include<stdio.h>
#include<pthread.h>
void  xyz (void) {
printf("Hi%d\n", getpid ());
}
void pqr (void) {
printf ("Hey %d\n", getpid ());
}
void ijk (void) {
printf ("Hai %d\n", getpid ());
}
int main () {
pthread_atfork (xyz, pqr, ijk);
fork();
return 0;
}
```

■ **Example:** The following program is to demonstrate what happens to fork handlers if the threads are created after fork.

We find that xyz function is getting executed only once. That is also in main process. It is expected that this function has to be called before fork. The function pqr is getting executed in parent process. While ijk is getting executed in child process. Also, these functions have got executed in their respective process before a child thread is created. Threads are executing their thread function, however in their respective parents which we can get support from PID values.

```
#include<stdio.h>
#include<pthread.h>
void xyz (void) {
printf("Hi%d\n", getpid ());
}
void pqr (void) {
printf ("Hey %d\n", getpid ());
}
void ijk (void) {
printf ("Hai %d\n", getpid ());
}
void * HI (void * x) {
pthread_atfork(xyz, pqr, ijk);
fork();
printf("Hello %d %u\n", getpid (), pthread_self());
}
int main () {
pthread_t tid;
printf("Parent Process PID=%d/n", getpid ());
pthread_create (&tid, NULL, HI, NULL);
pthread_join (tid, NULL);
return 0;
}
```

Snapshot of the above program is given as:



4.1.3 Why Concurrency?

On a single-processor machine, the operating system's support for ***concurrency*** allows multiple applications to share resources in such a way that applications appear to run at the same time. Since a typical application does not consume all resources at a given time, a careful coordination can make each application run as if it owns the entire machine.

There are a number of benefits for an operating system to provide concurrency:

1. Of course, the most obvious benefit is to ***be able to run multiple applications at the same time***.
2. Since resources that are unused by one application can be used for other applications, concurrency allows ***better resource utilisation***.
3. Without concurrency, each application has to be run to completion before the next one can be run. Therefore, concurrency allows a ***better average response time*** of individual applications.
4. Concurrency does not merely timeshare the computer; concurrency can actually achieve ***better performance***. For example, if one application uses only the processor, while another application uses only the disk drive, the time to run both applications concurrently to completion will be shorter than the time to run each application consecutively.

Concurrency also introduces certain drawbacks:

1. Multiple applications need to be protected from one another.
2. Multiple applications may need to coordinate through additional mechanisms.
3. Switching among applications requires additional performance overheads and complexities in operating systems (e.g., deciding which application to run next.)
4. In extreme cases of running too many applications concurrently will lead to severely degraded performance.

Overall, when properly used, concurrency offers more benefits than drawbacks.

4.2 Inter-process Communication Concurrency, Synchronisation

Independent threads have the following characteristics:

1. No states shared with other threads
2. Deterministic computation (output depends on input)
3. Reproducible (output does not depend on the order and timing of other threads)
4. Scheduling order doesn't matter

Cooperating threads have the following characteristics:

1. Shared states
2. Nondeterministic
3. Non-reproducible

For example, if you have two threads sharing the same display, they are likely to produce unpredictable results.

Thread A	Thread B
printf("ABC");	printf("123");

You may get "A12BC3" also as an outcome.

4.2.1 Why do we Need Cooperating Threads?

With additional mechanisms for coordination, threads can provide three major benefits:

1. Shared resources: multiple threads can share a single processor.
2. Speedup: an I/O-intensive thread can overlap the computation with a CPU-intensive thread.
3. Modularity: a word processor program can be decomposed into threads for particular functions, such as spelling check and grammar check.

4.2.2 Atomic Operations

An ***atomic operation*** always runs to completion, or it does not happen at all. The operation is indivisible.

On most machines, memory references and assignment (load and store) of words are atomic.

Many instructions are not atomic. For example, on most 32-bit architectures, double precision floating point store is not atomic; it involves two separate memory operations.

Race conditions are situations where two or more threads are reading or writing some shared data and the final result depends on who runs precisely when.

The idea of ***synchronisation*** is to use atomic operations to ensure cooperation between threads.

Mutual exclusion is a way to ensure one thread does a particular thing at a time, excluding the other threads.

Critical section is a piece of code that only one thread can execute at a time. Only one thread at a time will get into the section of code.

The use of a ***lock*** prevents someone from doing something. Therefore, a thread should lock before entering critical section, before accessing shared data. A thread should unlock when leaving the critical section, when done accessing shared data. A thread should wait if the critical section is locked. The key idea of synchronisation involves waiting.

4.2.3 The test_and_set Operation

Unlike disabling interrupts, `test_and_set` works on both uniprocessors and multiprocessors. The `test_and_set` operation atomically reads a memory location, sets it to 1, and returns the old value of the memory location.

```
value = 0;
Lock::Acquire() {
        while (test_and_set(value) == 1);
}
Lock::Release() {
        value = 0;
}
```

4.2.4 Busy-Waiting

The problem with both of the existing interrupt disable and test_and_set solutions is ***busy-waiting***, or consumption of CPU cycles while a thread is waiting for a lock. Busy- waiting is very inefficient.

Busy-waiting can be avoided with a waiting queue.

■ **Example**

Flag = 0	
Process P1	Process P2
while true	while true
do	do
-	-
-	-
while(flag) ;	while (flag) ;
flag=1	flag =1
CS	CS
Flag = 0	flag = 0
-	-
-	-
done	done

The main drawbacks of the above code for 2 process synchronisation or mutual exclusion are:

(i) It does not guarantee mutual exclusion under special circumstances

(ii) There is a danger of indefinite postponement of a process

(iii) It is strictly alternative type

(iv) Extending this to multi processes multiple critical sections is a cumbersome process.

4.2.5 Spin Locks

In the above example, a process which wants to enter into the critical section checks for a shared variable and waits on that variable (busy wait) if some condition is not satisfied otherwise it will enter into CS first changing this variable value such that no other process can enter into the critical section. This type of things are known as spin locks.

Sleep – wait things are called as suspend locks. Here, a lock is tied to an interrupt (signal)

■ **Example** Generalised Spin Lock with n possible values.

Shared var x : (1,2,3…n)
X=1

Process P1	Process P2	……..	Process Pn
Repeat – – until x = =1 CS X= X + 1	. . .		Repeat Until X = = n CS X =1

The main drawback of the above technique is

(i) longer waiting times

(ii) process must wait even if there are no conflicting requests at the same time

When critical section code is started, it should continue without interruption i.e, the code should be atomic (indivisible)
Requirements for Critical Section Solution:

(i) at most one process should be in the C.S

(ii) There should not be any deadlock

(iii) No preemption

(iv) Eventual entry i.e, a process attempting to enter into C.S should eventually succeed.

Granularity of the C.S
If it is coarse then it limits the parallelism, if it is fine code complexity, software overhead increases.
Conditional critical section is a syntactically delimited code in which code is permitted to access a protected variable. A conditional critical section also specifies a Boolean condition which must be true before the control enters into code region.
Semaphores are a type of generalised lock, first defined by Dijkstra in the late 60s. Semaphores are the main synchronisation primitive used in UNIX. Semaphores have a positive integer value, and support the following two operations:

- `P()`: an atomic operation that waits for semaphore to become positive, then decrements it by 1.
- `V()`: an atomic operation that increments semaphore by 1, waking up a thread waiting at `P()`, if any.

The P operation was an abbreviation for the Dutch word ***proberen***, meaning "to test," and the V operation was an abbreviation for ***verhogen***, meaning "to increment." Semaphores are like integers, except:

1. No negative values.
2. Only operations are `P()` and `V()`—can't read or write value, except to set it initially.
3. Operations must be atomic—two `P()` calls that occur together can not decrement the value below zero. Similarly, a thread that is going sleep in `P()` will not miss wakeup from `V()`, even if they both happen at about the same time.

A ***binary semaphore*** has a Boolean value, instead of an integer value. `P()` waits until the value is 1, then sets it to 0. `V()` sets the value to 1, waking up a thread waiting at `P()`, if any.

Two Uses of Semaphores

Mutual Exclusion

When semaphores are used for mutual exclusion, the semaphore has an initial value of 1, and `P()` is called before the critical section, and `V()` is called after the critical section.

```
semaphore s = 1;
P(s);
// critical section
V(s);
```

Scheduling Constraints

Semaphores can also be used to express generalised scheduling constraints. In other words, semaphores provide a way for a thread to wait for something. In this case, the initial value of the semaphore is usually 0 (but not always).

```
semaphore s1 = 0;
semaphore s2 = 0;
A() {                    B() {
        write(x);                P(s1);
        V(s1);                   read(x);
        P(s2);                   write(y);
        read(y);                 V(s2);
  }                        }
```

Producer-Consumer with a Bounded Buffer

One classic problem is the producer-consumer problem. A producer put things into a shared buffer; a consumer takes them out. Since it is inefficient to operate the producer and consumer in lockstep, a fixed-size buffer is used between them. The producer needs to wait if the buffer is full; the consumer needs to wait if the buffer is empty. The solution involves both scheduling and mutual exclusion.

There are three constraints for the solution:

(1) The consumer must wait if buffers are empty (scheduling constraint).

(2) The producer must wait if buffers are full (scheduling constraint).

(3) Only one thread can manipulate the buffer at a time (mutual exclusion).

For each constraint, we need a semaphore.

```
semaphore nLoadedBuffers = 0; // consumer waits on 0
semaphore nFreeBuffers = N;   // producer waits on 0
semaphore mutex = 1;          // one thread waits when
                              // another thread is
                              // modifying the
                              // buffer
```

```
Producer() {
        P(nFreeBuffers);
        P(mutex);
        // put 1 item in the buffer
        V(mutex);
        V(nLoadedBuffers);
  }
```

```
Consumer() {
        P(nLoadedBuffers);
        P(mutex);
        // take 1 item from the buffer
        V(mutex);
        V(nFreeBuffers);
  }
```

4.2.6 Monitors

Semaphore based concurrency control design makes code development difficult. The idea of monitors is to separate these two concerns: use locks for mutual exclusion and condition variables for scheduling constraints.

A ***monitor*** is a lock with zero or more conditional variables for managing concurrent access to shared data. The ***lock*** provides mutual exclusion to the shared data. A ***conditional variable*** allows a queue of waiting threads while being inside a critical section.

Lock

A lock provides two operations:

```
Lock::Acquire()
        // wait until the lock is free, then grab it
Lock::Release()
        // unlock, wake up anyone waiting in Acquire
```

As a general rule of thumb, always acquire a lock before accessing a shared data structure; always release a lock after finishing with the shared data structure. A lock is initially free. The following is a simple example of using a lock on a synchronised list:

```
AddToQueue() {
        Lock.Acquire();
        // put 1 item to the queue
        Lock.Release();
}

RemoveFromQueue() {
        Lock.Acquire();
        // if something on the queue
        //   remove 1 item from the queue
        Lock.Release();
        return item;
}
```

Condition Variables

Although the example queue is synchronised, if we want to perform waiting inside locked regions, we need additional mechanisms. For example, a process may want to wait for something to be added to the queue before the removal operation.

However, holding the lock while waiting prevents other processes from entering the locked regions. Condition variables make it possible to go to sleep inside a critical section by atomically releasing the lock and going to sleep.

Associated with each condition variable is a waiting queue of threads inside a critical section, and condition variables provide the following operations:

```
Wait(); // atomically release the lock and go to
    sleep; re-acquire lock on return
Signal(); // wake up a waiter, if any
Broadcast(); // wake up all waiters
```

Note

These operations should always occur inside locked regions. The following is a synchronised queue with waiting inside the remove operation.

```
AddToQueue() {
        lock.Acquire();
```

```
            // put 1 item to the queue
            condition.Signal(&lock);
            lock.Release();
    }
      RemoveFromQueue() {
            lock.Acquire();
            while nothing on queue
                  condition.Wait(&lock);
            // remove 1 item from the queue
            lock.Release();
            return item;
      }
```

Readers-Writers Problem

The readers-writers problem is commonly seen in database applications, where users are separated into readers who never modify the database, and writers who read and modify the database. Although we want only one writer at the same, using a single lock on the database would be overly restrictive, since many readers can read at the same time.

Constraints

1. A reader should wait when a writer is accessing or waiting for the database (`Condition okToRead`). (The writer has a higher priority over reader, since it is more difficult for a writer to achieve exclusive access of the database.)
2. A writer should wait when there is a reader or a writer accessing the database (`Condition okToWrite`).
3. A reader or a writer should wait when someone is modifying global states (Lock `lock`).

Basic Structure of the Solution

The following is the pseudocode for the readers-writers problem:

```
Reader
        // wait until no writers
        // access database
        // wake up waiting writers
Writer
        // wait until no readers or writers
        // access database
        // wake up waiting readers or writers
```

Code

```
// Global States
AR = 0; // number of active readers
AW = 0; // number of active writers
WR = 0; // number of waiting readers
WW = 0; // number of writing writers

Condition okToRead = NULL;
Condition okToWrite = NULL;
Lock lock = FREE;
```

```
Reader() {
    lock.Acquire();
    // manipulate global states
    while ((AW + WW) > 0) {
       WR++;
    okToRead.Wait(&lock);
       WR--;
    }
    AR++;
    lock.Release();
    // access database
    lock.Acquire();
    // manipulate global states
    AR--;
    if (AR == 0 && WW > 0) {
       okToWrite.Signal(&lock);
    }
    lock.Release();
  }
```

```
Writer() {
    lock.Acquire();
// manipulate global states
while ((AW + AR) > 0) {
       WW++;
okToWrite->Wait(&lock);
       WW--;
    }
    AW++;
    lock.Release();
    // access database
    lock.Acquire();
// manipulate global states
    AW--;
    if (WW > 0) {
okToWrite->Signal(&lock);
    } else if (WR > 0) {
okToRead->Broadcast(&lock);
    }
    lock.Release();
  }
```

Note that if writers keep on arriving, readers may starve. Also, although all readers are awaken by the `Broadcast()`, an arriving writer may put some readers back to the wait queue in the `while` loop before these readers get a chance to access the database.

Comparison Between Semaphores and Monitors

Although both semaphores and monitors provide atomic operations and queueing, they have very different behaviors. For example, the following implementation has a very different behavior.

```
Wait() { semaphore→P(); }
Signal() { semaphore→V(); }
```

First, condition variables only work inside a lock. Using semaphores inside a lock may cause a deadlock. Second, condition variables have no history, but semaphores have. If a thread calls `signal()`, and the waiting queue is empty, the `signal()` call does nothing. If a thread later arrives and calls `wait()`, the thread will wait. However, if a thread calls `V()`, and no one is waiting, the semaphore increments. If a thread later calls `P()`, the semaphore decrements, and the thread continues.

The next example takes the lock into account. However, the behavior is still different.

```
Wait(Lock *lock) {
        lock→ Acquire(); // [1]
        semaphore → P(); // [3]
        lock → Release();
}

Signal() {
        if (semaphore queue is not empty)     // [2]
            semaphore→V();                    // [4]
}
```

Third, semaphores cannot look at the contents of semaphore queue. Fourth, releasing the lock and going to sleep need to occur atomically. Should the semaphore queue be checked `[2]` right after releasing the lock `[1]` and before calling `P() [3]`, the corresponding `V() [4]` will never be called, and the thread sleeping in `P()` will never wake up.

4.3 Deadlock

Deadlock can be considered as a situation in which there will not be any effective work in the system. Roughly speaking, a ***deadlock*** occurs when threads are waiting for resources (e.g., devices and files) with circular dependencies. Deadlocks generally involve ***nonpreemptable resources*** (e.g., storage allocated to a file, a lock for mutual exclusion), which cannot be taken away from its current thread without causing the computation to fail. Deadlocks that involve ***preemptable resources*** (e.g., CPU) can usually be resolved by reallocating resources from one thread to another. Recall that ***starvation*** is a condition where a thread waits indefinitely. A deadlock implies starvation.

A deadlock can happen with any kind of resource. Deadlocks can also occur with multiple resources, and you cannot solve the deadlock for each resource independently. For example, one thread can grab all the memory; another grabs all the disk space, and yet another grabs all the tape storage. Each thread may need to wait for other to release. Round-robin CPU scheduling cannot prevent deadlocks (or starvation) from happening.

Deadlock can occur whenever there is waiting.

4.3.1 Conditions for Deadlocks

Fortunately, someone has identified four necessary (but not sufficient) conditions for a deadlock to occur. Without all of these four conditions, a deadlock cannot occur:

1. Limited access (lock-protected resources)
2. No preemption (if someone has the resource, it can't be taken away)
3. Wait while holding (holding a resource while requesting and waiting for the next resource).
4. Circular chain of requests

Deadlock Prevention

Preventing deadlocks involves removing one of the four conditions.

(a) Infinite resources (buy a very large disk)
(b) No sharing (totally independent threads)
(c) No waiting (phone company)
(d) Preempt resources (copying memory content to disk)
(e) Allocate all resources at the beginning (if you need 2 chopsticks, grab both at the same time)
(f) Make everyone use the same ordering in accessing resources. (all threads must grab semaphores in the order of `P(x)` and `P(y)`).
(g) A combination of techniques

4.3.2 Banker's Algorithm

One example of prevention algorithm (e) is the ***Banker's algorithm***, allows the sum of maximum resource needs of all threads to be greater than the total resources, as long as there is some way for all threads to finish without getting into deadlock.

1. A thread states its maximum resource needs in advance.
2. The OS allocates resources dynamically when resource is needed. A thread waits if its granting request would leads to deadlock (a request can be granted if some sequential ordering of threads is deadlock free.)

For example, you can allow a thread to proceed if the total available resources are greater than or equal to the maximum remaining resources that might be needed by this thread. To be specific, suppose two single-threaded processes are running on a machine with 256 Mbytes of physical RAM, and each process requires a maximum of 150 Mbytes of memory. Also, suppose each process has already allocated 100 Mbytes (200 out of 256 Mbytes of physical memory are allocated by both processes). Although the combined maximum (300 Mbytes) exceeds the physical memory limit (256 Mbytes), 56 Mbytes of available memory can actually carry any one of the two processes to completion, without the risk of exhausting the memory resource. Therefore, the current state of allocation for memory is considered ***safe***.

Physical memory:	256 Mbytes
Allocated:	200 Mbytes

	Maximum allocation
P_1	150 Mbytes
P_2	150 Mbytes

	Current allocation
P_1	100 Mbytes
P_2	100 Mbytes

On the other hand, if each process has already allocated 110 Mbytes, the system has reached an ***unsafe*** allocation state—the remaining 36 Mbytes is not enough for either process to complete.

Physical memory:	256 Mbytes
Allocated:	220 Mbytes

	Maximum allocation
P_1	150 Mbytes
P_2	150 Mbytes

	Current allocation
P_1	110 Mbytes
P_2	110 Mbytes

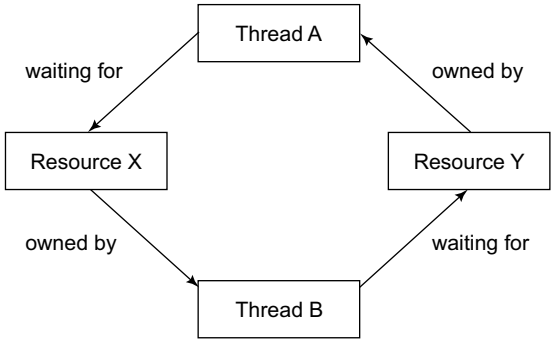
Since we cannot predict future, processes under the deadlock prevention approaches often overestimate resources and lead to inefficient use of resources.

Deadlock Detection and Recovery

The deadlock detection-and-recovery strategy is the most commonly used approach. The basic algorithm involves three steps:

1. Scan the resource allocation graph
2. Detect circular chains of requests
3. Recover from the deadlock

The idea is to remove the fourth deadlock condition of circular chain of requests.



Once the circular chain of requests are identified, there are some possible actions:

(a) Kill a thread and force it to give up resources.

(b) Rollback actions of a deadlocked thread.

Removing a deadlocked thread is not always possible, since a thread may own locks and resources. Removing a thread may leave the remaining system in an inconsistency state. Rolling back a thread requires ***checkpointing***, or taking snapshots of system states from time to time. So, a deadlocked thread can be ***rolled back*** (destroyed and restored) to a recent consistent state.

■ **Example** A file has to be shared among different processes, each of which has a unique no. The file can be accessed simultaneously by the processes as long as sum of all unique numbers associated with those processes which are currently accessing is less than N .

Comment on the following code whether it can extend such a control .

```
File_access monitor
Begin
Const int n
Const int N
Var delay : array[1..N ] of condition
Var delayed : array[1…N] of FALSE
Var total : int 0

Procedure entry Start Access ( id : int )
Begin
If total + id >= n do
Begin
Delayed[id] =TRUE
Delayed[id].wait
Total + = id
End
End

Procedure entry endAccess ( id :int )
Begin
  Total - = id
Let temp = n - total - 1
While temp > 0 and ~ delayed[temp]
Do
  Temp - = 1
If ( temp > 0 )
  Do {
      Delayed[id]=FALSE
      Delayed[id].signal
   }
end
end
```

The above code will guarantee mutual exclusion.

Whenever a process wants to access shared memory then it calls entry function of the monitor and before leaving the function calls endAccess function signaling some blocked processes if they can proceed .

■ **Example** There are 4 processes in the system and 4 resources types R1, R2, R3, R4 .There are 2 instances of R1 are available and remaining all are only 1 instance. Currently R1 is allocated to P1 , another R1 is allocated to P2 , R3 to P3 , R4 to P4 ,R2 to P3 .

The following requests are made

R1 ← P3

R1 ← P4

R3 ← P1

Is the system is deadlocked?

■ **Answer:** By drawing a resource graph for the above setup, even though we may find a cycle but it is dead lock free as there are multiple instances for R1 .

Normally in the case of problems of multiple resources we cannot say that there is a deadlock even though there is a circular wait or cycle as there are multiple resources of that type.

4.4 CPU Scheduling

Goals for a Scheduler

A ***scheduler*** handles the removal of the running process from the CPU and the selection of the next running process, based on a particular strategy. A scheduler aims to accomplish the following goals:

1. Minimise the ***wait time***, or the time for a process to receive its first unit of service from the processor.
2. Maximise the ***throughput***, or the number of completed jobs per unit time. There are two parts to maximising throughput:
   a. Minimise overhead (e.g., context switching).
   b. Efficient use of system resources (not only CPU, but disk, memory, etc.).
3. Achieve ***fairness***, so the CPU is shared among users in a fair way.

4.4.1 Scheduling Policies

First In, First Out (FIFO)

The **FIFO** strategy assigns the processor in the order of processor requests. The FIFO policy is ***nonpreemptive*** in the sense that a process keeps running on a CPU until it is blocked or terminated. The FIFO strategy is also known as the **FCFS** (first come first serve).

The main advantage of the FIFO policy is simplicity. The FIFO policy also provides a certain degree of fairness, since requests are served in order. However, the main disadvantage is that short jobs can get stuck behind long running jobs.

Round Robin (RR)

To overcome the FIFO problem, the ***round-robin*** policy releases the CPU from long-running jobs periodically (based on timer interrupts), so short jobs also can get a fair share of CPU. Therefore, round robin is a ***preemptive*** policy. Just about every real operating system does something of this flavour.

The interval between timer interrupts is referred to as the ***time slice***. After each time slice, the scheduler moves the process to the back of the queue. Choosing the size of time slice is tricky. If the time slice is too big, the wait time suffers. If the size is too small, throughput suffers, since the CPU spends a lot of time doing context switching.

In practice, we need to balance between these two scenarios. Typically, a modern operating system spends 1% overhead doing context switching.

Comparison Between FIFO and Round Robin

Assuming zero-cost time slice, is RR always better than FIFO? Not really. Suppose we have 10 jobs, each takes 100 seconds of CPU time, and the time slice for the round robin is 1 second. Let us look at job completion times.

Job Completion Times

Job #	FIFO	Round robin
1	100	991
2	200	992
…	…	…
9	900	999
10	1000	1000

Although both round robin and FIFO finish at the same time, the average turnaround time is much worse under round robin than under FIFO. Therefore, round robin is better for short jobs, but it is poor for jobs that are the same length.

STCF/SRTCF

STCF (Shortest Time to Completion First) runs whatever job has the least demand on CPU. The STCF is also known as **SJN** (Shortest Job Next) or **SJF** (Shortest Job First).

SRTCF (Shortest Remaining Time to Completion First) is just a preemptive version of STCF. If a job arrives that has a shorter time to completion than the remaining time on the current job, SRTCF immediately preempts the CPU for the new job.

The idea is to get short jobs out of the system. We can see a big improvement on turnaround times for short jobs, and a relatively small degradation of turnaround times for larger jobs, resulting in a better average turnaround time. In fact, STCF and SRTCF are the best you can possibly do at minimising the turnaround time.

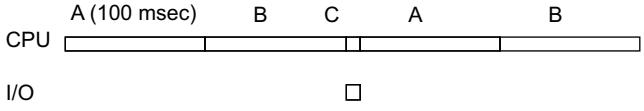
Comparison of SRTCF with FIFO and Round Robin

If all jobs are the same length, SRTCF becomes the same as FIFO. In other words, in that instance, FIFO is as good as you can do. Under SRTCF if jobs have varying length, short jobs do not get stuck behind long jobs.

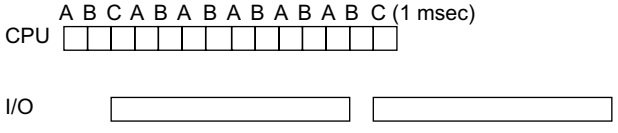
The following is an example that illustrates the benefit of SRTCF. Suppose we have three jobs: A, B, and C. A and B are both CPU bound, and each runs for a week. Job C consists of an I/O bound loop, which consumes 1 msec of CPU and 10 msec of disk I/O (assuming complete overlapping of I/O and CPU processing). By itself, C uses 90% of the disk. By itself, A or B could use 100% of the CPU. What happens if we try to share the system among A, B, and C?

Under FIFO, A and B will take two weeks before getting to job C.

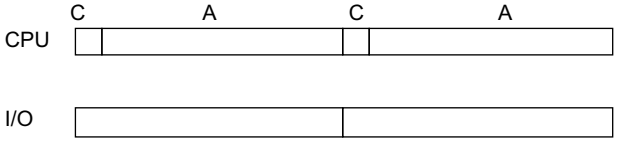
Under round robin with 100 msec of time slice, we can only get 5% of disk utilisation, since job C uses disk for 10 msec of every 201 msec.



However, if we reduce the time slice down to 1 msec, we can get 90% disk utilisation again—almost as good as job C running alone. Also, we have not slowed down A or B by all that much; they will get 90% of the CPU.



Under SRTCF, job C is run as soon as possible, then A or B is run to completion.



Overall, the advantage of the SRTCF is that it achieves the optimal average turnaround time. However, SRTCF can lead to the possibility of ***starvation***, where lots of short jobs can keep long jobs from making progress. In addition to unfair treatment of long jobs, SRTCF depends on predicting the CPU running time for various jobs.

One way to predict the length of a job is by asking the user. If a user cheats (if a job takes longer than specified), the job is killed. However, the user may not know any better. SRTCF is generally used as the optimal case for comparing various scheduling algorithms.

Multilevel Feedback Queues

One way to predict the future is from the existing behavior of processes. However, since a process may switch between CPU-bound and I/O-bound behaviors, a scheduling policy ideally can adapt changing behaviors based on recent behaviors.

Multilevel feedback queues use multiple queues, each with a different priority. An operating system does a round robin at each priority level—run highest priority jobs first; once those finish, run next highest priority, etc. A round-robin time slice increases exponentially at lower priorities.

	Priority	Time slice
Queue 1	1	1
Queue 2	2	2
Queue 3	3	4
Queue 4	4	8

Multilevel feedback queues adjust each job's priority as follows:

1. Job starts in the highest priority queue.
2. If time slice expires, drop the job by one level.
3. If time slice does not expire, push the job up by one level.

The resulting algorithm approximates SRTCF: CPU-bound jobs drop like a rock, while short-running I/O-bound jobs stay near the top. The multilevel feedback queues are still unfair for long running jobs. Some programs may sporadically generate meaningless I/Os just to keep the job's priority high.

Multilevel feedback queues are poor in handling situations like having one long-running job and 100 short-running ones. UNIX tries to ***age***, or increase the priority of, long-running jobs if they are not serviced for a period of time. However, the rate of aging jobs is difficult to determine. Also, for an overloaded system where no job can be serviced, every job increases in priority, and interactive short jobs begin to suffer as well.

Lottery Scheduling

Lottery scheduling is another adaptive scheduling approach that addresses the fairness problem of multilevel feedback queues. The lottery scheduler gives every job some number of lottery tickets, and on each time slice, it randomly picks a winning ticket. On average, the CPU time allotted is proportional to the number of tickets given to each job.

To approximate SRTCF, short jobs get more tickets, and long running jobs get fewer. To avoid starvation, every job gets at least one ticket (so everyone makes progress).

Lottery scheduling behaves gracefully as loads changes. Adding or deleting a job affects all jobs proportionally, independent of how many tickets each job has. For example, if short jobs get 10 tickets, and long jobs get 1 each, for different scenarios listed below, we can get meaningful behavior for all.

# short jobs/ # long jobs	% of CPU for each short job	% of CPU for each long job
1/1	91%	9%
0/2	N/A	50%
2/0	50%	N/A
10/1	10%	1%
1/10	50%	5%

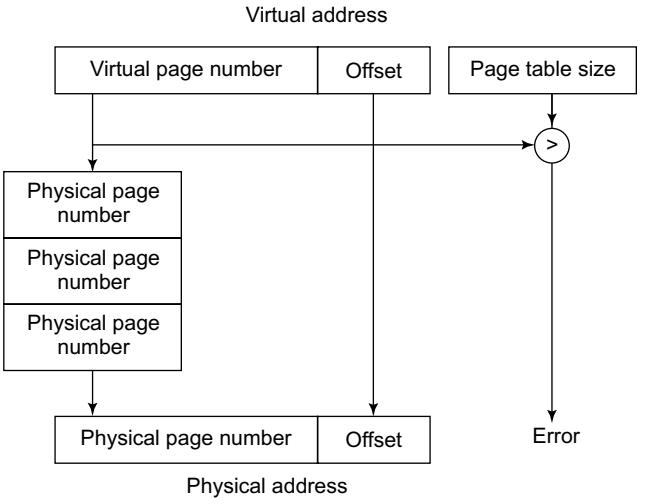
4.5 Memory Management and Virtual Memory

Main objective of virtual memory is to run large programs with limited RAM. This is achieved by loading small portions of the program into RAM as and when needed. Evidently, we have two types of VM systems:

Segmentation
Paging

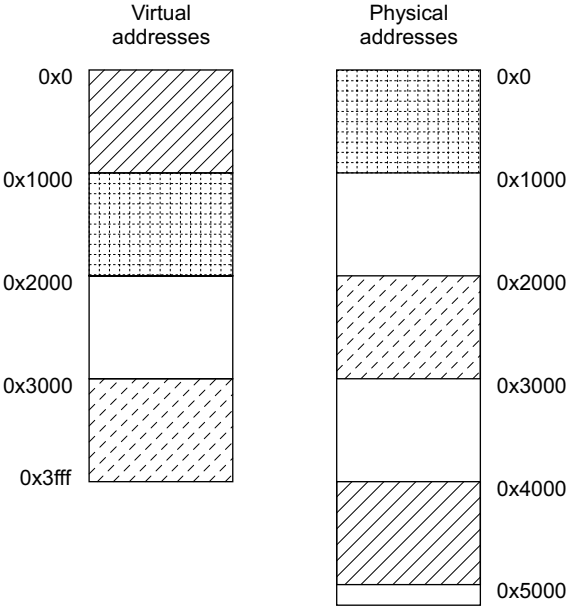
4.5.1 Paging

Paging-based translation reduces the complexity of memory allocation by having fixed-size chunks of memory, or ***pages***. The memory manager under paging can use a stream of 0s and 1s, or a ***bitmap***, to track the allocation status of memory pages. Each bit represents one page of physical memory—1 means a page is allocated; 0 means unallocated. Memory mapping is done at the granularity of a page, so a virtual page is mapped to a physical page of memory.



The following is an example of a page table with a page size of 4 Kbytes.

Virtual page number	Physical page number
0	4
1	0
3	2



Although it allows code sharing and easier memory allocation, paging has its own drawbacks. First, if a process sparsely uses its address space, the size of the page table is prohibitive, in particular, if a process has the starting virtual address of 0 for the code and $2^{31} - 1$ for stack (assuming a 32-bit architecture). With 1-Kbyte pages, a process will need 2 million table entries (memory spent for page table is called as table fragmentation). Second, if the page size is too big, paging suffers from ***internal fragmentation***, where allocated pages are not fully used.

Multi-Level Translation

To handle the sparse address space allocation, ***segmented-paging translation*** can break the page table into segments that are allocated as necessary—a significant reduction of page table size. The virtual address is now decomposed into three components: virtual segment number, virtual page number, and the offset.

Virtual segment number	Virtual page number	Offset

At the lowest level, memory allocation can still be done with a bitmap due to paging. Sharing can be performed at either the segment or the page level.

However, segmented paging also has a few drawbacks: (1) this approach still requires a certain overhead for storing additional pointers; (2) page tables still need to be contiguous; and (3) each memory reference now takes two or more memory table lookups.

Paged Page Tables

To further reduce the overhead of contiguous page tables, another solution uses ***paged page tables***, or a two-level tree of page tables. This approach reduces unwanted allocation of page table entries and can be generalised into ***multi-level paging***.

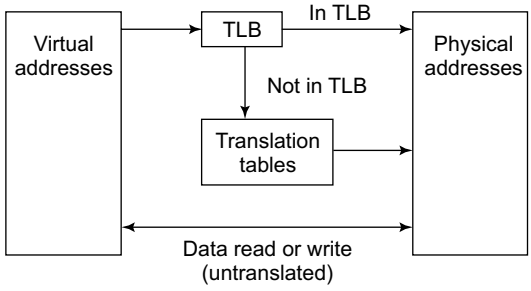
Multiple levels of page tables also mean multiple memory references before getting to the actual data. One way to speed up the lookup is to use ***Translation Lookaside Buffers*** (***TLB***s), which stores recent translated memory addresses for short-term reuses.

Inverted Page Tables

A simple and power approach to speed up lookups is to use a big hash table (***inverted page table***), where each virtual page number is hashed to a physical page number. The size of the table is independent of the size of address spaces, and proportional to the number of pages being used. However, managing hash collision can be complex and inefficient.

4.5.2 Caching Applied to Address Translation

Since a process often references the same page repeatedly, translating each virtual address to physical address through multi-level translation is wasteful. Therefore, modern hardware provides a ***Translation Lookaside Buffer*** (***TLB***) to track frequently used translations, to avoid going through translation in the common case. Typically, the TLB is on the CPU chip, so the lookup time is significantly faster than looking up from the memory.



Since Linux uses paging-based address translation, the remaining handout uses simple paging as the address translation scheme. The following is an example of the TLB content:

Virtual page number	Physical page number	Control bits
2	1	Valid, rw
–	–	Invalid
0	4	Valid, rw

TLB Lookups

There are a number of ways to look up a TLB entry.

1. Sequential search of the TLB table.
2. ***Direct mapping*** restricts each virtual page to using a specific slot in the TLB. For example, one approach is to use upper bits of the virtual page number to index the TLB.

```
if (TLB[UpperBits(vpn)].vpn == vpn) {
   return TLB[UpperBits(vpn)].ppn;
} else {
   ppn = PageTable(vpn);
   TLB[UpperBits(vpn)].control = INVALID;
```

```
    TLB[UpperBits(vpn)].vpn = vpn;
    TLB[UpperBits(vpn)].ppn = ppn;
    TLB[UpperBits(vpn)].control = VALID|READ| WRITE;
    return ppn;
}
```

By using the upper bits alone, two pages may compete for the same TLB slot. For example, a page referenced by the program counter may be competing for the same TLB entry that is used by the stack pointer page. Therefore, cache content may be tossed out even if still needed.

By using the lower bits alone, TLB references will be highly clustered, failing to take the full range of TLB entries. Thus, common approaches select a combination of high and lower bits.

3. ***Set associativity*** refers to having N separate hardware TLB banks, so N banks perform lookup simultaneously. The following diagram illustrates the two-way set associative cache.

   If either entries match, use the PPN; otherwise, translate and replace one of the two entries.

4. ***Fully associative*** cache allows looking up all TLB entries in parallel.

   If either entries match, use the PPN; otherwise, translate and replace one of the entries.

   Typically, the TLBs are small and fully associative. Hardware caches are larger, and are direct mapped or set associative to a small degree.

Replacements of TLB Entries

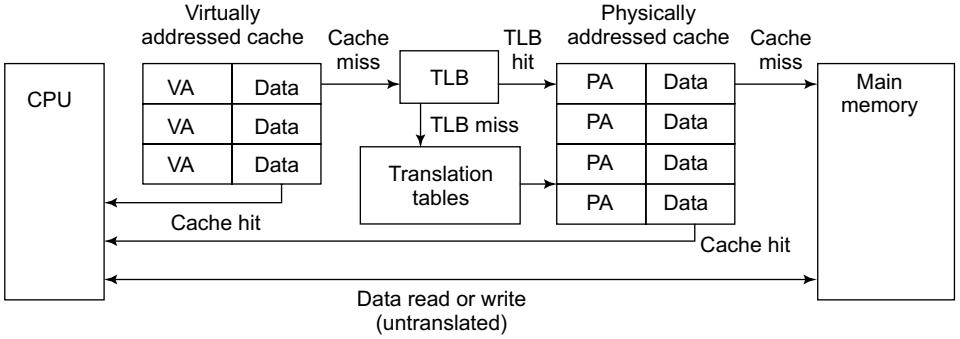
For direct mapping, the entry being mapped is replaced whenever there is a mismatch of the virtual page number. For set associative or fully associative cache, it is possible to replace a random entry, the least recently used entry, or the most recently used, depending on the reference patterns. In hardware, TLB replacement is mostly random because it is simple and fast. In software, memory page replacements typically do something more sophisticated. The tradeoffs are the CPU cycles and the cache hit rate.

Consistency Between TLB and Page Tables

Since different processes have different page tables, the on-chip TLB entries have to be entirely invalidated on context switches. An alternative is to include process IDs in TLB, at the additional cost of hardware and an additional comparison per lookup.

Relationship Between TLB and Hardware Memory Caches

A cache of memory values can be inserted in a number of places in the address translation process. The cache between the CPU and the translation tables is called the ***virtually addressed cache***. If the virtually addressed cache gets a cache hit, it returns the data immediately without translating the virtual address. The cache between the translation tables and the main memory is called the ***physically addressed cache***. If the physically addressed cache gets a cache hit, it returns the data without consulting the main memory.



If the cache data is modified, there are two ways to propagate the data back to the main memory.

1. The ***write-through*** approach immediately propagates update through various levels of caching, so the cached values are more consistent across different levels of memory hierarchy.
2. The ***write-back*** approach delays the propagation until the cached item is replaced from cache. The goal is to amortise the cost of update propagation across several cache modifications.

Since write-back policy is less costly, memory caches typically use write-back. On the other hand, file systems typically use write-through policy because they care more about the persistence of data to survive machines crashes.

4.5.3 Demand Paging

Demand paging allows the pages that are being referenced actively to be stored in memory; the remaining pages, on disk. Overall, demand paging allows bigger virtual address spaces and provides the illusion of infinite physical memory. Demand paging also allows more processes to fit in the physical memory, and to be running at the same time.

Demand Paging Mechanism

Demand paging means that page tables need to sometimes point to the disk locations as opposed to memory locations. A page table now needs an additional *present* or *valid* bit. If present, the page table entry points to a page in memory.

If not present, a reference to an invalid page (***page fault***) will cause the hardware to trap, and the OS performs the following steps while running another process in the meantime.

1. Choose an old page in memory to replace
2. If the old page has been modified, write contents back to disk
3. Change the corresponding page table entry and TLB entry
4. Load new page into memory from disk
5. Update page table entry
6. Continue the thread

4.5.4 Page Replacement Policies

There are a handful of replacement policies.

Random Replacement

TLBs use random replacement because this policy is easy to implement in hardware.

FIFO

The FIFO policy throws out the oldest page. The policy is fair in the sense of letting every page live in memory for the same amount of time. However, this policy may throw out pages that are heavily used, instead of those that are not frequently used.

MIN

The ideal strategy is to replace a page that will not be used for the longest time in the future.

LRU

The Least-Recently Used (LRU) policy replaces a page that has not been used for the longest time. If the past is a good predictor of the future, LRU is a good approximation to MIN. In real implementations, people do not even use LRU; they approximate it.

LFU

Another similar policy is to replace the Least Frequently Used page (LFU). Since LFU tracks the usage count of pages, it will take longer to replace pages with high count values, even if those pages are no longer in use.

■ **Example** Suppose that a cache can hold three pages. A process makes references to four pages: ABCABDADBCB. Under FIFO, we have the following page fault pattern. The page faults with the asterisk mark are compulsory misses.

Cache slot	A	B	C	A	B	D	A	D	B	C	B
1	*A					*D				C	
2		*B					A				
3			*C						B		

Under MIN, we have the following page fault pattern.

Cache slot	A	B	C	A	B	D	A	D	B	C	B
1	*A									C	
2		*B									
3			*C			*D					

Under LFU, we have the following page fault pattern. The grayed-out numbers indicate the number of times a page is referenced.

Cache slot	A	B	C	A	B	D	A	D	B	C	B
1	*A			2			3				
2		*B			2				3		4
3			*C			*D		2		C	

Under LRU, we have the following page fault pattern. The grayed-out X marks how recently a page is referenced.

Cache slot	A	B	C	A	B	D	A	D	B	C	B
1	*A			X			X			C	
2		*B			X				X		X
3			*C			*D		X			

Although LRU seems to work as well as MIN, LRU does not work well when the next reference is the least recently used page (i.e., a reference string of ABCDABCDABCD).

Cache slot	A	B	C	D	A	B	C	D	A	B	C	D
1	*A			*D			C			B		
2		*B			A			D			C	
3			*C			B			A			D

In fact, FIFO is as bad as LRU. However, MIN still works pretty well.

Cache slot	A	B	C	D	A	B	C	D	A	B	C	D
1	*A									B		
2		*B					C					
3			*C	*D								

Does Adding Memory Always Reduce the Number of Page Faults?

For the LRU and MIN replacement policies, the answer is yes, since the memory content for using X cache pages is a subset of the memory content for using X + 1 cache pages.

For the FIFO replacement policy, the answer is no. Since the memory content can be completely different with a different number of cache pages. In other words, it is possible for the FIFO replacement policy to get more page faults by increasing the cache size (***Belady's anomaly***). The following two tables demonstrate the Belady's anomaly.

Cache slot	A	B	C	D	A	B	E	A	B	C	D	E
1	*A			*D			*E					
2		*B			A					C		
3			*C			B					D	

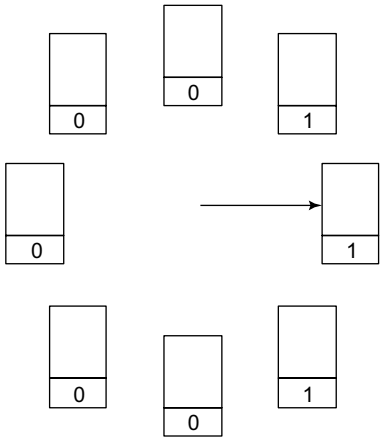
Cache slot	A	B	C	D	A	B	E	A	B	C	D	E
1	*A						*E				D	
2		*B						A				E
3			*C						B			
4				*D						C		

Implementing LRU

A perfect implementation of LRU requires a timestamp on each reference to a cache page, and the OS needs to keep a list of pages ordered by the timestamp. However, this implementation is too expensive considering the frequency of memory references. The common practice is to approximate the LRU behavior.

Clock Algorithm

The idea of approximating the LRU replacement policy is to replace an old page, not the oldest page. The ***clock algorithm*** arranges physical pages in a circle, with a clock hand. The hardware keeps a ***use bit*** for each physical page. The hardware sets the use bit to 1 on each reference. If the user bit is not set, it means that the page has not been referenced for some time.



On page fault, the clock hand starts to sweep clock-wise. If it encounters a use bit that is set to 1, it sets it to 0 and moves on. If the use bit is 0, the page is chosen for replacement.

The clock cannot tick indefinitely, since all each use bit is eventually cleared. A slow moving hand means that pages are either found quickly, or there are few page faults. A quick moving hand means lots of page faults, or lots of use-bit sets. One simple way to view the clocking algorithm is that of a crude partitioning of pages into young and old categories.

Nth Chance

The **nth chance algorithm** is a variant of the clocking algorithm, which does not throw a page out until the hand had swept by n times. With a large n, the nth chance algorithm has a better approximation of the LRU. A smaller n is more efficient. A common implementation is to use n = 1 for clean pages, n = 2 for dirty (modified) pages. After flushing the updates to the disk, n is set to 1.

States for A Page Table Entry

Many page table implementations maintain a four-bit per page table entry.

- ***use bit***: set when a page is referenced, cleared by the clock algorithm
- ***modified bit***: set when page is modified, cleared when a page is written to disk
- ***valid bit***: set when a program can make legitimate use of this page table entry
- ***read-only***: set for a program to read the page, but not to modify it (e.g., code pages)

Thrashing

Thrashing occurs when the memory is over committed, and pages are tossed out while still needed. For example, a program may have a hash table of 50 pages, while the physical memory has only 40 pages. Since the effective cache hit rate is 80%, the effective access time is computed as the following, assuming 2 nsec to make a memory reference and 2 msec to make a disk reference.

$$T = 80\%*2 \text{ nsec} + 20\%*(2 \text{ nsec} + 2 \text{ msec})$$
$$\sim= 400 \text{ micro seconds}$$

The OS should avoid thrashing, but the problem is that the system does not know what it is getting into. There are two ways to avoid thrashing.

1. At the level of a single process, a program should be written in a way that the memory requirement is small at a given time, so a process has better locality. For example, a matrix multiplication can be split into smaller sub-matrix multiplications.

2. At the level of multiple processes, the OS needs to figure out the memory needs per process, and run only the computations that fit in the physical RAM.

Working Set

A ***working set*** is a set of pages that a process has referenced in the last T seconds. If the T is infinite, the working set is the entire process. If T is smaller than the time to perform the page fault, the size of working set is one page and we have thrashing. A working-set-based algorithm is modeled on the observation that the number of page faults will not decrease significantly once the size of the working set has reached a threshold. Also, the threshold can be adjusted dynamically according to the rate of page faults. The goal is to use a minimum number of pages to get the fewest page faults.

For example, the LRU scheme results in many page faults with a cache of three pages, if we have the reference stream: ABCDABCDEFGH

Cache slot	A	B	C	D	A	B	C	D	E	F	G	H
1	*A			*D			C			*F		
2		*B			A			D			*G	
3			*C			B			*E			*H

With the cache size of four pages, the same reference stream causes only compulsory misses.

Cache slot	A	B	C	D	A	B	C	D	E	F	G	H
1	*A								*E			
2		*B								*F		
3			*C								*G	
4				*D								*H

However, if we further increase the cache size to hold eight pages, we can no longer reduce cache misses.

Cache slot	A	B	C	D	A	B	C	D	E	F	G	H
1	*A											
2		*B										
3			*C									
4				*D								
5									*E			
6										*F		
7											*G	
8												*H

A 2nd chance clocking algorithm can be modified to implement the working-set-based replacement policy.

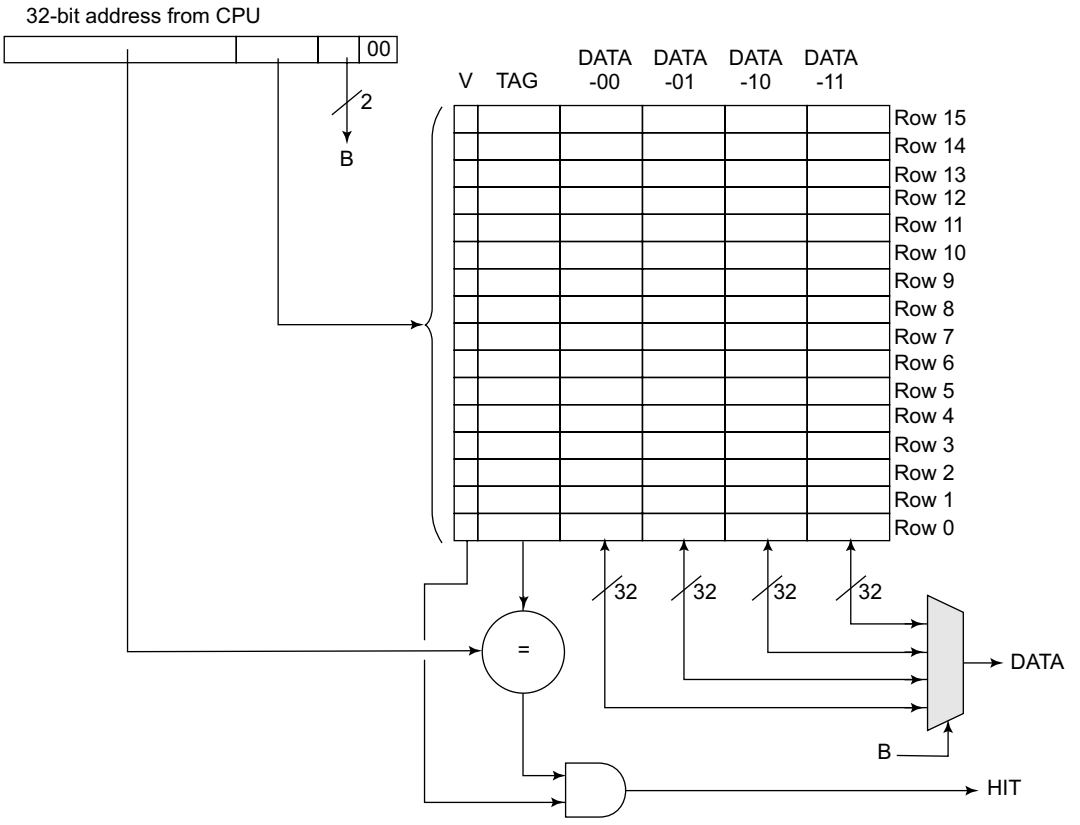
1. If n = 1, record the current time.
2. If n = 0, and if the age (current time – timestamp) of the page is greater than the threshold (no longer in the working set), the page is replaced.
3. If n = 0, and if the age of the page is younger than the threshold (still in the working set), find the oldest page and evict.

Global vs. Local Replacement Policies

For a ***global replacement policy*** (UNIX), all pages are in one pool for all processes. This policy is more flexible in the sense that if one process needs more memory, it can grab some from processes that do not require as much memory. The downside is that one misbehaving program can potentially drag down the whole system.

For the ***per-process replacement*** policy, each process has its own pool of pages (i.e., a separate clock for each process).

■ **Example** The following diagram illustrates a blocked, direct-mapped cache for a computer that uses 32-bit data words and 32-bit byte addresses.



A. What is the maximum number words of data from main memory that can be stored in the cache at any one time?

■ **Answer:** Maximum number of data words from main memory = (16 lines)(4 words/line) = 64 words

B. How many bits of the address are used to select which line of the cache is accessed?

■ **Answer:** With 16 cache lines, 4 bits of the address are required to select which line of the cache is accessed.

C. How many bits wide is the tag field?

■ **Answer:** Bits in the tag field = (32 address bits) - (4 bits to select line) - (4 bits to select word/byte) = 24 bits

D. Briefly explain the purpose of the one-bit V field associated with each cache line.

■ **Answer:** The tag and data fields of the cache will always have value in them, so the V bit is used to denote whether these value are consistent (valid) with what is in memory. Typically the V bit for each line in the cache is set to "0" when the machine is reset or the cache is flushed.

E. Assume that memory location 0×2045C was present in the cache. Using the row and column labels from the figure, in what cache location(s) could we find the data from that memory location? What would the value(s) of the tag field(s) have to be for the cache row(s) in which the data appears?

■ **Answer:** The cache uses ADDR[7:4] to determine where data from a particular address will be stored in the cache. Thus, location 0×0002045C will be stored in line 5 of cache. The tag field should contain the upper 24 bits of the address, i.e., 0×000204. Note that the bottom 4 bits of the address (0×C) determine which word and byte of the cache line is being referenced.

F. Can data from locations 0×12368 and 0×322FF8 be present in the cache at the same time? What about data from locations 0×2536038 and 0×1034? Explain.

■ **Answer:** Location 0×12368 will be stored in line 6 of the cache. Location 0×322F68 will be stored in line F of the cache. Since the lines differ, both locations can be cached at the same time. However, locations 0×2536038 and 0×1034 both would be stored in line 3 of cache, so they both could not be present in the cache at the same time.

G. When an access causes a cache miss, how many words need to be fetched from memory to fill the appropriate cache location(s) and satisfy the request?

■ **Answer:** There are 4 words in each line of the cache and since we only have one valid bit for the whole line, all 4 words have to have valid values. So to fill a cache line on a cache miss all 4 words would have to be fetched from main memory.

■ **Example** For this problem, assume that you have a processor with a cache connected to main memory via a bus. A successful cache access by the processor (a hit) takes 1 cycle. After an unsuccessful cache access (a miss), an entire cache block must be fetched from main memory over the bus. The fetch is not initiated until the cycle following the miss. A bus transaction consists of one cycle to send the address to memory, four cycles of idle time for main-memory access, and then one cycle to transfer each word in the block from main memory to the cache. Assume that the processor continues execution only after the last word of the block has arrived. In other words, if the block size is B words (at 32 bits/word), a cache miss will cost 1 + 1 + 4 + B cycles. The following table gives the average cache miss rates of a 1 Mbyte cache for various block sizes:

Block size (B)	Miss ratio (m), %
1	3.4
4	1.1
8	0.43
16	0.28
32	0.19

A. Write an expression for the average memory access time for a 1-Mbyte cache and a B-word block size (in terms of the miss ratio m and B).

Average access time = (1-m)(1 cycle) + (m)(6 + B cycles) = 1 + (m)(5+B) cycles

B. What block size yields the best average memory access time?

Block size B (m)	Access time Part (B) I + m (5+B)
1 (.0034)	1.204
4 (.0011)	1.099
8 (.0043)	1.056
16 (.0028)	1.059
32 (.0019)	1.0703

C. If bus contention adds three cycles to the main-memory access time, which block size yields the best average memory access time?

Block size B (m)	Access time Part (C) I + m (8+B)
1 (.0034)	1.306
4 (.0011)	1.132
8 (.0043)	1.069
16 (.0028)	1.067
32 (.0019)	1.076

D. If bus width is quadrupled to 128 bits, reducing the time spent in the transfer portion of a bus transaction to 25% of its previous value, what is the optimal block size? Assume that a minimum one transfer cycle is needed and don't include the contention cycles introduced in part (C).

Block size B (m)	Access time Part (D) I + m (5+roundup [B\4])
1 (.0034)	1.204
4 (.0011)	1.066
8 (.0043)	1.0301
16 (.0028)	1.0252
32 (.0019)	1.0247

■ **Example** The following four cache designs C1 through C4, are proposed for a processor. All use LRU replacement where applicable (e.g., within each set of a set associative cache).

Cache	C1	C2	C3	C4
Total Data words	8K	4K	8K	16K
Total Lines	8K	4K	4K	8K
Associativity	Fully	2-way S.A.	Direct Mapped	fully
Block size, words/line	1	1	2	2

A. Which cache would you expect to take the most chip area (hence cost) ?

Cache C4 would most likely take up the most chip area because it is fully associative, thereby requiring a comparator for each cache line, and because it has the most data word capacity.

B. Which cache is likely to perform worst in a benchmark involving repeated cycling through an array of 6K integers? Explain.

C2 would likely have the worst performance on a benchmark involving repeated cycling through an array of 6K integers since it is the only cache listed with less than 6K data word capacity.

C. It is observed that one of the caches performs very poorly in a particular benchmark which repeatedly copies one 1000-word array to another. Moving one of the arrays seems to cure the problem. Which cache is most likely to exhibit this behaviour? Explain.

We are told that one of the caches performs poorly in a particular benchmark which repeatedly copies one 1000-word array to another and that if one of the arrays is moved, the problem seems to be cured. This behaviour is most likely exhibited by cache C3 because it is a direct mapped cache which only has one location to put any particular address. If the lower bits (used to choose the cache line) for the addresses of the array overlap, poor performance could be observed. Moving the array so that the lower bits of the array addresses don't overlap could solve this problem.

D. Recall that we say cache A dominates cache B if for every input pattern, A caches every location cached by B. Identify every pair (A, B) of caches from the above list where A dominates B. Explain your reasoning.

So long as we are not using a random replacement strategy, it is always possible to come up with a benchmark that will make a particular type of cache have a miss on every data access. Thus, we cannot say that one particular type of cache always dominates another type of cache. However, we can compare two caches of the same type. Both C4 and C1 are fully associative caches with the same replacement strategy. We can say that C4 dominates C1 since C4 has a greater data word capacity.

■ **Example** The data-sheet for a particular byte-addressable 32-bit microprocessor reads as follows:
The CPU produces a 32-bit virtual address for both data and instruction fetches. There are two caches: one is used when fetching instructions; the other is used for data accesses. Both caches are virtually addressed. The instruction cache is two-way set-associative with a total of 2^{12} bytes of data storage, with 32-byte blocks. The data cache is two-way set-associative with a total of 2^{13} bytes of data storage, with 32-byte blocks.

A. How many bits long is the tag field in each line of the instruction cache?

There are $32 = 2^5$ bytes per block. The cache has 2^{12} total bytes and is 2-way set associative, so each set has 2^{11} bytes and thus $2^{11}/2^5 = 2^6$ cache lines. So the address is partitioned by the cache as follows:

[4:0] = 5 address bits for selecting byte/word within a block
[10:5] = 6 address bits for selecting the cache line
[31:11] = 21 address bit of tag field

B. How many address bits are used to choose which line is accessed in the data cache?

There are $32 = 2^5$ bytes per block. The cache has 2^{13} total bytes and is 2-way set associative, so each set has 2^{12} bytes and thus $2^{12}/2^5 = 2^7$ cache lines. So the address is partitioned by the cache as follows:

[4:0] = 5 address bits for selecting byte/word within a block
[11:5] = 7 address bits for selecting the cache line
[31:12] = 20 address bit of tag field

C. Which of the following instruction addresses would never collide in the instruction cache with an instruction stored at location 0x0ACE6004?

(A)	0×0BAD6004	(D)	0×0ACE6838
(B)	0×0C81C81C	(E)	0×FACE6004
(C)	0×00000004	(F)	0×0CEDE008

Collisions happen when instruction addresses map to the same cache line. Referring to the answer for (A), address bits [10:5] are used to determine the cache line, so location 0x0ACE6004 is mapped to cache line 0.

Only (D) 0x0ACE6838 maps to a different cache line and hence could never collide in the instruction cache with location 0x0ACE6004.

D. What is the number of instructions in the largest instruction loop that could be executed with a 100% instruction cache hit rate during all but the first time through the loop?

The instruction cache hold 2^{12} bytes or 2^{10} = 1024 instructions. So if the loop had 1024 instructions it would just fit into the cache.

■ **Example** The following questions ask you to evaluate alternative cache designs using patterns of memory references taken from running programs. Each of the caches under consideration has a total capacity of 8 (4-byte) words, with one word stored in each cache line. The cache designs under consideration are:

DM: a direct-mapped cache.

S2: a 2-way set-associative cache with a least-recently-used replacement policy.

FA: a fully-associative cache with a least-recently-used replacement policy.

The questions below present a sequence of addresses for memory reads. You should assume the sequences repeat from the start whenever you see **"..."**. Keep in mind that byte addressing is used; addresses of consecutive words in memory differ by 4. Each question asks which cache(s) give the best hit rate for the sequence. Answer by considering the steady-state hit rate, i.e., the percentage of memory references that hit in the cache after the sequence has been repeated many times.

A. Which cache(s) have the best hit rate for the sequence 0, 16, 4, 36, ...

DM: locations 4 and 36 collide, so each iteration has 2 hits, 2 misses.

S2: 100% hit rate. 0 and 16 map to the same cache line, as do 4 and 36, but since the cache is 2-way associative they don't collide.

FA: 100% hit rate. The cache is only half filled by this loop.

B. Which cache(s) have the best hit rate for the sequence 0, 4, 8, 12, 16, 20, 24, 28, 32, ...

DM: locations 0 and 32 collide, so each iteration has 7 hits, 2 misses.

S2: locations 0, 16 and 32 all map to the same cache line. The LRU replacement strategy replaces 0 when accessing 32, 16 when accesing 0, 32 when accessing 16, etc., so each iteration has 6 hits, 3 misses.

FA: has 0% hit rate in the steady state since the LRU replacement strategy throws out each location just before it's accessed by the loop!

C. Which cache(s) have the best hit rate for the sequence 0, 4, 8, 12, 16, 20, 24, 28, 32, 28, 24, 20, 16, 12, 8, 4, ...

All caches perform the same -- locations 0 and 32 trade places in the caches, so each iteration has 14 hits and 2 misses.

D. Which cache(s) have the best hit rate for the sequence 0, 4, 8, 12, 32, 36, 40, 44, 16, ..

DM: 32 collides with 0, 36 with 4, 40 with 8, 44 with 12, so each itreation has only 1 hit and 8 misses.

S2: locations 0, 16 and 32 trade places in the cache, so each iteration has 6 hits and 3 misses.

FA: 0 hits since LRU throws out each location just before it's accessed by the loop.

E. Assume that a cache access takes 1 cycle and a memory access takes 4 cycles. If a memory access is initiated only after the cache has missed, what is the maximum miss rate we can tolerate before use of the cache actually slows down accesses?

If accesses always go to memory, it takes 4 cycles per access. Using the cache the average number of cycles per access is 1 + (miss rate)*4

So if the miss rate is larger than 75% the average number of cycles per access is more than 4.

4.6 File Systems

A file system consists of the following major components:

- ***Disk management*** organises disk blocks into files.
- ***Naming*** provides file names and directories to users, instead of track and sector numbers.
- ***Protection*** keeps information secure from other users
- ***Reliability*** protects information loss due to system crashes.

User vs. System View of a File

At the user level, a user is only aware of the presence of individual files. At the system call level, each call processes a collection of bytes. However, at the operating system level, a *block* is the logical transfer unit, while a sector is the physical transfer unit. In UNIX, a block size is 4 Kbytes, which is greater than a 512-byte sector.

Given the discrepancy of user and system perceptions, a user access has to be translated into system accesses.

- If a process wants to read bytes 2 to 12, the operating system has to fetch the block containing those bytes, and return those bytes to the process.
- If a process wants to write bytes 2 to 12, the operating system has to fetch the block containing those bytes, modify those bytes, and write out the block.

Even though the system call provides byte-level accesses (e.g., `getc` and `putc`), the OS operates in blocks. Therefore, a ***file*** is a named collection of blocks.

File Usage Patterns

For a file system to perform well, a designer needs to understand user access behaviors. In general, there are three ways to access a file.

1. ***Sequential access***—bytes are accessed in order.
2. ***Content-based access***—bytes are accessed according to constraints on byte contents (e.g., return 100 bytes starting with "bhale bhale").
3. ***Random access***—bytes are accessed in any order.

Most file systems do not provide the content-based access.

There are also several file usage patterns.

- Most files are small (e.g., `.login` and `.c` files), and most file references are to small files.
- Large files use up most of the disk space (e.g., `mp3` files).
- Large files account for most of the bytes transferred between memory and disk.

These usage patterns are bad news for file system designers. To achieve high performance, a designer needs to make sure that small files are accessed efficiently, since there are many of them, and they are used frequently. A designer also needs to make sure that large files are accessed efficiently, since they consume most of the disk space, and account for most of the data movement.

4.6.1 Free Data Blocks Management

- **Bitmap based**
- **Linked List based**

In the bitmap based method, a bit vector of size equal to the number of disk blocks is used. If ith block is free then the bit vector value at ith location will be 0 otherwise 1. Thus, if we want ten data blocks to store a file content then we have to search for 10 consecutive 0's in this vector (of course, assuming contiguous allocation policy). This becomes more like string matching, which is little computational demanding. Also, memory requirements for storing bit vector also increases with the increase in the disk size.

■ **Example** If we are having a partition in which data blocks uses 4 byte addresses then the max number of data blocks in the disk will be 2^{32} and thus the bit vector size is also 2^{32} bits i.e , 512 MB.

In operating systems such as DOS and windows the datablock allocation policy is continuous. we are required to find out contiguous datablocks to store a file.

■ **Example** If we wanted to store a file which requires 100 data blocks then we are required to find a substring of 100 1's in the bit vector which becomes a substring matching which is usually more time consuming.

In the case of linked list approach, details about the free data blocks are maintained in a linked list known as free list. Space requirements and searching for free data blocks are relatively easy here.

Here, also we will be using first fit, best fit, worst fit and next algorithms to select free datablock.

Disk Management Policies

A file contains a ***file header***, which associates the file with its disk sectors. Also, a file system needs a ***disk allocation bitmap*** to represent free space on disk, one bit per block. Blocks are numbered in cylinder-major order, so that adjacent numbered blocks can be accessed without seeks or rotational delay.

Block number(s)	Sector number(s)	Track number(s)	Cylinder number(s)
0	0-7	0	0
1	8-15	0	0
…			
31	247-255	0	0
32	0-7	1	0
…			
511	247-255	15	0
512	0-7	0	1

Although the OS keeps a cache of recently used disk blocks in memory, we will ignore caching for now when comparing the performance of different disk management policies.

4.6.2 Contiguous Allocation

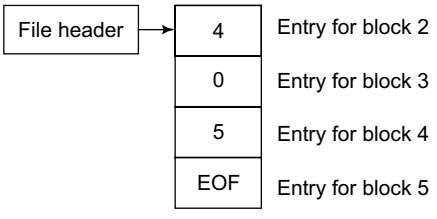
For ***contiguous allocation***, file blocks are stored contiguously on disk. A user specifies the file size in advance, and the file system will search the disk allocation bitmap according to various allocation policies (e.g., first-fit and best-fit policies) to locate the space for the file. The file header contains the first block of the file on disk, and the number of blocks in the file.

Contiguous allocation provides fast sequential access. A random disk location in a file can be trivially computed by adding an offset to the first disk block location of the file. The disadvantages are external fragmentation and difficulties to grow files.

Linked-List Allocation

For ***linked-list allocation***, each file block on disk is associated with a pointer to the next block. Perhaps, the most popular example is the MS-DOS file system, which uses the File Attribute Table (FAT) to implement linked-list files. A file header points to the table entry of the first file block, and the content of the file block contains the table entry number of the next block. A special marker is used to indicate the end of the file.

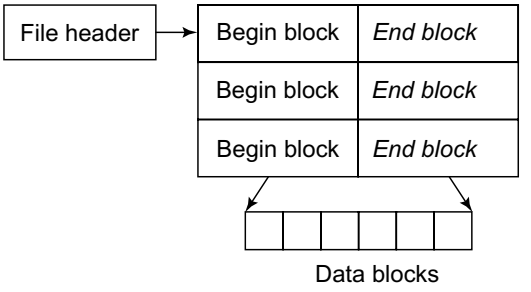
One advantage of the linked-list approach is that files can grow dynamically with incremental allocation of blocks. However, sequential accesses may suffer since blocks may not be contiguous. Random accesses are horrible and involve multiple sequential searches. Finally, linked-list allocation is unreliable, since a missing pointer can lead to loss of the remaining file.



Segment-Based Allocation

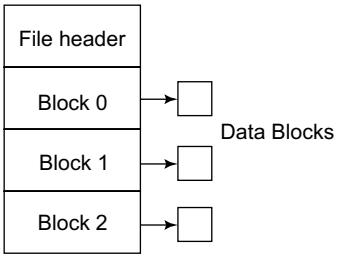
Segment-based allocation uses a segment table to allocate multiple regions of contiguous blocks. The file header points to a table of begin-block and end-block pairs.

Segment-based allocation provides the flexibility to break the allocation into a number of contiguous disk regions, while it still permits contiguous allocation to reduce disk seek time. However, as the disk becomes increasingly fragmented, in the extreme case, segment-based allocation needs a bigger and bigger table to locate piece-wise contiguous blocks. As an extreme case, segment-based allocation can potentially need one table entry per disk block. In addition, under this scheme, random accesses are not as fast as the contiguous allocation, since the file system needs to locate the pair of begin block and end block that contains the target byte before making the disk accesses.



Indexed Allocation

Indexed allocation uses an index to directly track the file block locations. A user declares the maximum file size, and the file system allocates a file header with an array of pointers big enough to point to all file blocks.

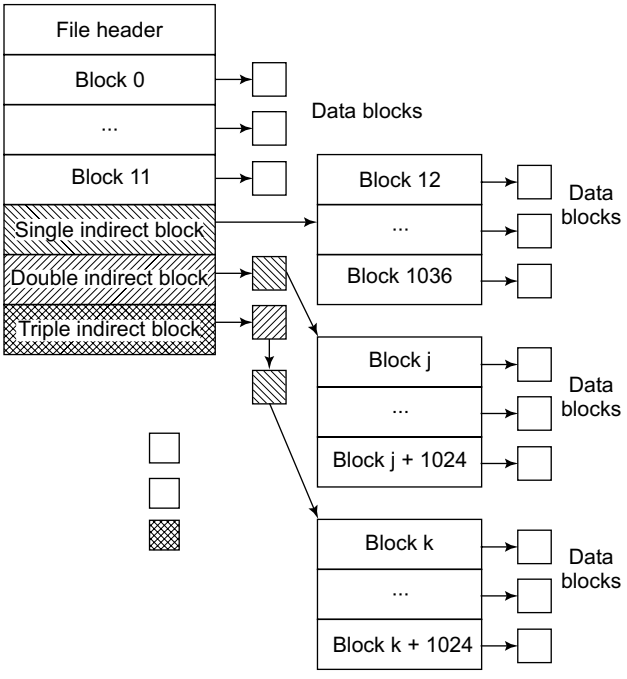


Although indexed allocation provides fast disk location lookups for random accesses, file blocks may be scattered all over the disk. A file system needs to provide additional mechanisms to ensure that disk blocks are grouped together for good performance (e.g., disk defragmentor). Also, as a file increases in size, the file system needs to reallocate the index array and copy old entries. Ideally, the index can grow incrementally.

Multilevel Indexed Allocation

Linux uses ***multilevel indexed allocation***, so certain index entries point to index blocks as opposed to data blocks. The file header, or the `i_node` data structure, holds 15 index pointers. The first 12 pointers point to data blocks. The 13^{th} pointer points to a ***single indirect block***, which contains 1,024 additional pointers to data blocks. The 14^{th} pointer in the file header points to a ***double indirect block***, which contains 1,024 pointers to single indirect blocks. The 15^{th} pointer points to a ***triple indirect block***, which contains 1,024 pointers to double indirect blocks.

This skewed multilevel index tree is optimised for both small and large files. Small files can be accessed through the first 12 pointers, while large files can grow with incremental allocations of index blocks. However, accessing a data block under the triple indirect block involves multiple disk accesses—one disk access for the triple indirect block, another for the double indirect block, and yet another for the single indirect block before accessing the actual data block. Also, the number of pointers provided by this data structure cap the largest file size. Finally, the boundaries between the last four pointers are somewhat arbitrary. With a given block number, it is not immediately obvious as to of which of the 15 pointers to follow.



■ **Example** Let Block size = A

Block addresses = b bytes

Blocking factor = N = A / b

According to Unix (Indexed) allocation strategy which uses third level indirection, largest possible single file size = 10 + N + N^2 + N^3 blocks (Do remember in Unix direct blocks are 10 only).

■ **Example** A Unix filesystem which uses 1K block size and 2 byte block addresses. Then calculate what is the largest possible single file size.

■ **Answer :** A = 1 KB

B = 2 bytes

N = 1024 / 2 = 512

So , Largest single file size = 10 + 512 + 512^2 + 512^3 blocks

~= 512^3 blocks

= 128 X 2 X 512 X 2 X 512 X 1 KB

= 128 GB

This is only theoretically possible value. But , we can disapprove it practically , as 2 byte addresses are used for data block . So, max of 2^{16} data blocks only available in the disk.

So, disk size only = 2^{16} X 1 KB = 64 MB.

■ **Example** A disk is formatted with 2 KB block size and 4 byte addresses , then find out largest possible single file size ?

■ **Answer:** N = 2 KB / 4 = 512

Max single file size = 256 GB

This is more practical as really as disk size = 8 Tera bytes .

■ **Example** If N is the blocking factor, how many blocks are used as index block in the worst case?

Max no of data blocks spend on Index blocks

= 1 + (1 + N) + (1 + N + N^2)

4.6.3 Inode Structure in Unix

Inode of a file does not contain the name of the file.

It contains the following things:

UID
GID
Permissions
Times
Links
10 direct addresses
1st level indirect addr
2nd level indirect addr
3rd level indirect addr

Inode Structure

- Assuming that a file's inode is already available in RAM (i.e, already located) and then to access any byte of a file, we require at most 4 disk accesses, where as to know the data block number which contains this byte may require at most 3 disk accesses.
- MINIX and XENIX file systems use uptil second level indirection only.
- Inode numbers of a file and its hard link file will be same whereas a file and its soft link will not be same.

- Whenever a hard link file is created, link count of the files inode will be incremented by 1 whenever either a hard link file or original file is deleted, link count value will be reduced by 1.
- When the link count value becomes 0 then all the data blocks consumed by that file will be marked as free and even the inode is marked as free.
- Symbolic links are extensively used to fine tune the application software, they can be also used to link files in different partitions.
- Inode is a 64 byte long.
- 0 is the inode no for root directory " / ".
- Binary name of a file or file descriptor or file handle of a file ' : '
- Whenever we open a file, the file's info such as mode of opening , permissions , offset , pointer to the virtual node all are maintained in a table known as Open file table.
- That Row index is known as file descriptor of that file. This number is meaningful and is associated with that file as long as the process is running and the file is not closed. This number is also known as binary name of that file and it can be called as dynamic number associated with that file.
- Inode number of a file can be called as static name or static number related to that file. Never inode number of a file is going to change during that files life.

Major numbers and minor numbers

Normally device drivers are set of functions. Some of them are automatically located during the booting time while others can be loaded as and when required.

In the Kernel space, there exists a device driver table, probably one for block oriented devices and character oriented devices.

Major number of a file is the one which conveys which device driver to be accessed while accessing that file. This is also a integer number (In current Unix kernel this is an 8 – bit no). To be specifically it is the row index of the device driver table in the kernel space which contains pointer to a structure having names of the functions which are to be used while really doing the operation on that device.

For some devices major numbers will be made fixed i.e, during the booting time the respective device drivers are registered such that their info is stored in specific row of the device driver table.

Minor numbers are used to distinguish the devices which use the same device driver software.

It is acceptable 2 files from different partitions to be having same inode numbers. However, as their partitions are different their major numbers may differ and minor numbers will differ.

Swap partitions are not indexed file systems like Unix file systems like Unix file systems rather they are realised as contiguous allocation.

FAT 12 filesystem is seen on the Floppy.

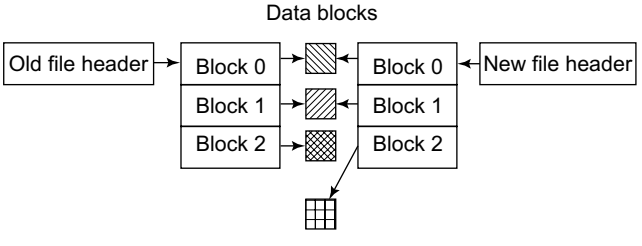
In Unix O.S, a directory block contains name of files and subdirectories in it along with their inode numbers, whereas in the case of FAT based systems, a directory info contains Names and the starting cluster no.

If the data block size increases, wastage through internal fragmentation increases.

If the block size reduces the file access time increases.

Inverted Allocation

If we use a disk as a device for backups (e.g., tape), the storage capacity may be the primary concern, and the speed of disk may not matter (as long as it is faster than tape). Since backup storage keeps track of all modifications, a small modification to a large file results in storing the entire modified file, ***Inverted allocation*** allocates a disk block by hashing the file block content to a disk block location. By doing so, different file blocks of the same content (e.g., empty blocks) can share the same disk block for storage. For example, if one block is modified in a N-block file, the storage requirement for both files is N + 1 blocks.



4.6.4 Disk System Performance Improvement

- If we happen to have more than one controller (say 2) and 2 hard disk drives then it is recommended to connect these two drives one for each controller rather than using them as master and slave for single controller.

In personal computers , interrupt no 14 is assigned to hard disks (IDE or EIDE).

RAID drives are exclusively designed to improve the file system performance interms of access times and also in terms of fault taulerency. In Parallel, we can access file from several disks to improve performance, if the file is distributed in disks.

Interleaving of Sectors

The interleaving factor is used to reduce the rotational latencies effects, specifically if we assume the sectors are sequentially numbered physically then after reading a sector, we require some finite amount of time to transfer the same to the RAM, during which period disk assembly will not be stopped from its rotation. Thus, when the head is ready to read the next sector, that would have been already crossed. Thus, the head has to wait till it comes. Inorder to reduce this delay, sector no's are staggered.

In today's hard disk drives, the interleaving factor is not under the control of the program.

Rather it is a function (complex function) of speed of the processor, bus, rotational speed of the drive etc.

Low level formatting tools will be giving freedom to change this interleaving factor.

4.6.5 Disk Caching

Disk Cache can be either hardware Cache or Software Cache. Main theme behind the disk caching is also same as memory caching i.e, most recently read data blocks are assumed to be requiring in the near future thus, retain them in the Cache memories.

Such that next time when we require the same datablock, we would be taking from the disk cache rather than from disk. Thus, program execution becomes fast.

Normally unlike memory caches, disk caches are not specifically designed memories.They are memories only with faster access times.

Disk caches are even realised in RAM's by the Operating systems. Very commonly the data blocks stored in the disk caches are chained.

Compared to memory caches, disk caches realization is little different. Especially in memory caches specially designed hardwares are employed whereas it is not true with disk caches.

The data blocks which are loaded into disk cache are circularly chained such that replacement, insertions, removals etc are little easy.

Here also, Cache management policies such as FIFO, LRU, NRU can be used. However LRU type is in wide use.

High level formatting creates a logical view of the filesystem i.e, it treats as 1-D array of data blocks, in the mean time it collects the info about the free data blocks as well.

Now a days there is no facility for low level formatting as the floppies or hard disks are coming pre-formatted.

While doing formatting or creating file system it contains some number of node, as some data blocks may be spoiled in between or some datablocks may not be movable like "command.com" (ex : format A:\ s)

In free list based data block management first free list is created during the formatting (high level formatting). In this list every node contains starting free data block and number of free data blocks. First few nodes details are saved in superblock and then remaining nodes details are stored in data block area of the partition. During the booting time, this free data blocks info is read from the superblock into the memory.

Only if currently available free list in the RAM cannot satisfy the requirement of a file then remaining part of the linked list is read from the data block area of the disk into memory.

In some O.S's whenever a file is removed , the data blocks whatever it was using will return to the free list available in the core memory (RAM). In some O.S's they will be kept at front of the list whereas some will be maintaining at the end of the list.

If we have written the free data block info and place at front of the linked list then extending "undelete" service becomes little difficult.

Major advantage of free list based data block management is the memory requirements and also disk requirements are relatively less.

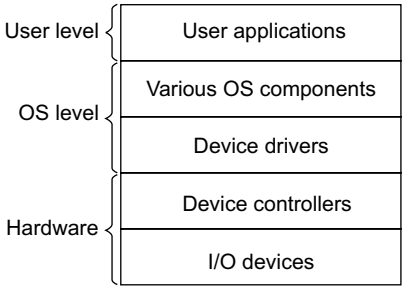
If External fragmentation increases then free list grows .

Under ideal conditions, after defragmentation one node should be available in the free list. However, if there exists some non-movable code or spoiled datablocks at the end then after defragmentation also free list size may be more than 1 node.

4.7 I/O Systems

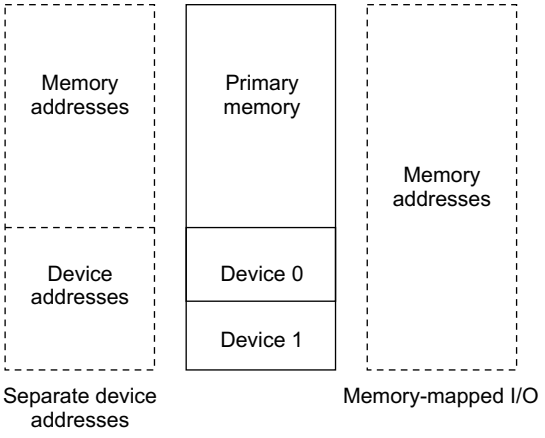
I/O devices can be roughly divided into two categories.
A ***block device*** (e.g., disks) stores information in fixed-size blocks, each one with its own address.
A ***character device*** (e.g., keyboards, printers, network cards) delivers or accepts a stream of characters, and individual characters are not addressable.

A device is connected to a computer through an electronic component, or a ***device controller***, which converts between the serial bit stream and a block of bytes and performs error correction if necessary. Each controller has a few device registers that are used for communicating with the CPU, and a data buffer that an OS can read or write. Since the number of device registers and the natures of device instructions vary from device to device, a ***device driver*** OS component is responsible hiding the complexity of an I/O device, so that the OS can access various devices in a relatively uniform manner.



Device Addressing

In general, there are two approaches to addressing these device registers and data buffers. The first approach is to assign each device a dedicated range of device addresses in the physical memory, so accessing those device addresses requires special hardware instructions associated with individual devices. The second approach (***memory-mapped I/O***) is not to distinguish device addresses from normal memory addresses, so devices can be accessed the same way as normal memory, with the same set of hardware instructions.



Device Accesses

Regardless of the device addressing approach, the operating system has to track the status of a device for exchanging data. The simplest approach is to use ***polling***, where the CPU repeatedly checks the status of a device for exchanging data.

However, wasting CPU cycles on busy-waiting is undesirable. A better approach is to use ***interrupt-driven I/Os***, where a device controller notifies the corresponding device driver when the device is available. Although the interrupt-driven approach is much more efficient than polling, the CPU is still actively involved in copying data between the device and

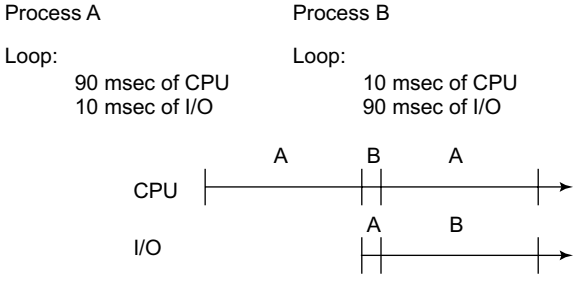
memory. Also, interrupt-driven I/Os still impose high overheads for character devices. For example, a printer raises one interrupt per byte, so the overhead of interrupt far exceeds the cost of transmitting a single byte.

An even better approach is to use an additional ***Direct Memory Access*** (***DMA***) controller to perform the actual movements of data, so the CPU can use the cycles for computation as opposed to copying data.

The use of DMA alone still has room for improvement. Since a process cannot access the data that is being brought into memory at the moment, due to mutual exclusion, a more efficient approach is to pipeline the data transfer. The ***double buffering*** technique uses two buffers in the following way: while one is being used, the other is being filled. Double buffering is also used extensively for graphics and smooth animation. While the screen displays an image frame from one buffer in the video controller, a separate buffer is being filled pixel-by-pixel in the background, so a viewer does not see the line-by-line scanning on the screen. Once the background buffer is filled, the video controller switches the roles of the two buffers and displays from the freshly filled buffer.

4.7.1 Overlapped I/O and CPU Processing

By freeing up CPU cycles while devices are serving requests, CPU-bound processes can be executed concurrently with I/O-bound processes. For example, if process A is CPU-bound, and process B is I/O-bound, the system as a whole can reach high utilisation by overlapping CPU and I/O processing effectively.



4.7.2 Disk as An Example Device

The hard disk is a 30-year-old storage technology, and is incredibly complicated. A modern hard drive comes with 250,000 lines of micro code to govern various hard drive components.

Hardware Characteristics

Briefly, a hard drive consists of a disk arm and disk platters. ***Disk platters*** are coated with magnetic materials for recording. The ***disk arm*** moves a comb of disk heads, among which only one ***disk head*** is active for reading and writing.

One fascinating detail is that heads are aerodynamically designed to fly as close to the surface as possible. In fact, the distance is so close that there is no room for air molecules, and a hard drive is filled with special inert gas to fly disk heads. If a head touches the surface, it results in a head crash, which scrapes off magnetic information.

Each disk platter is further divided into concentric ***tracks*** of storage, and each track is divided into ***sectors*** (typically 512 bytes). Each sector is a minimum unit of disk storage. A ***cylinder*** consists of all tracks with a given arm position. (Fig. 4.1)

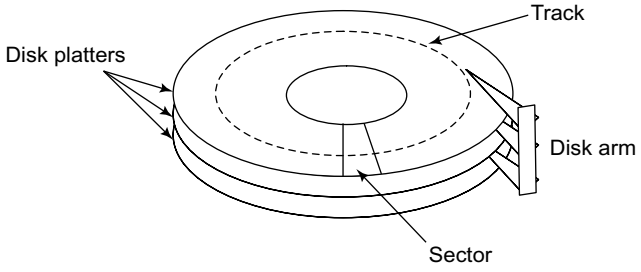


Figure 4.1 Disk Structure

A modern hard drive also takes advantage of the disk geometry. Disk cylinders are further grouped into ***zones***, so zones near the edge of the disk can store more information than zones near the center of the disk due to the differences in stor-

age area (also known as ***zone-bit recording***). More information stored in outer zones also means that the transfer rate (rotational speed multiplied by the information stored in a cylinder) is higher near the edge of the disk.

Since, moving a disk arm from one track to the next takes time, the starting position of the next track is slightly skewed (***track skew***), so that a sequential transfer of bytes across multiple tracks can incur minimum rotational delay.

A hard drive also periodically performs ***therm-calibrations***, which adjusts the disk head positioning according to the changes in the disk radius caused by temperature changes. To account for other minor physical inaccuracies, typically 100 to 1000 bits are inserted between sectors.

A Simple Model of Disk Performance

The access time to read or write a disk section includes three components:

1. ***Seek time***: the time to position heads over a cylinder (~8 msec on average).
2. ***Rotational delay***: the time to wait for the target sector to rotate underneath the head. Assuming a speed of 7,200 rotations per minute, or 120 rotations per second, each rotation takes ~8 msec, and the average rotational delay is ~4 msec.
3. ***Transfer time***: the time to transfer bytes. Assuming a peak bandwidth of 58 Mbytes/sec, transferring a disk block of 4 Kbytes takes 0.07 msec.

Thus, the overall time to perform a disk I/O = seek time + rotational delay + transfer time.

The sum of the seek time and the rotational delay is the disk ***latency***, or the time to initiate a transfer. The transfer rate is the disk ***bandwidth***.

If a disk block is randomly placed on disk, then the disk access time is roughly 12 msec to fetch 4 Kbytes of data, or a bandwidth 340 Kbytes/sec.

If a disk block is randomly located on the same disk cylinder as the current disk arm position, the access time is roughly 4 msec without the seek time, or a bandwidth of 1.4 Mbytes/sec.

If the next sector is on the same track, the access time is 58 Mbytes/sec without the seek time and the rotational delay.

Therefore, the key to using the hard drive effectively is to minimise the seek time and rotational latency.

Disk Tradeoffs

One design decision is the size of disk sector.

Sector size	Space utilisation	Transfer rate
1 byte	8 bits/1008 bits (0.8%)	80 bytes/sec (1 byte / 12 msec)
4 Kbytes	4096 bytes/4221 bytes (97%)	340 Kbytes/sec (4 Kbytes / 12 msec)
1 Mbyte	(~100%)	58 Mbytes/sec (peak bandwidth)

A bigger sector size seems to get a more effective transfer rate from the hard drive. However, this allocation granularity is wasteful if only 1 byte out of 1 Mbyte is needed for storage.

Disk Controller

Two popular disk controllers are SCSI (Small Computer Systems Interface), and IDE (Integrated Device Electronics). Since they are not a part of the OS, please surf the net for more information.

4.7.3 Disk Device Driver

One major function of the disk device driver is to reduce the seek time for disk accesses. Since disk can serve only one request at a time, the device driver can schedule the disk request in such a way to minimize disk arm movements. There are a handful of disk scheduling strategies. For further reference consult Nutt's book for detailed examples.

FIFO

Requests are served in the order of arrival. This policy is fair among requesters, but requests may land on random spots on disk. Therefore, the seek time may be long.

SSTF (Shortest Seek Time First)

The shortest seek time first approach picks the request that is closest to the current disk arm position. (Although called the shortest seek time first, this approach actually includes the rotational delay in calculation, since rotation can be as long as

seek.) SSTF is good at reducing seeks, but may result in starvation.

SCAN
SCAN implements an elevator algorithm. It takes the closet request in the direction of travel. It guarantees no starvation, but retains the flavor of SSTF. However, if a disk is heavily loaded with requests, a new request at a location that has been just recently scanned can wait for almost two full scans of the disk.

C-SCAN (Circular SCAN)
For C-SCAN, the disk arm always serves requests by scanning in one direction. Once the arm finishes scanning for one direction, it quickly returns to the 0^{th} track for the next round of scanning.

■ **Example** Describe the organ-pipe distribution and mention why it is a good tool increase the performance of disks.

■ **Answer:** Placing the most used blocks of data close together in order to reduce the seek time, which is the most important delay in disk performance. The organ-pipe distribution places data this way, using a histogram and allowing the most used blocks of data to be in the same track, the next most used blocks in the next tracks, etc.

■ **Example** What is the sequence of software layers that are traversed from the time a user needs to read a disk block until the time the data is available to the user (from library call to the return from the library call). Among the layers are: (a) libraries, (b) page replacement algorithms. (c) ISRs, (d) Device-independent OS software, (e) data placement algorithms, (f) de-framentation software, (g) device drivers, (h) controllers, (i) device itself.

■ **Answer:** libraries → Device-independent OS software → device drivers → controller → device → ISR → device drivers → Device-independent OS software → libraries.

■ **Example** In a virtual memory system, an 8KB page is swapped into main memory. This page is stores on a hard disk with the following parameters:
Average track diameter = 6cm
Speed of rotation = 10,800 RPM
Average seek time = 5ms
Seek time between consecutive tracks = 1ms
Linear density = 100,000 bits/cm
Controller overhead = 2ms

a. How many sectors and tracks are required to store one page of data?
   Data per track = phi * 6 * 100,000 = 235500 bytes
   Number of tracks = 8KB/23500B = 0.035
   Number of sectors = 8KB/512=16
b. How long will it take to search and read this page from disk?
   Time needed= ave seek time + ave rotational delay + transfer time + control overhead
   = 5ms + 0.5*60*1000/10800ms + 0.035*60*1000ms + 2ms = 10ms
c. If clock rate is 500MHz, how many clock cycles are needed to search and read the page?
   500M cycles/sec * 10 ms = 5000000 cycles
d. If the disk in consideration is an Ultra SCSI then how many clock cycles it will take to transfer this information over the I/O channel? (Ultra SCSI supports 20MHz bus and 40MB/sec data rate).
   20M cycles/sec * 8KB/(40MB/sec) = 4000 cycles.

4.8 Protection and Security

Security refers to the policy of authorising accesses. ***Protection*** refers to the actual mechanisms implemented to enforce the specified policy. Security aims to prevent intentional misuses of a system, while protection aims to prevent either accidental or intentional misuses.
A secure system tries to accomplish three goals:

1. ***Data confidentiality:*** secret data remains secret.
2. ***Data integrity:*** unauthorised users should not be able to modify any data without the owner's permission.

3. ***System availability:*** nobody can disturb the system to make it unusable.

There are three components of security:

1. ***Authentication*** determines who the user is.
2. ***Authorization*** determines who is allowed to do what.
3. ***Enforcement*** makes sure that people do only what they are supposed to do.

4.8.1 Authentication

Authentication involves sharing a secret between two parties. One common approach of authentication is the use of ***passwords***. However, there are some problems with the password approach:

First, the system must keep a copy of the secret (password). To prevent a malicious user from gaining the access to this list of passwords, a system needs to use ***encryption***, that is, using a key to transform the data and make it difficult to reverse without the key. For example, the UNIX `/etc/passwd` file stores passwords that are encrypted using one-way transformations. Since, the system only stores the encrypted version, a malicious user cannot read those passwords. When you type in the password, the system first encrypts your password and then compares it to the encrypted version.

The second problem with the password scheme is that it is difficult to come up with good passwords. Short passwords are easy to crack, but people tend to write down long passwords.

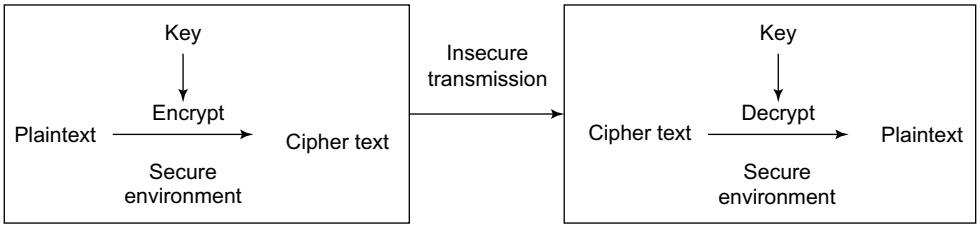
The third problem concerns whether we can trust the encryption algorithm. If there is a backdoor, decryption may not require exhaustive search.

4.8.2 Authentication in Distributed Systems

In the distributed environment, encryption is needed for authentication and guarding the secrecy of data in transit.

Private Key Encryption

The idea of ***private key*** encryption is to use an encryption algorithm that can be easily reversed, given the correct key (and hard to reverse without the key).



Without the key, one cannot decode the cipher text without exhaustive searches. From the plain text and the cipher text, one cannot derive the key. As long as the key stays secret, private key encryption provides both secrecy and authentication.

The tricky part of the private key approach is to distribute keys in the first place. It usually involves an ***authentication server***, which keeps a list of keys and provides a way for two parties to talk to each other, as long as they trust the server.

Suppose Key_{xy} denotes the key for talking between x and y, and Key_{xy}[message] means to encrypt a message with Key_{xy}. Also, suppose we have clients A and B, and the server S. A and B already own Key_{AS} and Key_{BS}, respectively in order to talk to S. If A wants to talk to B, we go through the following steps under the ***Kerberos protocol***.

Client A first asks S: "Yo, ring me B, and I want Key_{AB}"

$A \rightarrow S$: Key_{AS}[give me Key_{AB}]

S gives back Key_{AB} to A, and a message for B signed by Key_{BS}, containing Key_{AB}.

$S \rightarrow A$: Key_{AS}[here is Key_{AB} and a message to B]

A sends the message to B

$A \rightarrow B$: Key_{BS}[use Key_{AB} to talk to A]

There are additional details:

1. The server adds timestamps to limit how long a key can be used; this will prevent a machine from replaying messages later (e.g., "deposit $100").

2. The encrypted message also includes checksums to prevent a malicious user from inserting things into the message (e.g., "deposit $1,000").
3. To reduce the exposure of Key_{AS} (or Key_{BS}), A can periodically ask the server to give a temporary key $Key_{A'S}$ that is different each time to serve the function of Key_{AS}.

In a nutshell, the Kerberos protocols are proposed to have secured transactions between the processes running on diff m/c's which are connected thru a comp N/W. Here, the following assumptions are made,

(i) The network is insecure

(ii) The O.S is not trusted or insecure

(iii) The authentication server is trusted

Here, the Client will be contacting the Authentication Server (A.S) by presenting its credentials along with the server info also. The A.S will generate a ticket having session key encrypted using Client's key and some other field s such as Client ID and session key which are encrypted using Server Key and then sent to the Client. Client extracts the session key from this ticket by decrypting using its key and remembers the session key. Then, the client sends the ticket containing the other fields as it is to the server, the server now applying its key decrypts the message and identifies the session key. Now, both client and server are having exclusively designed keys and they can communicate each other using this session key.

Digital signatures have to be acquired from the authentication servers. Digital signatures are generated by using email ID, time, some string specified by the user by the authentication server.

Public Key Encryption

Public key encryption is an alternative to the private key encryption, which separates authentication from secrecy. Encryption and decryption involves a pair of public keys and private keys. With private key systems, a key is used for both encryption and decryption:

Encryption(Key, plaintext) = cipher text

Decryption(Key, cipher text) = plaintext

With the public key scheme, if the public key is used for encryption, the private key is used for decryption; if the private key is used for encryption, the public key is used for decryption:

Encryption(Key_{public}, text) = cipher text

Decryption($Key_{private}$, cipher text) = plaintext

Encryption($Key_{private}$, text) = cipher text

Decryption(Key_{public}, cipher text) = plaintext

The idea is that the private key is kept secret, while the public key is listed in the directory. So, we can have the following variations of encrypted transmissions:

- Key_{my_public}[Hi, venkat]: anyone can create it, but only I can read it (secrecy).
- $Key_{my\_private}$[I'm venkat]: everyone can read it, but only I can send it (authentication).
- Key_{your_public}[$Key_{my\_private}$[I know your secret]]: only I can send it, and only you can read it.

However, how can you trust public keys?

4.8.3 Authorisation

The ***Access matrix*** is a formalisation of all the permissions in the system, describing who can do what. For example, in the following matrix, it says that Bart can read Lisa's diary.

	File1	Lisa's diary	File3	…
Venkat	read, write	read		
Lalitha		read, write		
Magadh			read	
…				

However, due to the size and the sparse use of the matrix, almost all systems implement two alternatives: the access control list and the capability list.

The ***Access control list*** stores all permissions for all users with each object. An analogy is a guard standing in front of a

door with a list of people who are allowed to enter. However, as the number of users increases, this list may become very long. Under UNIX, the permission of each file is specified according to its owner, user group, and the world (everyone). Therefore, a file may be specified as world readable, group readable and writeable, and owner executable.

The ***Capability list*** stores all objects the process has permission to touch for each process. An analogy is that each person owns a set of keys. Whoever has the key has the right to enter the door. Page tables are an example. Each process has a list of pages it has access to, *not* each page has a list of processes that have access permission.

Access control list allows an object to easily know who is allowed to access the object. However, it is difficult to know which objects a user can access. Therefore, it is more difficult to revoke a user's access rights to a set of objects. The capability list allows a user to easily know the list of objects to access. However, it is difficult to discover the list of people who can access an object. Therefore, it is more difficult to revoke capabilities of an object from a set of users. Most of the operating systems today use an access control list for most resources.

Enforcement

The enforcer checks passwords, access control lists, and so on. Any bug in enforcer means that a malicious user can gain the ability to do almost anything. In UNIX, the superuser has all the powers of the UNIX kernel. Because of the coarse-grained access control, lots of things have to run as superuser in order to work. If there is a bug is in any one of these programs, you are hosed!

To reduce the number of bugs, the enforcer should be as small as possible, which often leads to a simple protection model. Minimal-privilege-based enforcers tend to be complex and more prone to bugs.

State of the World in Security

Authentication in the single-machine environment is mostly password-based, and people still use poor passwords. Authentication in distributed systems mostly depends on encryption, but almost nobody encrypts (e.g., emails).

Authorisation is largely based on the access control list. However, many systems provide only very coarse-grained access control (e.g., UNIX). Therefore, often protection mechanisms are turned off to enable sharing.

Enforcement is mostly achieved through the kernel model. It is hard to write a million lines of code without bugs, and any bug is a potential security loophole.

4.8.4 Classes of Security Problems

Eavesdropping

Eavesdropping is the listener approach. One can tap into the serial line on the Ethernet, and see everything typed in; almost everything goes over network unencrypted. For example, your password goes over the network unencrypted when you `rlogin` to a remote machine.

The military approach to defeat wired eavesdropping is to use pressurised cables. If the air pressure drops, someone may be trying to tap into the cable.

Abuse of Privilege

If the superuser is evil, there is nothing you can do.

Imposter

An imposter breaks into the system by pretending to be someone else. For example, if a system authenticates by a person's voice or facial image, the system can be fooled. A countermeasure against the imposter attack is to use ***behavioral monitoring*** to look for suspicious activates (e.g., overwriting the boot block).

Trojan Horse

A ***Trojan horse*** is a seemingly innocent program that contains code that will perform an unexpected and undesirable function. A countermeasure against the Trojan horse is ***integrity checking***. Periodically, the system should check the content of the disk against the original checksums for various files.

Salami Attack

The idea is to build up a chunk one-bit at a time. A programmer at a bank can reprogram the accounting program, so that the partial pennies go into his account. A countermeasure is for companies to have ***code reviews*** as a standard practice.

Logic Bombs

A programmer may secretly insert a piece of code into the production system. As long as the programmer feeds the system password periodically, it does nothing. However, if the programmer is suddenly fired, the logic bomb does not receive its password, so it goes off. Logic bombs can be also prevented by code reviews.

Denial-of-Service Attack

Denial-of-service attacks refer to attacks on system availability. A handful of compromised machines can flood a victim machine with network packets to disrupt its normal use. Currently, researchers are still looking for effective countermeasures.

Some important points about File System security

- "chmod" command of Unix can be executed on a file by the owner of that file only (exception for the superuser who can change permissions of any file of any directory of any user).
- Chmod command is used and file permissions are changed then they are visible even we logout.
- Main drawback of chmod command is that a user cannot give permissions to a specified User. He can give to a group of people only.
- If we wanted in Unix system to have freedom to give permissions to a specified user then, you have to create groups such that there will be only one member.
- If we have execution permission for a directory then we can "cd" into that directory.
- Sticky bit if set for an executable file then next time when it is running it will be giving better response time.
- If the sticky bit is set for a directory then anyone can create files or subdirectories in it while maintaining their privacy rules.
- If gid bit is set for a file then kernel enforces mandatory locking if required.
- If uid bit is set for a (executable) file which belongs to some other user and when we run that file, we will be acquiring the privileges of that user. Example Passwd command.
- To be specifically, external authentication is required to login into the system whereas internal authentication includes making sure that one user's process does not disturb the privacy rules of other user processes or other user resources.
- In order to protect the systems from external users , some application programs are been made such that if any one tries to login with superuser name as "root" or operator etc. They will not be permitted to login . In secured Telnets also it is not possible.
- In most of the versions of Unix when a user tries to login from a terminal unsuccessfully for 3 consecutive times, we may get a message known as timed out and that same info is recorded in log files. There are some administration tools which when executed, they report about these attempts.
- Some application SWs such as Oracle, when installed it will be creating some user accounts also, special accounts to admin oracle who will be having super user privileges. They may be having well known passwords and usernames. It is mandatory that the Unix administrator to change the passwords of those well known accounts, soon after their installations.
- **Password Aging:** Is small security measure in which administrators will be enforcing that the users has to change their passwords within some stipulated amount of time.
- Great security loop hole in Unix O.S is the super user himself
- In Unix designers during the system booting time by pressing some hot keys or by entering some boot options one can enter into the system in single user mode. If the Unix system is unattended, there is a danger that users can power off the m/c, on it again and login in single user mode and then they can take over the control of the system.

4.9 Introduction to Queuing Theory

- Poisson arrival with λ constant arrival rate (customers per unit time) each arrival is independent.
- $P(\tau \le t) = 1 - e^{-\lambda t}$
- Probability n customers arrive in time interval t is:

  $$e^{-\lambda t}(\lambda t)^n / n!$$

- Assume random service times (also Poisson): μ constant service rate (customers per unit time)
- What is the CPU utilisation (fraction of time that CPU is busy?)
  - Every $1/\lambda$ seconds, one new arrival

- Takes $1/\mu$ seconds to process one job
- Utilisation: $(1/\mu)/(1/\lambda) = \lambda/\mu = \rho$

- On average, how many jobs are in the system?
  - Using a bit of queuing theory: $\rho/(1-\rho)$

Little's Theorem:

- W_q = mean time a customer spends in the queue
- λ = arrival rate
- L_q = number of customers in queue
- W = mean time a customer spends in the system
- L = mumber of customers in the system
- $L_q = \lambda W_q$
- $L = \lambda W$

- Server Utilisation: $\rho = \lambda/\mu$
- Time in System : $W = 1/(\mu-\lambda)$
- Time in Queue: $W_q = r/(\mu-\lambda)$
- Number in Queue (Little): $L_q = \rho^2/(1-\rho)$

■ **Example**

Arrival rate $\lambda = 2$ jobs/sec

Service rate $\mu = 3$ jobs/sec

Utilisation $\rho = \lambda/\mu = 66.66\%$

Time in system $W = 1/(\mu - \lambda) = 1$ sec

Time in queue $W_q = \rho/(\mu - \lambda) = .6666$ sec

Length of queue $L_q = \lambda^* W_q = 1.3333$

■ **Example** An I/O system with a single disk gets about 10 requests/second and the average time for a disk IO to serve is 50 ms. What is utilisation?

■ **Answer:**

Arrival Rate =10

Service Rate = 1/50ms = 20

r = 10/20 = 0.5

Average No of IO requests in the system = Arrival rate * Waiting time

■ **Example** Suppose the average time to satisfy a disk request is 50ms and arrival rate is 200 requests/sec. What is the mean no of I/O requests at the disk server?

■ **Answer:**

Length $_{server}$ = Arrival rate * Time $_{server}$

= 200*0.05 sec=10

■ **Example** Suppose a processor sends 10 disk IO requests per second, these requests are exponentially distributed and average disk service time is 20ms then calculate

a. On average how disk is utilised
b. What is the average time spent in the queue
c. What is 90^{th} percentile of queue time
d. What is the average response time for a disk request, including the queuing time and service time?

■ **Answer:**

Arrival rate = 10

Service rate =1/20ms

Utilisation = 10/(1/0.02) 0.2

$Time_{queue}$ = $Time_{server}$ * Server Utilisation/(1-Server utilisation) = 20ms*0.2/(1 – 0.2) = 5ms
The average response time = $Time_{queue}$ + $Time_{server}$
= 5 + 20 = 25ms
90^{th} percentile of queue time = 11.5ms

■ **Example** There is a restaurant which holds 100 people. The average dinner takes two hours and there are 25 people waiting ahead of you. Calculate the average waiting time. Departure rate is 50 people per hour.
100 people = 50 people/hour * 2 hours
You have to wait approximately 30 minutes

■ **Example** In a city with 1 lakh population an average person lives 70 yrs. Death rate, birth rate is same. That is system is closed system.

■ **Answer:** Death rate * average time in system = number in system
Death rate * 70 = 100000
Death rate = 14000 per year

■ **Example** Suppose we have a single processor system, and jobs arrives at a rate of 10 jobs/second. Each job takes 50 ms. What is the expected number of jobs in the system and average time in system.

■ **Answer:**
The arrival rate = 10
Service rate = 20
Mean service time = 1/(20 – 10) = 1/10 = 100 msec
No of jobs in the system = ρ/(1– ρ) (where ρ is utilisation given as λ/μ = 10/20 = 0.5)
= 0.5/(1– 0.5) = 1
The same can be got from Liitle's theorem
No of jobs in the queue = Mean service Time * arrival rate = 100ms *10 = 1

■ **Example** The mean arrival to a hot bathe service is 0.05 customers/sec and the mean bathe is 0.1 customers/sec. Calculate mean service time.

■ **Answer:**
Mean service time = 1/0.1 – 0.05 = 20 sec

■ **Example** Consider a terminal concentrator with four 4800 bps input lines and one 9600 bps output line. The mean packet size, 1/m is 1000 bits. Each of four line delivers poison traffic with an average of λI = 2 packets/sec. What is the mean delay experienced by a packet from the moment the last bit arrives at the concentrator until the moment that bit is retransmitted to the output line? Also what is the mean number of packets in the concentrator including the one in service?

■ **Answer:**
Arrival rate = 2*4 = 8
Service rate = 9600/1000 = 9.6
Mean delay = 1/(9.6-8) = 625 msec
Mean no of packets in the concentrator = (8/96)//(1–8/9.6) = 5

■ **Example** Two computers are connected by a 64 Kbps line. There are eight parallel services using the line. Each session generates poisson traffic with a mean 0f 2 packets/sec. The packet lengths are exponentially distributed with a mean of 2000 bits. The designers must choose between giving each session a dedicated 8kbps of bandwidth (FDM or TDM) or having all packets compete for a singe 64 kbps shared channel. Which is better?

■ **Answer:**
Analysis (Each service a separate channel)
Arrivals = 2
Service rate=8000/2000 = 4
Mean dealy=1/(4–2) = 500 ms
Analysis(Shared channel)

Arrivals = 2*8
Service rate = 64/2000 = 32
Mean delay = 1/(32 – 16) = 66.7 msec

■ **Example** A communication channel capable of transmitting at a rate of 50 kbps will be used to communicate 10 sessions each generating poison traffic at a rate 150 packets/min. Packets lengths are exponentially distributed with mean 1000 bits.

a. For each session find the average number of packets in queue, the average number of packets in the system, and the average delay per packets when the line is allocated to the session is using 10 equal capacity TDM line.
b. Repeat the above for the case where 5 of the sessions transmit at a rate of 250 packets/min while the other at a rate of 50 packets/min.

■ **Answer:**

$\lambda = 5/60 = 1/12$ sec

$N = 5$ minutes

$T = 20 + 5$ minutes

4.10 Solved Questions

1. The following is a set of three interacting processes (P1, P2, P3) that can access two shared semaphores:

```
semaphore U = 3;
semaphore V = 0;
```

```
[P1]            [P2]            [P3]
while(TRUE)     while(TRUE)     while(TRUE)
begin           begin           begin
 wait(U)         wait(V)         wait(V)
 print("C")      print("A")      print("D")
 signal(V)       print("B")    end
end              signal(V)
                end
```

Within each process the statements are executed sequentially, but statements from different processes can be interleaved in any order that's consistent with the constraints imposed by the semaphores. When answering the questions below assume that once execution begins, the processes will be allowed to run until all 3 processes are stuck in a wait() statement, at which point execution is halted.

a. Assuming execution is eventually halted, how many C's are printed when the set of processes runs?

Answer: Exactly 3. Each time P1 executes the "wait(U)" statement, the value of semaphore U is decremented by 1. Since there are no "signal(U)" statements, the loop in P1 will execute only 3 times (i.e., the initial value of U) and then stall the fourth time "wait(U)" is executed.

b. Assuming execution is eventually halted, how many D's are printed when this set of processes runs?

Answer: Exactly 3. P1 will execute its loop three times (see the answer to the previous question), incrementing "signal(V)" each time through the loop. This will permit "wait(V)" to complete three times. For every "wait(V)" P2 executes, it also executes a "signal(V)" so there is no net change in the value of semaphore V caused by P2. P3 does decrement the value of semaphore V, typing out "D" each time it does so. So P3 will eventually loop as many times as P1.

c. What is the smallest number of A's that might be printed when this set of processes runs?

Answer: 0. If P3 is scheduled immediately after P1 executes "signal(V)", then P2 might continue being stalled at its "wait(V)" statement and hence never execute its print statements.

d. Is CABABDDCABCABD a possible output sequence when this set of processes runs?

Answer: No. Here are the events implied by the sequence above:

start: U=3 V=0
print C: U=2 V=1
print A: U=2 V=0
print B: U=2 V=1
print A: U=2 V=0
print B: U=2 V=1
print D: U=2 V=0
print D: oops, impossible since V=0

e. Is CABACDBCABDD a possible output sequence when this set of processes runs?

Answer: Yes:

start: U=3 V=0
print C: U=2 V=1
print A: U=2 V=0
print B: U=2 V=1
print A: U=2 V=0
print C: U=1 V=1
print D: U=1 V=0
print B: U=1 V=1
print C: U=0 V=2
print A: U=0 V=1
print B: U=0 V=2
print D: U=0 V=1
print D: U=0 V=0

f. Is it possible for execution to be halted with either U or V having a non-zero value?

Answer: No. If U has a non-zero value, P1 will be able to run. If V has a non-zero value, P3 will be able to run.

2. In a system, memory pages(frames) are 512 words long. Consider the following Pascal program segement:

```
Var A :array [1..128] of array [1..128] of integer
for j := 1 to 128
do for i :=1 to 128
do A[i][j]=0;
```

Assuming that 4 memory words are needed to store an integer and that arrays are stored in row-major order internally, how many page faults would be generated by the above program segment.

Modify the above program such that the number of page faults is reduced. Assume FIFO policy.

Answer: As array size is 128 elements with each one element 4 bytes, we need 512 bytes for one row.

However, the above program access the elements A[0][0], A[1][0],….A[127][0], A[0][1], A[1][1], ..A[127][1], etc,. That is, 0^{th} column elements are accessed first then 1^{st} column elements and vice versa. When j = 0 and i = 0, we need A[0][0], however whole 0th row is read. However, after using A[0][0] next page (next row) will be loaded without using elements of the current row elements that are available already in the memory. If we assume the number of frames allocated for this program is limited (say one) we get page fault for each element reference. By exchanging the for loops we can have better program which accesses all the elements of row once it is loaded into page.

```
Var A :array [1..128] of array [1..128] of integer
 for i := 1 to 128
 do for j :=1 to 128
 do A[i][j] = 0;
```

If we observe the above modified code, we will be accessing A[0][0], A[0][1], A[0][2],…. A[0][127], A[1][0], A[1][1]…..A[1][127], etc. Thus, when we load first row while referring A[0][0], along with A[0][0] all other elements will be loaded into memory and are used. Thus, page faults will be dramatically reduced.

3. A 32 bit system has a page size of 8K bytes. If a two-level paging scheme is used, where each page of the page table is 4K bytes, calculate the size of the outer page table.

Answer:

Page size = 8K =13 bits is offset

Page of the page table is 4K. Therefore, 12 bits are for second level page table.

Total address bits = 32 bits

Therefore, no of bits for the first level page table = 32–12 –13 = 7

No of entries in the outer page table = $2^7 = 128$

4. What is a stack algorithm?

Answer: A stack algorithm is a page replacement algorithm for which the pages in memory for n frames is always a subset of the set of pages for n+1 frames. Such algorithms do not exhibit Belady's Anomaly, and the optimal and LRU algorithms are examples.

5. What is hint? Explain.

Answer:

The idea of having the processor use the cached data before the tag match completes can be applied to caches. A subset of the tag, called a hint, can be used to pick just one of the possible cache entries mapping to the requested address. This datum can then be used in parallel with checking the full tag. The hint technique works best when used in the context of address translation.

6. What is victim cache?

Answer:

A victim cache is a cache used to hold blocks evicted from a CPU cache due to a conflict or capacity miss. The victim cache lies between the main cache and its refill path, and only holds blocks that were evicted from that cache on a miss. This technique is used to reduce the penalty incurred by a cache on a miss.

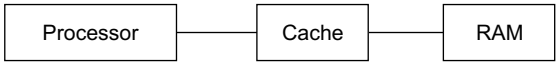
The original victim cache on the HP PA7200 was a small, fully-associative cache. Later processors, such as the AMD K7 and K8, used the very large secondary cache as a victim cache, to avoid duplicate storage of the contents of the large primary cache.

7. What is a trace cache?.

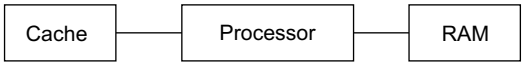
Answer: A trace cache is a mechanism for increasing the instruction fetch bandwidth and decreasing power consumption (in the case of the Pentium 4) by storing traces of instructions that have already been fetched and decoded.

8. What are front-side and back-side caches?

Answer: Front side cache means it will not be having any special bus. Where as, back-side cache will be having a separate bus which makes it to operate at the speeds of the core.



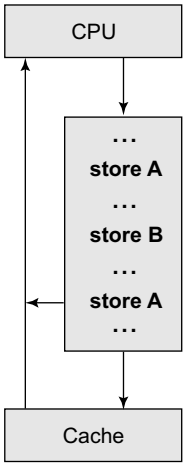
Front- Side Cache



Back-side Cache

9. What is the use of Write buffer?

Answer: Store operations in to cache may take some time and there is a danger of CPU to get stalled. To reduce this, write buffer is used. Write buffer is FIFO queue of incomplete write and is also called as store buffer or write-behind buffer. If write buffer contains a data item A, then it also indicated that its counterart is memory is stale.



10. What is meant by inclusive or exclusive caches in multilevel caches?

Answer: Exclusive means a particular cache line may be present in exactly one of the cache levels and no more than one. Inclusive means the line may be present simultaneously in more than one level of cache. Nothing prevents the line widths from being different in differing cache levels.

11. What cache line coloring?

Answer: Every cache-able address has one and only one corresponding index line, which can cause problems. For instance, if the processor reads a sequence of addresses that accidentally happen to correspond to the same cache index, the cache line must be evicted and re-fetched on each read. Such a situation easily can happen in, say, a for loop reading elements of a structure that happens to be about the same size as the cache. For directly mapped caches, the index sometimes is called the cache colour, and this problem is called the cache-line colouring problem.

12. What is the use of ASID or RID?

Or

What is line doubling?

Answer: Virtual tagging introduces a serious hazard known as line doubling. If two tasks share a single physical line which appears under different addresses in their virtual memory spaces, and one task modifies its copy, the other task may not be aware of it and continue using an obsolete copy. The easiest way to avoid this issue is to flush cache on every task switch, though it isn't much attractive in means of performance. Another way is to supply every line with an auxiliary field known as ASID (Address Space IDentifier) or RID (Region IDentifier).

13. A file system checker can also be used to check files and directories. Suppose that the following table is constructed:

I-node number:	0 1 2 3 4 5 6 7 8 9
File count:	1 0 1 1 0 1 1 0 1 0
I-node count:	1 0 0 1 0 2 1 0 1 0

The file count is obtained by traversing from the root directory and computing the number of times an i-node is used by a file. The i-node count is just the link counts stored in each i-node. Are there any errors? If so, how serious are they, and how can they be fixed?

Answer: I-node 2 has an error. The file count is larger than the I-node count. This means a link to the file is not counted. This could be serious. If any file is erased, the file is lost. Set the i-node count to 1 to fix it.

I-node 5 has an error. The I-node count is larger than the file count. That means I-node points to some file that no longer exists. This is not serious. Set the i-node count to 1 to fix it.

14. Consider the following program. Explain what it does?

```
1   main (int argc, char * arcv []) {
2     int pid;
3
```

```
4     pid = fork () ;
5
6     if (pid< 0) {
7       fprintf (stderr, "Exit") ;
8       exit (-1) ;
9     }
10
11    else if (pid==0) {
12      execlp ("/bin/ls", "ls", NULL);
13    }
14
15    else {
16     wait (NULL);
17     printf ("Done");
18     exit (0);
19    }
20  }
```

Answer: It creates a child process. In the child process, execlp() is used to load "ls" program. While child is running "ls" command, parent process will be waiting. After child's completion, parent displays "Done".

15. Which of the following instructions should be privileged?

a. Set value of timer
b. Read the clock
c. Clear memory
d. Issue a trap instruction
e. Turn off interrupts
f. Modify entries in device-status table
g. Switch from user to kernel mode
h. Access I/O device

Answer: The following operations need to be privileged: Set value of timer, clear memory, turn off interrupts, modify entries in device-status table, access I/O device. The rest can be performed in user mode.

16. When a process creates a new process using the fork() operation, which of the following state is shared between the parent process and the child process?

a. Stack
b. Heap
c. Shared memory segments

Answer: Only the shared memory segments are shared between the parent process and the newly forked child process. Copies of the stack and the heap are made for the newly created process.

17. Researchers have suggested that, instead of having an access list associated with each file (specifying which users can access the file, and how), we should have a user control list associated with each user (specifying which files a user can access, and how). Discuss the relative merits of these two schemes.

Answer:

- File control list. Since the access control information is concentrated in one single place, it is easier to change access control information and this requires less space.
- User control list. This requires less overhead when opening a file.

18. Consider the following three threads and three semaphores:

/* Initialize semaphores */

s1 = 1; s2 = 0; s3 = 0; x = 0;

void thread1(){
x = x + 1;
}
	void thread2(){
	x = x + 2;
	}
		void thread3(){
		x = x * 2;
		}

Add P(), V() semaphore operations (using semaphores s1, s2, s3) in the code for thread 1, 2 and 3 such that the concurrent execution of the three threads can only result in the value of x = 6.

Answer:

void thread1(){
P(s1)
x = x + 1;
V(s2)
}
	void thread2(){
	P(s2)
	x = x + 2;
	V(s3)
	}
		void thread3(){
		P(s3)
		x = x * 2;
		V(s1);
		}

If second and third threads starts first, they gets blocked. When first thread starts, s1 value becomes 0 after P operation and x value will be initialized to 1. Then it signals to thread2 which is waiting on s2. Thread2 changes x value to 3 then signals thread3, which in turn double x value, thus x value becomes 6.

19. This question will test your understanding of concurrent programming, specifically deadlocks. For these questions, assume each function is executed by a unique thread on a uni-processor system.

1 void thread1() { 2 P(lock1); 3 P(lock2); 4 P(lock3); 6 /* do some work */ 7 V(lock2); 8 V(lock3); 9 V(lock1); 10 }	11 void thread2() { 12 P(lock1); 13 P(lock2); 14 P(lock3); 15 /* do some work */ 16 V(lock1); 17 V(lock2); 18 V(lock3); 19 }

Does this code contain a deadlock? If so, write instruction sequence (through their line numbers) leads to deadlock.

20. What will be the effect of the following program?

```
main(){
for(;;) fork();
}
```

Answer: It creates infinite number of processes. It exhausts available swap memory and one may get error message such as "unable to swap". We may get another error such as "unable to fork". Usually, there will be system resource limit known as "number of processes which a process can create through fork". By repeated calling of fork(), we might be exceeding this limit, thus we may get error such as "unable to fork".

21. What is the difference between deadlock and starvation?

	Deadlock	Stravation
reason	Usually due to the program design. It is difficult for OS to detect and prevent it.	Usually due to the scheduling algorithm of OS.
appearance	Two or more processes are waiting for each other and formed a circle.	Can happen to a single process.
solve	Can not be solved naturally, even if no other processes are running.	If no other process is running, it can process.

22. List general methods used to pass parameters to the OS.

- Pass parameters in registers
- Registers pass starting addresses of blocks of parameters
- Parameters can be placed, or pushed, onto the stack by the program and popped off the stack by the operating system

23. If a thread causes an unrecoverable error such as stack overflow, memory access violation, divide-by-zero, and etc, what would probably happen to the process containing the thread? Will other threads within the same process be affected?

Answer: The process containing the faulty thread would likely exit abnormally. As a result, all threads within the process will exit.

24. The following pair of processes share a common variable X:

P1	P2
int Y;	int Z;
A1: Y = X*2;	B1: Z = X+1;
A2: X = Y;	B2: X = Z;

X is set to 5 before either process begins execution. As usual, statements within a process are executed sequentially, but statements in P1 may execute in any order with respect to statements in P2.

a. How many different values of X are possible after both processes finish executing?

Answer: There are four possible values for X. Here are the possible ways in which statements from P1 and P2 can be interleaved.

A1 A2 B1 B2: X = 11

A1 B1 A2 B2: X = 6

A1 B1 B2 A2: X = 10

B1 A1 B2 A2: X = 10

B1 A1 A2 B2: X = 6

B1 B2 A1 A2: X = 12

b. Suppose the programs are modified as follows to use a shared binary semaphore S:

P1	P2
int Y;	int Z;
wait(S);	wait(S);
A1: Y = X*2;	B1: Z = X+1;
A2: X = Y;	B2: X = Z;
signal(S);	signal(S);

S is set to 1 before either process begins their execution and. As before, initial value of X is 5. Now, how many different values of X are possible after both processes finish executing?

Answer: The semaphore S ensures that, once begun, the statements from either process execute without interrupts. So now the possible ways in which statements from P1 and P2 can be interleaved are:

A1 A2 B1 B2: X = 11

B1 B2 A1 A2: X = 12

c. Finally, suppose the programs are modified as follows to use a shared binary semaphore T:

P1	P2
int Y;	int Z;
A1: Y = X*2;	B1: wait(T);
A2: X = Y;	B2: Z = X+1;
signal(T);	X = Z;

T is set to 0 before either process begins their execution. As before, X is set to 5.

Now, how many different values of X are possible after both processes finish executing?

Answer: The semaphore T ensures that all the statements from P1 finish execution before P2 begins. So

now there is only one way in which statements from P1 and P2 can be interleaved:

A1 A2 B1 B2: X = 11

25. The following pair of processes share a common set of variables: "counter", "tempA" and "tempB":

Process A	**Process B**
...	...
A1: tempA = counter + 1;	B1: tempB = counter + 2;
A2: counter = tempA;	B2: counter = tempB;
...	...

The variable "counter" initially has the value 10 before either process begins to execute.

a. What different values of "counter" are possible when both processes have finished executing? Give an order of execution of statements from processes A and B that would yield each of the values you give. For example, execution order A1, A2, B1, B2 would yield the value 13.

Answer: There are three possible values for X. Here are the possible ways in which statements from A and B can be interleaved.

A1 A2 B1 B2: X = 13

A1 B1 A2 B2: X = 12

A1 B1 B2 A2: X = 11

B1 A1 B2 A2: X = 11

B1 A1 A2 B2: X = 12

B1 B2 A1 A2: X = 13

b. Modify the above programs for processes A and B by adding appropriate signal and wait operations on the binary semaphore "sync" such that the only possible final value of "counter" is 13. Indicate what should be the initial value of the semaphore "sync".

Answer: We need to ensure that A and B run uniterrupted, but it doesn't matter which runs first.

semaphore sync = 1;

Process A	**Process B**
wait(sync);	wait(sync);
A1: tempA = counter + 1;	B1: tempB = counter + 2;
A2: counter = tempA;	B2: counter = tempB;
signal(sync);	signal(sync);

26. Explain how a program address space is initiated?

To start a program, the OS must

- allocates swap space: space on the backing store that can hold images of the pages in memory
- allocates a page table for all of the pages a program might use. This table must be contiguous since the hardware indexes into it.
- allocates a disk block descriptor table to hold the backing store information
- copies each page of the program into the swap space noting the disk block and swap device number of the page in the appropriate disk block descriptor table entry
- sets all of the valid bits in the page table to be zero
- runs the program

27. Explain about first fetch instruction of a program.

Answer:

The very first fetch of an instruction will cause a page fault to occur since it will be attempting to read page 0. Since the valid bit is clear, the OS will take a page fault exception immediately, go to the disk block descriptor entry for page 0, find the disk block number, get a free frame, load the page into the frame, update the frame table entry with the pointer to the page table entry, update the page table entry with the frame number, set the valid bit to 1, and restart the faulting instruction. When the programs runs onto page 1, or jump to another page, it will be faulted in accordingly.

28. Which pages are pinned?

Answer: Kernel pages.

29. What is the difference between logical addresses and pointers?

Answer: The difference between logical addresses and pointers is that all pointers are user objects, and thus pointers only point from one place in logical memory to another place in logical memory. The mapping from logical to physical is only visible to the designer of the system.

30. What is secondary page fault?

Answer: In multi level page tables, if the required second level page block is not available in the physical memory, system raises as secondary page fault.

31. What are the different means of implementing page table?

Answer:

- Fast registers if the page table size is small
- memory
- content addressable memories (caches)

32. Why must the TLB be flushed on a context switch?

Since the TLB translates virtual addresses to physical addresses and processes don't share physical memory in general, the virtual address translations in the TLB will not be valid for the new process.

33. In which of the following C language's for loops, the miss rate is more? Why?

```
Loop 1
int I, J, a[1024] [1024]
for (I = 0; I < 1024; ++I) {
   for (J = 0; J < 1024; ++J) {
      a[I] [J] = 0;
   }
}
Loop 2
int I, J, a[1024] [1024]
for (I = 0; I < 1024; ++I) {
   for (J = 0; J < 1024; ++J) {
      a[J] [I] = 0;
   }
}
```

Answer: In the second one, miss rate is more. Do read about row and column major in data structures unit.

34. Is it possible for a program to specify in which physical address a instruction has to occupy while it is running?

Answer: No.

35. What is the difference between buffer and Cache?

In computing, a buffer is a region of memory used to temporarily hold output or input data, comparable to buffers in telecommunication. The data can be output to or input from devices outside the computer or processes within a computer. Buffers can be implemented in either hardware or software, but the vast majority of buffers are implemented in software. Buffers are used when there is a difference between the rate at which data is received and the rate at which it can be processed, or in the case that these rates are variable, for example in a printer spooler.

The difference between buffers and cache:

Buffers are allocated by various processes to use as input queues, etc. Most of the time, buffers are some processes' output, and they are file buffers. A simplistic explanation of buffers is that they allow processes to temporarily store input in memory until the process can deal with it.

Cache is typically frequently requested disk I/O. If multiple processes are accessing the same files, much of those files will be cached to improve performance (RAM being so much faster than hard drives), it's disk cache.

36. Consider a system that has many threads of two difference types (typeA and typeB). All the threads call the below subroutine named routine passing their type as an argument.

```
enum { typeA, typeB} ThreadType;
Semaphore semaX = 0; // Initial value of semaX
is zero.
Semaphore semaY = 0; // Initial value of semaY
is zero.
void routine(ThreadType threadType){
if (threadType == typeA) {
P(semaX);
V(semaY);
}
if (threadType == typeB) {
V(semaX);
V(semaX);
P(semaY);
}
DoIt();
}
```

Do comment about how many times DoIt() will be executed by typeA threads compared to typeB threads.

Answer: Because each typeA thread waits on X once while each typeB thread signals X twice, there can be up to twice as many typeA as typeB threads passing through DoIt() at the same time. However, there cannot be more typeA threads than this quantity, because the number of A threads is limited by the number of signals to X (two) that each typeB thread makes.

Likewise, because each typeB thread waits on Y once while each typeA thread signals Y once, there can be up to as many typeB as A threads passing through DoIt() at the same time. There cannot be more typeB threads than this quantity because the number of typeB threads is limited by the number of signals to Y (one) that each typeA thread makes.

The range of valid ratios of typeA threads to typeB threads passing through DoIt() at the same time can therefore be expressed as:

$b <= a <= 2b$

Where a is the number of typeA threads and b is the number of typeB threads.

37. A computer has three commonly used resources designated A, B and C. Up to three processes designated X, Y and Z run on the computer and each makes periodic use of two of the three resources.

- Process X acquires A, then B, uses both and then releases both.
- Process Y acquires B, then C, uses both and then releases both.
- Process Z acquires C, then A, uses both and then releases both.

Describe a scenario in which deadlock occurs if all three processes are running simultaneously on the machine. How to avoid it?

Answer: All three processes make their first acquisition then hang waiting for a resource that will never become available. We can avoid this re-ordering the resource requests of process Z as A and C; for that matter we can re-order requests of other processes also.

38. What is a race condition?

Answer: A race condition is a situation where two or more processes are reading or writing some shared data and the final result depends on which process finishes last.

39. What is the priority inversion problem?

Answer: A priority inversion problem is a situation in which a low-priority process is blocking a high-priority process.

40. Consider the following section of code:

```
main() {
 int i = 3;
  int pid;
  while(i > 0) {
   if ((pid = fork()) > 0) {
    printf("In parent %d.\n", i);
    exit(0);
   } else {
    printf("In child %d.\n", i);
    i--;
  }
  }
 }
```

Assume that the fork() system call is successful.

How many processes will be created when the code is executed?

What will be printed?

Answer: 4 processes (one parent plus 3 child processes) will be created.

41. A file system checker has constructed the counters shown below:

Block number:	0 1 2 3 4 5 6 7 8 9
In use:	1 0 0 2 1 0 1 0 1 1
Free:	0 1 1 0 1 1 0 0 0 0

Are there any errors? If so, how serious are they (can they be fixed)? Explain.

Answer: The block number 3 has an error. The block has been used by 2 files. This error might not be able to fixed and need to be reported. The file system checker can copy the content of this block into a free block list such as the block 1. The file system states are then in consistency but the file problems might not solved. So this error needs to be reported.

The block number 4 has an error. It is both in use and free. The error can be fixed. The file system checker just set the free bit to 0. This block becomes a block in use.

The block number 7 has an error. It is neither in use nor free. The error can be fixed. The file system checker just set the in use bit to 0 and add it into the free list.

42. The aging algorithm with a = 1/2 is being used to predict run times. The previous for runs, from oldest to most recent, are 40, 20, 40, and 15 msec. What is the prediction of the next time? (5 points)

Answer: If take all four previous run times into consideration, the prediction

is =(((40 + 20) / 2 + 40) / 2 + 15) / 2 = ((30 + 40) / 2 + 15) /2

= (35 + 15) / 2 = 25

If we take only two previous run times into consideration, the prediction= (40 + 15)/2= 27.5

43. How many processes will be created when the following program is executed?

Assume that all fork system calls are successful.

```
main(){
int i, pid;
for (i=1; i<=3; i++)
pid = fork();
}
```

Answer: There are 8 processes (1 parent and 7 child processes) created when this program is executed.

44. P is a set of processes. R is a set of resources. E is a set of request or assignment edges. These sets P, R, and E are as follows:

P = {P1, P2, P3}, R = {R1,R2,R3},

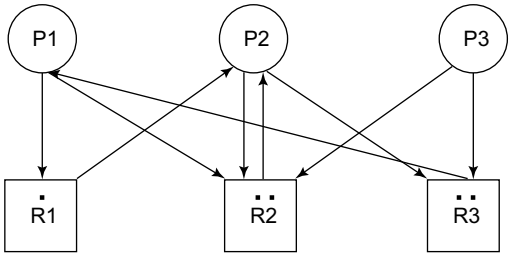
E = {P1→R1, P1→R2, P2→R2, P2→R3, P3 → R2, P3→R3, R1 →P2, R2 →P2, R3 → P1}.

R1 has one instance. R2 has two instances. R3 has one instance.

(a) Draw the resource-allocation graph.

(b) Is there any deadlock in this situation? Briefly Explain.

Answer: Resource allocation graph is given as:



Consider the resource-allocation graph. There are four cycles in the system:

P1 → R1 → P2 →!R3 → P1, P1 → R2 → P2 → R3 → P1, P2 → R2 → P2, P2 → R3 → P1 → R1 → P2.

P1 cannot finish because P1 needs R1 and R2 but can only acquire R2 while R1 is held by P2.

P2 cannot finish because P2 needs R2 and R3 but can only acquire R2 while R3 is held by P1.

P3 cannot finish because P3 needs R1 and R3 but can only acquire R2 while R3 is held by P1.

P1, P2, and P3 cannot progress. The deadlock occurs.

45. Consider a system has p processes. Each process need a maximum of m resources and a total of r resources available. What condition must hold to make the system deadlock free?

Answer: If a process has m resources it can finish and cannot be involved in a deadlock. Therefore, the worst case is where every process has m – 1 resources and needs another one. If there is one resource left over, one process can finish and release all its resources, letting the rest finish too. Therefore, the condition for avoiding deadlock is r >= p(m-1)+1.

46. Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock-free.

Answer: Suppose the system is in deadlock situation. This implies that each of the three processes is holding one resource and is waiting for another resource which is held by one of the other two processes. Since there are three processes and four resources, one process must be able to obtain two resources. This process requires no more resources; therefore it will eventually terminate and returns its two resources back which can be used by other two processes to execute and terminate.

47. What are the contents of an entry in the System-wide-open-file table (I-node table), and an entry in the per-process-open-file table (fd table)?

Answer: Contents of an entry in the system-wide-open-file table:

- Copy of the FCB of each open file.
- Number of processes that have opened the file.

Contents of an entry in the per-process-open-file table (fd table):

- Pointer to the appropriate entry in the system-wide-open-file table.
- Pointer to the current location in the file.
- Access mode with which the file is opened.

48. A CPU scheduling algorithm determines an order for the execution of its scheduled processes. Given n processes to be scheduled on one processor, how many possible different schedules are there? Give a formula in terms of n.

Answer: n!

49. Name items in the Process Control Block (PCB).

(1) Process state
(2) Process id
(3) Program counter
(4) CPU registers
(5) CPU scheduling information
(6) Memory-management information
(7) Accounting information
(8) I/O status information

50. Briefly explain the steps that the operating system takes in order to perform the fork() system call including the blocking and signaling the processes. What would be the value of PID in: PID = fork() after the fork is performed?

Answer:

(i) When a parent process invokes the "fork()" system call, the operating system performs a context-switch and dispatches proper operating system processes to take care of the child process creation.

(ii) The parent process is blocked and is put at the end of the ready queue to be scheduled later;

(iii) The whole contents of the parent's PCB is copied into an empty entry (slot) of the PCB table, the new PCB will be the PCB of the newly created child process and the index number of the new PCB would become the child's PID;

(iv) The child's PCB is put at the end of the ready queue to be scheduled later;

(v) After being dispatched, both parent and child processes with resume their executions from the fork() system call; the parent process will receive the PID of its child and the child will receive 0 as the PID which is an indication to this process that it is the child process;

(vi) The parent process will then block itself by executing "wait()" system call and will wait in a queue until all its child (children) processes are terminated, at which time the parent process is signaled to be waken up and synchronise itself with its child (children).

51. Why do cycles are avoided in a directory structure, and how can we guarantee that no cycles are generated in the directory structure?

Answer: Allowing cycles in a directory structure causes revisiting the same directories several times (may even cause infinite-loop) during traversing a directory structure to find a specific file or directory. The followings are sample solutions to guarantee no cycles:

- Allow only links to files not subdirectories
- Every time a new link is added use a cycle detection algorithm to determine whether a cycle has been created or not.

52. Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation one for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also, assume that the context switching overhead is 0.1 millisecond and that all processes are long running tasks. What is the CPU utilisation for a Round-Robin scheduler when:

(a) The time quantum is 1 msec.

(b) The time quantum is 10 msec.

Answer:

(a) The scheduler incurs a 0.1 msec. of context-switching cost for every context switch. There is 1 unit of CPU time followed by 0.1 units of context switch for both CPU and I/O bound jobs. So CPU utilisation =1.0/(1.0+0.1) = 1/1.1=0.91 or 91%.

(b) This is slightly complicated because I/O jobs take only 1 msec of CPU time while CPU uses 10 msec in each turn. Since time quantum is 10 units, after CPU finishes 10 msec of CPU and is context switched there will be exactly one I/O job waiting in the ready queue. However, it would only use 1 msec. After this CPU-bound job gets the CPU again. This way, on the long run, for every one turn of CPU-bound job, I/O bound job gets a turn. So for every 11 units of CPU usage, 0.2 units are overhead. CPU utilisation=11/11.2 = 0.982 or 98.2%.

53. A CPU scheduler updates priorities of the processes at regular intervals. The updating is a linear function of recent CPU usage of a processes. It is observed that two processes with their recent CPU usages as 40,18 will be getting their priorities as 80 and 69. What is the scheduler's priority updating policy?

Answer: Let P = priority, x=recent CPU usage. Then, from the given details of two processes, we have

$80 = m*40 + c$

$69 = m*18 + c$

By solving these equations, we may find m as 0.5 and c as 60. Thus, scheduler uses

$P = 0.5x + 60$ as the priority updating function.

54. What are protection fault, page fault, segment fault?

Answer: A protection fault occurs when an operation (usually read, write, or execute) is attempted in a segment that is flagged as prohibiting the particular operation. A segment fault occurs when the requested segment is not present in memory. A page fault occurs when the requested page is not in memory.

55. Suppose each process spends 40% of its time in an I/O state. How many such processes are needed to bring the system CPU utilisation to higher than 95%?

Answer: CPU utilisation is defined as $1 - p^n$, where $p = 0.4$ and n is the number of processes. We have the following table:

n	p^n	CPU Utilisation ($1 - p^n$)
1	0.4	0.6
2	0.16	0.84
3	0.064	0.936
4	0.0256	0.9744

Therefore, four processes are required to have CPU utilisation higher than 95%.

56. Describe what happens when a page fault occurs (assume a system based on a paged virtual memory).

- An interrupt is generated by the memory manager (hardware).
- The current state of things must be saved (all registers need to be saved, etc).
- OS (kernel code) is invoked from interrupt service routine.
- Kernel determines the virtual page that caused the fault, makes sure the page is valid (or generates SEGV in offending process, etc).
- If no frames are free, page replacement algorithm is invoked to boot a page from physical memory (may need to be written to backing store).
- Virtual page is copied from backing store to physical memory and page table for relevant process is updated. While this is happening other processes can run. Once I/O is complete, the process that generated the fault can continue – the instruction that generated the fault is restarted.

57. A program is interrupted. However, the OS is taken over the interrupt handling. When such a situation arises?.

1. HW timer is off
2. The program calls a System Call

3. Other HW triggers an interrupt (disk, keyboard, etc.)
4. Program explicitly yields to CPU via a system call

58. Consider a paging system with the page table stored in memory. If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

1. For flat page tables
2. For two level page tables
3. For three level page tables

Answer: For flat page tables it is 400 ns as first we have to check page table which is in memory and then we access the instruction from memory.

For two level is 600 ns(200ns for first level table accesses + 200 ns for second level + 200ns for accessing instruction).

For three level is 800 ns.

59. In a demandpaged memory system, page table is held in registers. It takes 8 milliseconds to service a page fault if an empty frame is available or if the replaced page is not modified and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70% of the time. What is the maximum acceptable page fault rate for an effective access time of no more than 200 nanoseconds?

Answer: Average time to service a page fault: 0.7 * 20 ms + 0.3 * 8 ms = 16.4 ms

Let p be the page fault rate. Therefore, effective access time:

(1 – p) * 100 ns + p * 16.4 ms <= 200 ns

p * (16.4 ms – 100 ns) <= 100 ns

p <= (100 ns) / (16.4 ms –100 ns)

$p <= 6.1 * 10^{9}$

60. Which parameters influences effective memory access time in a flat page table based virtual memory system?

PT = probability of a TLB miss

PP = probability of a page fault, given a TLB miss occurs

TT = time to access TLB

TM = time to access memory

TD = time to transfer a page to/from disk

PD = probability page is dirty when replaced

61. What are some of the differences between a processor running in *privileged mode* (also called *kernel mode*) and *user mode*? Why are the two modes needed?

In user-mode:

- CPU control registers are inaccessible.
- CPU management instructions are inaccessible.
- Parts of the address space (containing kernel code and data) are inaccessible.
- Some device memory and registers (or ports) are inaccessible.

The two modes of operation are required to ensure that applications (running in user-mode) cannot bypass, circumvent, or take control of the operating system.

62. Assuming a page size of 4KB and that a page table entry takes 4 bytes, how many levels of page tables would be required to map a 64 bit address space, if the top level page table fits in a single page? Do calculate number of physical pages that are required to hold the entire page table.

Answer: Since each page table entry is 4 bytes and each page contains 4K bytes, then a one-page page table(first level) would point to $1024 = 2^{10}$ pages, addressing a total of $2^{10} * 2^{12} = 2^{22}$ bytes. The address space however is 2^{64} bytes. Adding a second layer of page tables, the top page table would point to 2^{10} page tables, addressing a total of 2^{32} bytes. Continuing this process,

Depth	Address Space
1	2^{22} bytes
2	2^{32} bytes
3	2^{42} bytes
4	2^{52} bytes
5	2^{62} bytes
6	2^{72} bytes (>= 2^{64} bytes)

We can see that 5 levels do not address the full 64 bit address space, so a 6th level is required. But only 2 bits of the 6th level are required, not the entire 10 bits. So instead of requiring virtual addresses of 72 bits long, we can mask out and ignore all but the 2 lowest order bits of the 6th level. This gives a 64 bit address. Top level page table then would have only 4 entries.

Number of physical pages that are required to hold the entire page table:

Depth	Physical 4Kb Pages Required
1	1
2	$(1*2^{10})+1 > 2^{10}$
3	$((1*2^{10})+1)*2^{10}+1 > 2^{20}$
4	at least 2^{30}
5	at least 2^{40}
6	at least 2^{40}

and the required size of memory would be:

(2^{40} pages) * (4 Kb / page) = 2^{42} Kb = 2^{32} Mb

63. Describe a plausible sequence of activities that occur when a timer interrupt results in a context switch.

- The interrupt generates an interrupt (exception) which transfers control to the kernel-mode handler.
- The handler switches to the kernel stack base for the current process.
- It saves the user-level state (registers, sp, ip) on the stack.
- It determines a timer interrupt occurred and calls the timer interrupt handler.
- The handler acknowledges the interrupt.
- It calls the dispatcher (scheduler).
- The scheduler chooses a new process to run.
- The current in-kernel context is saved on the kernel stack, the sp stored in the PCB (or TCB).
- The new process's sp is loaded; the new kernel stack base is substituted for the old processes' stack base.
- The new processes' kernel context is restored.
- It returns to the assembly routine to restore the user-level state.
- The user-state of the new process (as opposed to the old) is restored.

64. The Unix inode structure contains a reference count. What is the reference count for? Why can't we just remove the inode without checking the reference count when a file is deleted?

Answer: Inodes contain a reference count due to hard links. The reference count is equal to the number of directory entries that reference the inode. For hard-linked files, multiple directory entries reference a single inode. The inode must not be removed until no directory entries are left (ie, the reference count is 0) to ensure that the filesystem remains consistent.

65. Inode-based filesystems typically use block groups. Each block group consists of a number of contiguous physical disk blocks. Inodes for a given block group are stored in the same physical location as the block groups. What are the advantages of this scheme? Are they any disadvantages?

Answer: Block groups keep the inodes physically closer to the files they refer to than they would be (on average) on a system without block groups. Since accessing and updating files also involves accessing or updating its inode, having the inode and the file's block close together reduces disk seek time, and thus improves performance. The OS must take care that all blocks remain within the block group of their inode.

66. Assume an inode with 10 direct blocks, as well as single, double and triple indirect block pointers. Taking into account creation and accounting of the indirect blocks themselves, what is the largest possible number of block reads and writes in order to:

a. Read 1 byte

b. Write 1 byte

Assume the inode is cached in memory.

Answer: To write 1 byte, in the worst case:

- 4 writes: create single indirect block, create double indirect block, create triple indirect block, write data block.
- 3 reads, 2 writes: read single indirect, read double indirect, read triple indirect, write triple indirect, write data block
- Other combinations are possible

To read 1 byte, in the worst case:

- 4 reads: read single indirect, read double indirect, read triple indirect, read data block

67. Assume you have an inode-based filesystem. The filesystem has 512 byte blocks. Each inode has 10 direct, 1 single indirect, 1 double indirect, and 1 triple indirect block pointer. Block pointers are 4 bytes each. Assume the inode and any block free list is always in memory. Blocks are not cached.

a. What is the maximum file size that can be stored before
   1. The single indirect pointer is needed?
   2. The double indirect pointer is needed?
   3. The triple indirect pointer is needed?

b. What is the maximum file size supported?

c. What is the number of disk block reads required to read 1 byte from a file
   1. In the best case?
   2. In the worst case?

d. What is the number of disk block reads and writes required to write 1 byte to a file
   1. In the best case?
   2. In the worst case?

Answer: In one block, we can store 512/4=128 block addresses (blocking factor)

a. 1. 5KB (as inode contains 10 direct addresses, we can a file of size at most 10x512bytes, i.e., 5KB)
   2. 69KB (10 direct blocks + 128 indirect blocks. Therefore, 138x512bytes = 69KB)
   3. 8261KB (10+128+128^2blocks=16522x512bytes = 8261KB)

b. 1056837K (10+128+128^2+128^3blocks)

c. 1. 1
   2. 4

d. What is the number of disk block reads and writes required to write 1 byte to a file
   1. 1w
   2. 4r/1w

68. Suppose at time *x*, the user presses return on his keyboard. At time *x + 1* (seconds), a user process executes a **read(0, &c, 1)** call, which returns at time *x + 2*. Below are 12 events that can occur in the operating system. Choose the events that *must* occur in this scenario, and put them in the correct order that they will occur. Note, not all events will occur at all, and some events may occur more than once. Choose events from this list only.

a. The operating system fields a hardware interrupt and suspends the currently running process.

b. If the keyboard buffer is not full, the operating system reads a character from the keyboard device driver, typically using a privileged memory location.

c. The operating system reads a character from the keyboard device driver, typically using a privileged memory location.

d. The operating system checks its keyboard buffer to see if it is empty.

e. The operating system takes a character out of the keyboard buffer, and places it into a register.

f. The operating system checks its keyboard buffer to see if it is full.

g. The operating system calls the scheduler, which selects a user process to execute.

h. The operating system fields a software exception suspending the currently running process.

i. The operating system resets the keyboard DMA device driver so that the keyboard can do another transaction.

j. If the keyboard buffer is not full, the operating system puts the character into the buffer.

k. The operating system schedules a DMA transaction with the keyboard device driver, writing to the keyboard buffer.

l. The operating system takes a character out of the keyboard buffer, and places it into user memory.

Answer:

a: The user presses the return, and the keyboard generates an interrupt.

c: The OS reads the character from the keyboard device driver, thereby freeing the keyboard for further use.

f: Since the user has not asked for the character yet, it goes into the keyboard buffer. Thus, the OS needs to check to see if that buffer is full.

j: And if not full, the character goes into the buffer.

g: Now we're ready to call the scheduler.

h: One second later, the user process makes the **read()** system call.

d: Now we check to see if the keyboard is empty.

l: Since it is not empty, we put a character into user memory -- not into a register, since **read()** passes a pointer as an argument.

g: Back to the scheduler.

a/h/g: Since a second passes from the system call to its return, other processes must get executed in the meantime. Therefore steps *a*, *h* and *g* must be executed. Arguably, between steps *g* and *h* above, we can say the same thing.

69. Why is it important to try to balance file system I/O among the disks and controllers on system in a multitasking environment?

Answer: By spreading disk activity out amongst multiple disks and controllers you can have many different disk accesses all being served at once. A single disk/controller can only handle one request at a time. Spreading disk activity out should improve efficiency.

70. Explain what is meant by context and process switches. Under what conditions will they occur?

Answer: Context Switch

A context switch is where the context of the CPU (the collection of the CPU's registers) is replaced with another context. Usually the original context will be saved somewhere in memory.

A context switch usually occurs as a result of an interrupt.

Process Switch

A process switch is where the process that is currently running on the CPU is replaced with another process. A process switch may include the following steps:

- save the context of the current running process into its PCB
- change the status of the current process from running to something
- move the process to a particular queue
- select the next process to be made running
- remove this new process from the ready queue
- change the status of the new process
- place its context onto the CPU

A process switch will occur due to any of the following factors

- the current running process finishes
- the current running process requests an I/O operation
- the current running process' quantum runs out
- the current running process is suspended
- another I/O event completes resulting in a higher priority process obtaining the CPU
- the current running process generates an error condition

71. In virtually all systems that include DMA modules, DMA access to main memory is given higher priority than processor access ot main memory. Why?

Or

Cycle stealing occurs when the I/O processor and the CPU try to access the same memory module or the same bus simultaneously. Why does the I/O processor normally get priority?

Answer: A major aim of multi-programming is to achieve efficient use of the computer systems resources. This means you want the CPU executing instructions, the printer printing documents, the display displaying output and the disk drive storing information. That means you want interleaved I/O and CPU execution.

I/O devices are very, very slow when compared to the CPU. The CPU will want to access memory and the bus much more regularly than I/O devices of any description.

I/O is given precedence because you want the I/O device to use the memory or the bus and then to start doing its I/O operation. Once serviced the I/O device will take a long time before asking for the bus again. During this period the CPU can be executing.

72. What is the advantage of having different quantum sizes on different levels of a multilevel queuing system?

Answer: It sorts out IO bound processes from CPU bound and gives them higher priority any process with burst times less than quantum in highest queue gets priority over others good with preemptive scheduling between queues.

73. On a disk with 1000 tracks and 1000 sectors per track, with 2048 bytes per cluster calculate the size of a File Allocation Table

- in memory (in bytes)
- on disk (in blocks {=sectors})

Answer:

Sector Size (Commonly) = 512 bytes

Number of Sectors needed per cluster = 2048/512 = 4

Total No of Clusters = 1000*1000/4=250000

Assuming 4 bytes are used for each entry in the FAT, we need =250000x4 bytes per FAT

As in the disk we store 2 copies of the FAT we need the 2x250000x4 bytes in the disk.

74. Calculate the space needed for i-nodes where an i-node has 224 characters of assorted name and attribute data, and 8 pointers to (disk blocks). Assume block addresses are 4bytes.

Answer: Inode size = 224 + 8×4= 256bytes

75. Why a file access time will be low if it is physically stored contiguously on a disk?

Answer: Because of less seek time (horizontal latency).

76. Consider a Unix system with a disk of 10 million blocks, each of 8Kb, and an expected 5 million files.

i. How big must a block pointer be?
ii. How many inodes should be allocated?
iii. How much space on disk will the inodes take, if each is 256 bytes?

Answer:

For 10M blocks, the pointer must be at least 24 bits (16M variants).

Need at least 1 inode per file, so allocate 8M inodes.

8M inodes take up =8M x 256 = 2GB (of 80GB).

77. Estimate the data transfer rates, in bytes per second, of

i. A disk with 1000 sectors per track, each of 4KB. The disk spins at 7200 rpm.
ii. A memory bus 64 bits wide, with a cycle time of 166 nanoseconds.

Also comment whether memory bus can cope up with the disk.

Answer:

7200 rpm=120 revolutions per second. That is, 8 msec/revolution.

Data Reading Rate (rps * sectors/rev * bytes/sector)

= 120 * 1000 * 4KB = **480 MB/sec.**

We move 8 bytes in 166 nanoseconds on memory bus, so 48 bytes/microsecond. Thus data transfer rate on memory bus = **48 MB/sec**.

So, this memory bus will not keep up with the disk.

78. In an indexed allocation, block sizes are 256 bytes and block addresses are 4 bytes. If an index block is not sufficient to store the block addresses then another

index block is used and is linked with the current index block which is full. How many disk blocks are required to access 300'th block of a file in this organisation?

Answer:

Number addresses which we can store in an index block =256/4= 64

However, as the file is larger, one address is used to point to next index block. Thus, effectively 63 block numbers can be stored in an index block.

The address of 300'th block is available in index block 5 ((int) 300/63 + 1).

Thus, we have to read five index blocks to know the 300'th block number and one more disk access is needed to read that block.

79. Calculate how many physical sectors of size 512 bytes are required at most to store FAT in FAT-16 system.

Answer: We know two copies are stored.

As the file system is FAT-16, the disk addresses are 2 bytes and at most 216 entries will be seen in the FAT.

Thus, total memory required for both the FAT's = 2*216*2 bytes = 218 bytes.

No. of physical sectors needed = 218/512 = 512

80. A FAT-16 system uses 4 sectors as a cluster. What is the maximum acceptable partition size?

Answer: At most 2^{16} clusters are seen in this file system. Cluster size is 4×512 bytes.

Thus, maximum possible partition size = 2^{16}*4*512= 128MB.

81. A disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order, is 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130.

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk-scheduling algorithms?

1 FIFO

2 SSTF

3 SCAN (Elevator)

4 C-SCAN (Modified ELevator)

Answer:

FIFO

Service Order: 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Total Distance Head to traverse = (143–86) + (1470–86) + (1470–913) + (1774–913) + (1774–948) + (1509–948) + (1509–1022) + (1750–1022) + (1750–130) = 7081

SSTF

Service Order: 130, 80, 913, 948, 1022, 1470, 1509, 1750, 1774 = (143–130) + (130–80) + (1774–80) = 1757

SCAN

Service Order: 913, 948, 1022, 1470, 1509, 1750, 1774, 130,80 = (1774–143) + (1774–80) = 3325

CSCAN

Service Order: 913,948,1022,1470,1509,1750,1774,80, 130 = (1774–143)+(1774–80) + (130–80) = 3375

82. How long does it take to load a 64K program from a disk with an average seek time of 30ms and rotation time of 20ms, and with a track capacity of 32K:

(i) For a 2K page size?

(ii) For a 4K page size?

Assume that each page is stored contiguously on the disk, but that separate pages are distributed randomly on the disk.

Answer:

For 2K Page Size

Average Seek Time = 30 ms

Rotational Latency = 20/2=10ms

Reading Time to readk 2K = 20 × 2K/32K=1.25

Total time needed to load a page = 30 + 10 + 1.25 = 31.25ms

Program Size = 64K

No of Pages = 64K/2K =32

Therefore, time needed to load the program = 32×31.25 = 1000ms = 1sec

For 4K Page Size

Average Seek Time = 30 ms

Rotational Latency = 20/2=10ms

Reading Time to readk 2K = 20 × 4K/32K=2.5

Total Time Needed to Load a Page = 30 + 10 + 2.5 = 32.5ms

Program Size = 64K

No of Pages = 64K/4K =16

Therefore, time needed to load the program = 16×32.5 = 520ms.

83. A disk has the following specification:

Average seek time: 10 ms

Average latency: 5 ms

Average transfer time: 10 ms / block

A 150 Kb file is stored on the disk in 8 Kb blocks which are randomly distributed over a single cylinder. Calculate the expected total time to read the file.

Answer: As all the blocks are in the same cylinder, once head is positioned in the required track/cylinder we need not required to move. Thus, we have to seek once only.

Thus, seek time (horizontal latency) = 10ms.

Time per block including rotational latency and transfer time = 5ms + 10ms=15ms.

Thus total time need to read the file = 10 ms + 150/8 × 15 = 291.25ms (Blocks are randomly distributed over the cylinder. Certainly, we may employ some ordering on block reading to further reduce this time).

84. Assuming block size to be 1 KB and a disk address to be 4 bytes long, what is the maximum file size possible to store Unix inode based file organisation.

N= blocking factor = 1KB/4=1024/4=256

Single Largest File Size = $10 + 256 + 256^2 + 256^3$ blocks = 16.7GB

85. A Unix file system is having disk block size of 512 bytes and block address as 4 bytes. Find out how many disk accesses are needed to read.

(i) 1000^{th} byte of a file.

(ii) 10000^{th} byte of a file

(iii) 100000^{th} byte of a file.

Assume the file inode is currently in RAM.

Answer:

(i) 1 disk access: (The required byte (1000^{th}) is 2nd (1000/512 +1) block of the file whose number is directly available in the inode. Do remember that in the above statement, the divison operation is carried out in integer mode. That is, 1000/512 is taken as 1. We know, first ten data block's of the file will be having their numbers in the files inode itself. Thus, we need one disk access to read that data block having 1000^{th} byte..

(ii) 2 disk accesses: The required byte will be 20th block of the file. Inode contains first 10 data block numbers. Remaining 512/4 blocks addresses will be available in a indirect block. As we want 20th block of the file, first we have to read the indirect block and then the required block. Thus, we need 2 disk accesses.

(iii) 3 disk accesses (Here we need to access second indirect block).

86. In the Unix operating system, suppose the root directory "/" inode is in memory, and everything else is not in memory, assuming that all directories fit in one disk block, how many disk accesses are needed to read the first file block of "/home/ram/cs423/homework/report.txt"? Describe the purpose of each disk access

Answer:

1. Load table(content) for / root directory.
2. Load inode for home directory.
3. Load table(content) for home directory.
4. Load inode for alice directory.
5. Load table(content) for ram directory.
6. Load inode for cs423 directory.
7. Load table(content) for cs423 directory.
8. load inode for homework directory
9. load table for homework directory
10. Load inode for report.txt.
11. Load first file block for report.txt.

Thus, in total 11 disk accesses are needed.

87. A computer whose processes have 1024 pages in their address spaces keeps its page tables in memory. The overhead required for reading a word from the page table is 5 nsec. To reduce this overhead, the computer has a TLB, which holds 32 (virtual page, physical page frame) pairs and can do a lookup in 1 nsec. What hit rate is needed to reduce the mean overhead to 2 nsec?

Answer: If h is the hit ratio and we assume the page table lookup overhead includes the TLB miss, then the effective overhead instruction time is h + 5(1–h). If we equate this formulae to 2 ns, and solve for h, we find that h must be at least 0.75.

If h is the hit ratio and we assume the page table lookup overhead excludes the TLB miss, then the effective overhead instruction time is h + 6(1– h). If we equate this formulae to 2 ns, and solve for h, we find that h must be at least 0.8.

88. Does it mean a thread is a procedure call or subroutine?

Answer: No, a thread has nothing to do with a procedure call. For example, a program consists of 10 procedure calls (i=0,1,...,9), and the procedure i (i=1,...9) needs the results from the procedure call i-1 before it can run. In this case, a procedure cannot concurrently run with another procedure, due to the dependency. This process can only have one execution unit (one thread) even with 10 procedure calls. Remember, the key is to have multiple concurrently running entities.

89. Explain about passive vs. active entity in a process.

Answer: This can be easily understood for multithread process, each thread (with its TCB) is an active entity, and others such as code, global data and files are passive entities, shared by all threads within the same process. If you look at this carefully, what it implies is that only the thread part within a process is an active entity, and it has the state definition.

90. What is the state for a process with multiple threads?

Answer: There is no state definition for a process with more than one thread. Each thread (as an active entity) within a process has a state. Simply, suppose you have two threads in a process, one thread is in ready state, the other in waiting state, we cannot define the state for that process. Or even if two threads are in ready states, we do not define the state of the process to be in ready state.

91. Explain the tasks of hard and symbolic links.

Answer: Both hard and symbolic links are used to share files and directories between different users, or to access the same file from different directories of the same user. Creating a hard link causes that the "open file count" in the I-node table (system-wide open file table) to be incremented by one and therefore deleting that hard-link only decrements the "open file count" until it reaches to zero and then the file will be deleted. A sym-link is a file that contains a full pathname for another file (or directory). Deleting the target file causes a dangling sym-link pointer that must be deleted using garbage collection operation. We can have symbolic link to a directory in addition to the files/directories of other partitions. When we create a symbolic link file, a new inode and free data block is used. In the free data block, original files path is saved. Inode numbers of original and its symbolic link files are different unlike hard link files. When link count of a file becomes 0, its data blocks will be moved to free list.

92. For 64-bit machines, which address mapping techniques are used commonly?

1. Hierarchical paging technique
2. Hashed page table technique

Answer: In summary, for 64-bit architectures, hierarchical page tables are generally inappropriate. For example, the 64-bit UltraSparc would require seven levels of paging (meaning 7 memory accesses) for just translation of address. This is a prohibitively high overhead for address translation.

With "hashed page table technique" a large 64-bit address is first mapped onto a hash value (using a hash function) which is used as an index to lookup a hash-table entry. Then, the original 64-bit address is compared with the addresses that are already mapped and stroed in a link-list of addresses in the corresponding hash table entry, in order to find the matched address and that associated memory frame number. This technique is much faster than the first alternative above.

93. For each of the disk allocation techniques (contiguous, linked, indexed) specify its characteristics in terms of: 1) Ease of growth of file size; 2) How the free disk blocks are assigned; 3) Ease of random access to a location in a file: 4) Advantages and disadvantages.

- **Contiguous allocation:**
  (1) Files cannot grow, the max size of the file is set from the beginning
  (2) Contiguous free disk blocks are assigned to a file
  (3) Random access is easily performed
  (4) Advantage: scheduling is easy but external fragmentation wastes disk space and needs de-fragmentation.
- **Linked allocation:**
  (1) File grow easily by adding a new free block at the end of the linked free blocks
  (2) The blocks connected by linking a new block to the last block, where a small portion of each block is used for the address of the next block. In FAT technique, linked list of block numbers are kept in the memory.
  (3) Random access is not easy and needs to traverse the link of addresses.
  (4) Advantage: easy to grow or shrink, no external fragmentation. Disadvantage: random access is time consuming.
- **Indexed allocation**
  (1) Files grow easily and using indirect index tables a file can grow almost unlimited.
  (2) Each file has an index-block which is a regular disk block and the address of a free disk block is assigned to the next location in this index block.
  (3) Random access is easily performed with very low overhead.
  (4) Advantage: data blocks and index blocks are regular disk blocks so the method is very flexible, and does not need a FAT table to be kept in memory. Disadvantage: for small size files the index block space is not used efficiently; however in Unix a small number of indexes are kept in the FCB to accelerate the address mapping and eliminate the need to read index-block from file.

94. Twelve page requests occur in the following order: 9, 36, 3, 13, 9, 36, 25, 9, 36, 3, 13, and 25. Assume that physical memory initially starts empty and fully-associative paging is used. Which of the following statements are true?

I. If the physical memory size is 3 pages, then most-recently-used (MRU) paging will result in the same number of faults as if the optimal algorithm was used.

II. If first-in-first-out (FIFO) paging is used, and the physical memory size is raised from 3 pages to 4 pages, then Belady's anomaly will appear.

III. If least-recently-used (LRU) paging is used, and the physical memory size is raised from 3 pages to 4 pages, then Belady's anomaly will appear.

Answer: The optimal algorithm is to remove the page that will be needed farthest in the future. If this algorithm is used with 3 pages, then physical memory contains the following values:

After 9	(which caused a fault):	9
After 36	(which caused a fault):	36, 9
After 3	(which caused a fault):	3, 36, and 9 [from farthest to soonest used]
After 13	(which caused a fault):	13, 36, and 9
After 9	(which hit):	13, 9, and 36
After 36	(which hit):	13, 36, and 9
After 25	(which caused a fault):	25, 36, and 9
After 9	(which hit):	9, 25, and 36
After 36	(which hit):	9, 36, and 25
After 3	(which caused a fault):	3, 36, and 25
After 13	(which caused a fault):	13, 3, and 25
After 25	(which hit):	13, 3, and 25

Total of 7 faults

Suppose that MRU is used with 3 pages. Then physical memory contains the following values:

After 9	(which caused a fault):	9
After 36	(which caused a fault):	36, and 9
After 3	(which caused a fault):	3, 36,and 9[from most to least recently used]
After 13	(which caused a fault):	13, 36, and 9
After 9	(which hit):	9, 13, and 36
After 36	(which hit):	36, 9, and 13
After 25	(which caused a fault):	25, 9, and 13
After 9	(which hit):	9, 25, and 13
After 36	(which caused a fault):	36, 25, and 13
After 3	(which caused a fault):	3, 25, and 13
After 13	(which hit):	13, 3, and 25
After 25	(which hit):	25, 13, and 3

Total of 7 faults

Clearly, MRU and the optimal algorithm happen to generate the same number of faults in this case (though this is not a universal truth for memory reference strings in general). Thus, I is true.

LRU never demonstrates Belady's anomaly, which is when increasing the number of frames also increases the number of faults. In contrast, FIFO sometimes results in Belady's anomaly, so it is necessary to check. Suppose that FIFO is used with 3 pages. Then physical memory contains the following values:

After 9 (which caused a fault):	9
After 36 (which caused a fault):	9, 36
After 3 (which caused a fault):	9, 36, and 3 [from first to last loaded]
After 13 (which caused a fault):	36, 3, and 13
After 9 (which caused a fault):	3, 13, and 9
After 36 (which caused a fault):	13, 9, and 36
After 25 (which caused a fault):	9, 36, and 25
After 9 (which hit):	9, 36, and 25
After 36 (which hit):	9, 36, and 25
After 3 (which caused a fault):	36, 25, and 3
After 13 (which caused a fault):	25, 3, and 13
After 25 (which hit):	25, 3, and 13

Total of 9 faults

Suppose that FIFO is used with 4 pages. Then physical memory contains the following values:

After 9 (which caused a fault):	9
After 36 (which caused a fault):	9, 36
After 3 (which caused a fault):	9, 36, and 3 [from first to last loaded]
After 13 (which caused a fault):	9, 36, 3, and 13
After 9 (which hit):	9, 36, 3, and 13
After 36 (which hit):	9, 36, 3, and 13
After 25 (which caused a fault):	36, 3, 13, and 25
After 9 (which caused a fault):	3, 13, 25, and 9
After 36 (which caused a fault):	13, 25, 9, and 36
After 3 (which caused a fault):	25, 9, 36, and 3
After 13 (which caused a fault):	9, 36, 3, and 13
After 25 (which caused a fault):	36, 3, 13, and 25

Total of 10 faults

Thus, FIFO does demonstrate Belady's anomaly in this case. As noted earlier, LRU never does. So II is true and III is false.

95. Consider a cache with the following characteristics:

32-byte blocks

8-way set associative

256 sets

32-bit addresses

writeback policy

LRU replacement policy

(i) How many bytes of data storage are there?

$256 \times 8 \times 32 = 2^{18} = 64$ KB

(ii) How many tag bits per set?

32 -- $\log_2$ 256 --$\log_2$ 32 = 19 bits per set

(iii) What operation is needed upon a read-miss (the program wants to read from a memory location that is not in the cache)?

a. Find the LRU cache line to replace. If it is dirty, write the block to next level cache / memory.

b. Fetch 32-byte memory data from next level cache/ memory of that memory address and other data from the same block/line;

c. Update the tag bit of that cache line.

(iv) What operation is needed upon a write-miss (the program wants to write to a memory location that is not in the cache)?

a. Find the LRU cache line to replace. If it is dirty, write the block to next level cache / memory.

b. Fetch 32-byte memory data from next level cache/ memory of that memory address and other data from the same block/line;

c. Write to the memory location in that cache line. Mark the cache line as dirty. Update the tag bit of that cache line.

96. Consider a 3-way set associative cache. A, B, C, D are memory addresses that have the same index bits but different tag bits from each other. In a program, the reference sequence is as follows:

A, B, A, C, D, A, D, C, A, C

(i) What is the miss rate if the cache is using LRU replacement policy?

40%

(ii) What is the miss rate if the cache is using MRU replacement policy?

50%

(iii) Assuming the memory addresses being accessed are still A, B, C and D, provide a case of memory reference sequence with length=10, in which MRU performs better than LRU. Show the reference sequence and the miss rate of LRU and MRU policy.

For example: A,B,C,D,A,B,C,D,A,B

97. Given a 2 Kbytes two-way set associative cache with 16 byte lines and the following code:

```
for (int i = 0; i < 1000; i++) {
   A[i] = 40 * B[i];
}
```

(i) Compute the overall miss rate (assume array entries require 4 bytes)

Each array contains 1000 elements. Each cache line contains 4 words. Since each array element will be accessed only once, all misses shall be compulsory misses. Since every 4 words will be loaded or written back simultaneously in a cache, the miss rate is 25% since every 4th access misses.

(ii) What kind of cache locality is being exploited?

Answer: Spatial locality.

98. Given a 100 MHz machine with a with a miss penalty of 20 cycles, a hit time of 2 cycles, and a miss rate of 5%, calculate the average memory access time (AMAT). Suppose doubling the size of the cache decrease the miss rate to 3%, but causes the hit time to increase to 3 cycles and the miss penalty to increase to 21 cycles. What is the AMAT now?

Answer: AMAT = hit_time + miss_rate x miss_penalty

Since the clock rate is 100 MHz, the cycle time is: 1/(100 MHz) = 10 ns

Thus, AMAT = 10 ns $\times$ (2 + 20 $\times$ 0.05) = 30 ns

Here we needed to multiply by the cycle time because the hit_time and miss_penalty were given in cycles.

For the new setup, AMAT = 10 ns $\times$ (3 + 21 $\times$ 0.03) = 36.3 ns

99. Show the design of a 2-way set associative translation look-aside buffer with 32 entries. Assume the virtual address is 32 bits, the page offset is 12 bits, and the physical address is 20 bits. How many bits are required to implement the TLB if each table entry has a physical page number, a tag, a valid bit, and three access bits.

Answer:

Since the page offset is 12 bits, the remaining 20 bits from the 32 bit address make up the virtual page number. The TLB has the following parameters:

(1) Number of sets: 16 sets (Since (no. of entries)/(entries/set = 32/2)

(2) TLB index size: 4 bits (Since 2^4 = 16 sets)

(3) TLB tag size: 16 bits (Remaining bits in virtual address 32 – 12 – 4)

(4) Block size: 8 bits (remaining bits in physical address 20 – 12)

(5) # of blocks: 32 blocks (number of TLB entries)

TLB bits = number of blocks $\times$ (block size + tag size + 4) = 32 $\times$ (8 + 16 + 4) = 896 bits

OBJECTIVE TYPE QUESTIONS

1. Unix commands indicate successful completion by returning
A. 0 B. 1 C. –1 D. None

2. Exit status of a command in Unix is available in
A. A Shell variable
B. The program name itself
C. $$
D. $?

3. Programs use environment variables to
A. Store data between login sessions
B. Pass configuration settings to other programs
C. Act like cookies
D. None

4. A shell command is called "built-in" if
A. The shell does not call another program
B. The command is already compiled into the kernel
C. It is an internal command
D. it is an external command

5. When Unix opens a file, it selects an unused file descriptor
A. with the lowest value
B. that was closed most recently
C. With the highest value
D. None

6. Connections to a pipe remain in effect after exec because
A. The array of open files is part of the process, not the program
B. The parent adds the pipe to the environment
C. Child inherits all the opened files
D. None

7. A pipe can connect two children of the same process if
A. The parent creates the pipe before creating both children
B. The parent creates two pipes and connects them
C. First child will have always such an ability to connect to other brother processes
D. None

8. You cannot use fopen for pipes because
A. Pipes do not have file names
B. Pipes have a limited capacity
C. Pipes are not occupying any disk space
D. None

9. A pipe can be used to transfer data between
A. Exactly two processes
B. Any number of processes
C. One process to many processes
D. None

10. A server can handle several requests at once by
A. Using fork to create a new process for each request
B. Using socket to create a new socket for each request
C. Using createthread
D. None

11. A web server sends output of programs to the client by
A. Opening a new socket for the program
B. Using dup2 to redirect standard output
C. Opening httpd
D. None

12. All the threads in a process have access to
A. Only static global variable
B. All global variables in the process
C. All variables
D. None

13. When one thread calls exit
A. Only that thread stops running
B. All threads in the process stop
C. Unpredictable
D. None

14. Anonymous pipes are
A. Full-duplex B. Half-duplex
C. Simplex D. Broadcasting

15. Names pies are
A. Full-duplex B. Half-duplex
C. Simplex D. Broadcasting

16. Find wrong statement
A. Anonymous pipes can connect two processes in a network
B. Names pipes can connect two processes in a network
C. Sockets are end points of a connection
D. Port numbers are integers

17. Big-endian representation of decimal 258 is
A. 00000010 00000001
B. 00000001 00000010
C. 11111111 00000010
D. None

18. Exit status of a program in Windows is available in
A. A Shell variable
B. The program name itself
C. ERRORLEVEL environment variable
D. $?

19. Which system call replaces a process' core image?
A. fork B. execv C. run D. None

20. Which is not the benefit of using threads?
A. Responsiveness B. Economy
C. Fairness D. None

21. Which scheduler decides the degree of multiprogramming?
A. Admission scheduler B. Memory scheduler
C. CPU scheduler D. None

22. Which is not a CPU scheduling criterion?
A. CPU utilisation B. Response time
C. Reliability D. Waiting time

23. The aging algorithm with a = 1/2 is being used to predict run times. The previous 3 runs, from oldest to most recent, are 20, 40, and 20 msec. What is the prediction of the next time?
A. 20 B. 25 C. 30 D. None
Explanation: ((20 + 40)/2 + 20)/2 = 25

24. Which strategy is used in the Banker's algorithm for dealing with deadlocks?
A. Deadlock Ignorance
B. Deadlock Detection
C. Deadlock Avoidance
D. Deadlock Prevention

25. A system has 256 MB memory. The time to read or write a 32-bit memory word is 10 nsec. Assume the total processes take 3 times of memory taken by holes. What is the time needed to eliminate holes by compaction?
A. 167.772 ms B. 335.544 ms
C. 503.316 ms D. None

26. Priority inversion between two processes, one with high priority and the other with low priority, that share a critical section, will cause the following problem:
A. High priority process executes before low priority process and finishes faster than it should
B. Low priority process executes before high priority process and finishes faster than it should
C. High priority process waits for low priority process (that holds the semaphore) to finish, but the low priority process never gets scheduled
D. Low priority process changes priority temporary to the priority of the high priority process

27. Consider the 2-process solution to the critical section problem ('i' refers to the current process and 'j' is the other one):

```
repeat
while turn <> i do no-op;
<critical section>
turn:=j;
<remainder section>
until false;
```

This solution does NOT satisfy
A. Mutual Exclusion B. Progress
C. Bounded Waiting D. Both (B) and (C)
E. None of the above

28. Let us assume four processes being forked and all four processes are already active for 30 minutes in a single processor computer system. Let us assume that two of these four processes wait for information from the disk. Which of the state information about these processes is correct?
A. P1: ready, P2: running, P3: running, P4: blocking
B. P1: running, P2: blocking, P3: ready, P4: blocking
C. P1: blocking, P2: running, P3: blocking, P4: running
D. P1: blocking, P2: running, P3: new, P4: blocking

29. For a fixed number of processes, shortest-job-first scheduling algorithm:
I. Minimises average waiting time
II. Minimises average turn-around-time
III. Minimises CPU throughput
IV. Maximises average response-time
Which answers are true?
A. Only I & II B. Only I & IV
C. Only I, II, & III D. Only I, II, & IV
E. Only II & IV

30. There are four queues in a multi-level queuing scheduling system: The first queue runs the First-Come-First-Serve scheduling policy, and the three other queues run Round-Robin Scheduling Policies with quantum = 8, 16, and 32 milliseconds, respectively. What is the quantum of the First-Come-First-Served queue?
A. 8 milliseconds B. 64 milliseconds
C. Infinity D. None of the above

31. What is a signal?
A. Convoy situation caused by N-1 I/O processes and 1 CPU-bound process

B. Context switch between processes
C. Software notification of an event to a process
D. None

32. Let us assume two events (jobs) arriving each with the constant arrival rate of 10 processes per second. Let us assume one server (processor) with the constant service rate of 100 processes per second. What is the server utilisation under these assumptions?
A. 20% B. 100% C. 5% D. None

33. Let us consider a pre-emptive Shortest Job First scheduling where process A arrives at time 0 and needs to run for 1 hour. From the start time 0, other (short) processes will arrive every 1 minute and run for 2 minutes each. This situation will cause
A. Deadlock for all processes
B. Starvation for process A
C. Starvation for the short processes
D. None of the above

34. The CPU detects an interrupt
A. Using busy bit
B. Using interrupt handler
C. Using interrupt request line
D. None

35. How large a file can you access using only the single indirect, double indirect, and triple indirect pointers in the file inode if the block size is 8K (2^13) and pointers are 64 (2^6) bits?
A. 8,388,608 bytes (8MB = 2^23)
B. 8GB (2^33)
C. 8TB (2^43)
D. 16MB (2^24)
E. 32GB (2^35)
F. 64TB (2^46)
G. (a) + (b)
H. (a) + (b) + (c)
I. (d) + (e) + (f)
J. (d) + (e)

Explanation: Block addresses=8Bytes
Blocking factor=8KB/8B=1024
Therefore, maximum file size:
File data blocks that use single indirect= 1024*8KB = 8MB
File data blocks that use Double Indirect = 1024 *1024 × 8KB = 8 GB
File data blocks that use Triple Indirect = 1024×1024 × 1024 = 8TB
Thus, maximum single file size = 8MB + 8GB + 8TB

36. In a UNIX file-system the block-size has been set to 4K. Given that the inode blocks are already allocated on disk, how many free blocks need to be found to store a file of size 64K?
A. 16 B. 17 C. 64 D. 65
E. None of the above

Explanation: 64/4=16. First 10 data block numbers are stored in the i-node itself. To store next 6 data block numbers, we need one free data block. Thus, we need 16+1 blocks for this file.

37. Which of the following methods would you choose if the file requires frequent direct access and also external fragmentation is to be avoided (to keep disk utilisation high)?
A. Linked allocation B. Contiguous allocation
C. Indexed allocation D. None

38. Given a file of 100 blocks, what is the minimum number of disk I/O operations needed to insert a block in the middle of the file if linked list allocation is used (assume the block to be inserted is already in the memory)?
A. 2 B. 52 C. 101 D. 151

39. Given a file of 100 blocks, what is the minimum number of disk I/O operations to insert a block in the middle of the file if contiguous allocation is used (assume the block to be inserted is already in memory)?
A. 2 B. 52 C. 101 D. 151

40. Which of the following types of binding takes place during runtime?
A. Symbolic names to virtual addresses
B. Virtual to physical addresses
C. Both (A) and (B)
D. None

41. If the page table is broken up into pages, with an outer page-table to select the correct inner table, we are using
A. Paged swapping B. Inverted page table
C. Multi-level paging D. None of the above

42. The total space required for an inverted page table is proportional to the number of
A. Pages in the virtual memory
B. Pages * holes
C. Pages + overlays
D. Page frames in the physical memory
E. None of the above

43. Which one is not a privileged instruction:
A. Setting the timer register
B. Read the clock

C. Turn on timer interrupt
D. Clear memory

44. Which of the following methods is not typically used for passing parameters between a running program and the operating system:
A. Pass parameters in registers.
B. Store the parameters in a table in memory and pass the table address as a parameter.
C. Write the parameters in a file and pass the file address (location) in a register.
D. Push the parameters onto the stack by the program and pop off the stackby the operating system.

45. A currently running process can be put on a ready queue or one of the I/O queuesby each of the following except:
A. The process did an illegal memory access.
B. The process issued an I/O request
C. There was an interrupt
D. The process issued a system call

46. What will be the probable output of the following program?

```
int main(){
int value = 5;
pid_t pid;
pid = fork();
if(pid==0){
value +=15;
}
printf("%d ", value);
   return 0;
   }
```

A. 5 20 B. 20 5 C. 5 5 D. a & b
E. None

47. Which of the following components of program state are shared across threads in a multithreaded process?
A. Register values B. Heap memory
C. Global variables D. Stack memory

Answer: The Global values and heap memory are shared across a multi-threaded process. Register values and stack memory are private to each thread.

48. Match the operating system abstractions in the left column to the hardware components in the right column

a. Thread	1. Interrupt
b. Virtual Address Space	2. Memory
c. File System	3. CPU
d. Signal	4. Disk

A. a-2, b-4, c-3, d-1 B. a-1, b-2, c-3, d-4
C. a-3, b-2, c-4, d-1 D. a-4, b-2, c-2, d-1

49. Which of the following file streams is NOT opened automatically in a UNIX program?
A. Standard input B. Standard output
C. Standard error D. Standard terminal

50. Which of the following is NOT an advantage provided by shared libraries?
A. They save disk space
B. They save space in main memory
C. Multiple versions of the same library can be loaded into main memory
D. None of the above

51. Which of the following frees the CPU from having to deal with transfer of memory to/from I/O devices?
A. Interrupts B. DMA
C. Buffer Cache D. Device Driver

52. Hard disk format technique that compensates for track-to-track seek time (to enhance disk performance when reading multiple tracks).
A. Buffer Cache B. Virtual Geometry
C. Cylinder Skew D. Interleaving

53. Which of the following scheduling algorithms will not have starvation?
A. FIFO B. Round Robin
C. Shortest-Job-First D. Priority
E. Both (A) and (B) F. Both (B) and (C)
G. None of the above

54. In FCFS, I/O bound processes may have to wait long in the ready queue waiting for a CPU bound job to finish. This is known as
A. Aging B. Priority inversion
C. Belady's anomaly D. Convoy effect
E. None of the above

55. In a system with paging only (every program is a collection of pages, no segments), which of the following can occur?
A. External fragmentation
B. Internal fragmentation
C. Deadlock when two processes each has been allocated a page frame, and each wants to acquire one more page frame, but the system has no page frame that is still free.
D. All of the above

56. If the page table is broken up into pages, with an outer page-table to select the correct inner table, we are using

A. Paged segmentation
B. Inverted page table
C. Multilevel paging
D. None of the above

57. What is the expected CPU utilisation for a system that has three processes running, each of which spends 80% (on average) of its time waiting for I/O.

A. 98% B. 34% C. 56% D. 49%

Explanation: $1–0.8^3=0.488$

58. Find incorrect one

A. NTFS file system cannot be installed on floppy drives
B. NTFS file system cannot be installed on a hard disk of size 100MB
C. NTFS file system does not need any repairing utilities
D. NTFS file system supports Unicode
E. None

59. ____ number of records in NTFS MFT are system related

A. 2^8 B. 16 C. 256 D. 1024

60. Largest file size in FAT-32 file system

A. 2GB B. 4GB
C. 2bytes less than 4GB D. None

61. Find correct one

A. Race conditions will not occur in uniprocessor systems
B. SJF can be implemented a priority algorithm where arrival time can be taken as priority
C. A process in ready state can go to either running or exit.
D. Two-phase lock protocols guarantees deadlock free concurrent transactions.

62. On a system a process executes for T time units before making an I/O request. Context switching overhead is S time units and time slice is Q units. If Q becomes negligible then efficiency becomes

A. T/(T+S) B. T/(T+Q)
C. Infinity D. Approaches 0

63. Find incorrect statement with regard to increasing file size in Unix file system.

A. Add quadruple (4^{th} level) in-direct block.
B. Increase the block size
C. Both A & B
D. None

64. Find incorrect one regarding the use of large sized pages in VM systems.

A. Reduces the number of processes that can have pages in RAM, i.e. the level of multiprogramming which is possible
B. Very expensive to swap in due to a lot of information to retrieve from disk
C. Greater possibility of internal fragmentation with programs which are smaller than page size
D. Larger amount of process information in RAM therefore the page fault should be lower
E. None

65. Find incorrect one regarding the use of small sized pages in VM systems.

A. Allows more processes to have pages in RAM which makes it possible to interleave I/O and CPU to a higher extent
B. Very little information to retrieve so swapping is faster
C. Much lower possibility of internal fragmentation
D. Due to small size of pages it is possible that page fault rate could be very high
E. None

66. Which of the following instructions (or instruction sequences) are run in other than kernel mode?

A. Disable all interrupts.
B. Set the time of day clock.
C. Change the memory map.
D. Write to the hard disk controller register.
E. Write all buffered blocks associated with a file back to disk (fsync).

67. The long-term scheduler:

A. Is responsible for moving jobs between queues in a multi-level feedback queue scheduling algorithm.
B. Makes decisions about the distant future of a system.
C. Sets a process's initial priority.
D. Decides whether a process enters the system, or whether it must wait for other processes to exit before entering the system.

Explanation: Option A is wrong, because movement in an MLFQ is dictated by the job's short-term behavior. Option B is too vague, and Option C is wrong because priority is typically set by a combination of the user and the user's priority.

68. Which of the following is the definition of a safe state:

A. A safe state is one where there is an ordering of the processes P1 to PN such that if Pi is less than Pj, then Pi does not need any of Pj's resources to complete.

B. A safe state is one where there is an ordering of the processes P1 to PN such that if Pj is less than Pi, then Pi does not need any of Pj's resources to complete.

C. A safe state is one where there is an ordering of the processes P1 to PN such that when all Pi, i < j, have completed, Pj can get all of its resources without waiting.

D. A safe state is one where there is an ordering of the processes P1 to PN such that when all Pi, i > j, have completed, Pj can get all of its resources without waiting.

Explanation: Answer is option C. However, option a is close to correct, however Pj and Pi may use instances of the same resource class.

69. Why do we not want a time quantum to be too small?

A. Because CPU-bound processes will not get their fair share of the CPU, and overall CPU utilisation will be decreased.

B. Because I/O-bound processes will not get their fair share of the CPU, and overall CPU utilisation will be decreased.

C. Because CPU-bound processes will not get their fair share of the CPU, and overall average process turnaround time will be increased.

D. Because I/O-bound processes will not get their fair share of the CPU, and overall average process turnaround time will be increased.

E. Because context-switch overhead will increase the average process turnaround time.

F. Because the operating system will own the CPU for too long, and overall average process turnaround time will be increased.

Explanation: Answer is E. Clearly, a time quantum that is too small will spend too much time context-switching. There are no problems with fairness (eliminating answers A through D). So what is the difference between answers E and F ? The reason that context switches are expensive has more to do with cache-flushing than operating system overhead. Thus, E is the correct answer.

70. Why do we not want a time quantum to be too big?

A. Because CPU-bound processes will not get their fair share of the CPU, and overall CPU utilisation will be decreased.

B. Because I/O-bound processes will not get their fair share of the CPU, and overall CPU utilisation will be decreased.

C. Because CPU-bound processes will not get their fair share of the CPU, and overall average process turnaround time will be increased.

D. Because I/O-bound processes will not get their fair share of the CPU, and overall average process turnaround time will be increased.

E. Because context-switch overhead will increase the average process turnaround time.

F. Because the operating system will own the CPU for too long, and overall average process turnaround time will be increased.

Explanation: Answer is D. Longer time quanta will be harder on I/O bound processes. However, this has nothing to do with CPU utilisation -- FCFS scheduling with an infinite time quantum will have better CPU utilisation than any round robin scheduling. However, process turnaround time is greatly increased when CPU-bound jobs get more of the CPU.

71. Which of the following statements is false?

A. Kernel-only memory locations protect devices on the memory bus from user processes.

B. Kernel-only memory locations protect the interrupt vector from user processes.

C. DMA protects the CPU from mass disk transfers.

D. A hardware timer protects the system from programs that do not voluntarily give up the CPU.

Explanation: Answer is C. This one has to do with the definition of "protects." Certainly kernel-only memory locations prevent users from corrupting devices and the interrupt vector, so A and B are true. Similarly, D is true, because without a hardware timer, a user process could prevent other processes from running. How about option C? Certainly DMA frees up the CPU -- without DMA, the CPU will have to be more involved in mass transfers. But that is not a correctness feature, but a performance feature. Therefore, the answer is C. However, option "none of the above" is also if we want to argue that C is true. However, A, B and D are clearly true, and will not be accepted as answers.

72. Two FAT 16 file systems with 8KB and 32KB cluster sizes are available. When we store 1000 files each of size 40KB it was observed that in the first one 391MB they are occupying while in the second the same are occupying more than 620MB. Probable reason for this difference is

A. External fragmentation

B. Internal fragmentation

C. Less addresses

D. None

Explanation: Answer is option A. In the first case the 40KB file exactly occupies 5 clusters of 8KB each

while in the second one two clusters are used in which the second cluster contains only 8KB of data and remaining goes as wastage (external fragmentation). Thus, more space is consumed in the second system.

73. Find incorrect statement about FAT-32 compared to VFAT.

A. The FAT 32 system enables users to manage 8 GB volumes with cluster sizes of just 4 kB. For hard drives up to a maximum of 32 GB, the cluster size is 16 kB.

B. FAT 32 also does not limit the number of directories or files in the root directory.

C. FAT-32 supports long file names.

D. None

74. In general, there will be _____ inodes than directories in Unix file system.

A. More B. Less C. Same D. None

75. In which page replacement the following objective is employed "replace the page whose next reference will be furthest in the future".

A. FIFO B. LRU

C. Second Clock D. Optimal

76. Consider the following C code:

```
/* Initialize semaphores */
mutex1 = 1;
mutex2 = 1;
mutex3 = 1;
mutex4 = 1;
```

void T1() {	void T2() {
P(mutex4);	P(mutex1);
P(mutex2);	P(mutex2);
P(mutex3);	P(mutex4);
/* Access Data */	/* Access Data */
V(mutex4);	V(mutex1);
V(mutex2);	V(mutex2);
V(mutex3);	V(mutex4);
}	}

Which of the following instruction execution sequences leads to deadlock?

A. T1's P(mutex4), T2's P(mutex1), T2's P(mutex2), T1's P(mutex2)

B. T1's P(mutex4), T2's P(mutex1), T2's P(mutex2), T2's P(mutex4)

C. T1's P(mutex4), T2's P(mutex1), T2's P(mutex2), T2's P(mutex4), T1's P(mutex2)

D. None

77. Possible output of the following program is:

```
int main() {
if (fork() == 0) {
printf("a");
exit(0);
}
else {
printf("b");
waitpid(-1, NULL, 0);
}
printf("c");
exit(0);
}
```

A. abc B. acb C. bac D. bca

E. cab

78. Assuming that the file "barfoo.txt" contains a string "barfoo", what will be the output from the following program?

```
int main(){
int fd;
char c;
fd = open("barfoo.txt", O_RDONLY, 0);
read(fd, &c, 1);
if (fork() == 0) {
read(fd, &c, 1);
exit(0);
}
wait(NULL); /* wait for child to terminate */
read(fd, &c, 1);
printf("c = %c", c);
exit(0);
}
```

A. c=b

B. c=a

C. c=r

D. Cannot be predictable because of fork() system call

Explanation: Opened file descriptors are inheritable by the child processes. In the given program, a file "barfoo.txt" is opened and then fork() is called after reading one character. Parent process blocks till child finishes its duty as wait is called. Child process reads another character from the file and then exits. When parent process is started again, it reads one more character (i.e., r) and stores in the variable c. The same will be printed. Thus, we get c=r output from this program.

79. What is the output of the following program?

```
int main(){
```

```
int fd1, fd2, fd3;
fd1 = open("foo.txt", O_RDONLY, 0);
fd2 = open("bar.txt", O_RDONLY, 0);
close(fd1);
fd3 = open("baz.txt", O_RDONLY, 0);
printf("%d", fd3);
exit(0);
}
```

A. 0 B. 1 C. 2 D. 3

Explanation: Answer is D. Every process will be having its own opened file table. Row indexes of this table is called as file descriptor. Every process will be having file streams known as input, output and error stream which are referred with file descriptors 0, 1 and 2 which are indirectly refers to the rows of opened file table. Whenever we open a file, OS searches this table from 0^{th} row onwards for an empty row, whenever it founds one it assigns that for the file which we are trying to open. Thus, that row index becomes that files, file descriptor as long as that process is running or till we close. In this table, each contains file related dynamic information such as how many bytes are read/written in addition to file security related information. In the above program, we have opened two files thus fd1 and fd2 values becomes 3 and 4. After that fd1 is closed and then another file is opened. Thus, fd3 value becomes 3 as row 3 of the table is free now.

80. Consider the following C program, where the disk file barfoo.txt consists of the 6 ASCII characters"barfoo".

```
int main(){
int fd1, fd2;
char c;
fd1 = open("barfoo.txt", O_RDONLY, 0);
fd2 = open("barfoo.txt", O_WRONLY, 0);
read(fd1, &c, 1);
c = 'z';
write(fd2, &c, 1);
read(fd1, &c, 1);
printf("%c", c);
exit(0);
}
```

The output is:

A. b B. a C. r D. None

Explanation: Two file descriptors are created for the samefile"barfoo.txt"; through fd1we can read from the file while through fd2 we can write into the file. We read a character through fd1 and write something through fd2. Both are treated separately. Thus, when we try to read another character through fd1, c value becomes character 'a'. Thus, we get 'a' as output.

81. What is the output of the following program?

```
int main(){
int fd1, fd2;
char c;
fd1 = open("barfoo.txt", O_RDONLY, 0);
fd2 = open("barfoo.txt", O_RDONLY, 0);
read(fd1, &c, 1);
dup2(fd2, fd1);
read(fd1, &c, 1);
printf("%c", c);
exit(0);
}
```

A. b B. a C. r D. None

Explanation: We are creating a duplicate descriptor for fd2 as fd1. At this junction, we have not used fd2 for any I/O at all. Thus, offset value in the opened file table will be having 0. When we call dup2(), the fd2 row as a whole copied to fd1. Thus, when we read via fd1, we get first character in the file, 'b' as the output.

82. Assume that the main() calls the following function test() exactly once.What will be the possible output?

```
void test(void){
if ( fork() == 0 ){
printf(''0'');
exit(0);
}
printf(''1'');
}
```

A. 01 B. 10 C. 11 D. A and B

Explanation: When fork() is called, new process is created. Statements in if condition are executed in the child process. Thus, we may get "0" as the standard output. Similarly, statement printf("1") is executed in parent process. As we do not know which process runs after fork(), we may get 01 or 10 as the outputs of the above program.

83. Assume that the main() calls the following function test() exactly once, what is the possible output?

```
void test(void){
int status;
int counter = 0;
if ( fork() == 0 ){
counter++;
printf(''%d'', counter);
exit(0);
```

```
}
wait(&status);
if ( fork() == 0 ){
counter++;
printf(''%d'', counter);
exit(0);
}
wait(&status);
}
```

A. 00 B. 01 C. 11 D. 10

Explanation: Child process inherits the automatic variables of its parent. However, whatever operations child carries on these inherited members will not visible in parent. Thus, when first fork() is invoked a new child process will be created which increments counter value to 1 and prints the same before terminating. During this time parent will be blocked because of wait() system call. By the time second fork() is called, parent sees counter value as 0. Thus, when second child is created, it also sees counter value as 0 initially. Then, it increments counter value and prints. Thus, we get "1" again. In a nutshell, the above program gives "11" as output.

84. Consider the following program

```
1. int main() {
2. int i = 0;
3. pid_t pid1, pid2;
4. if((pid1 = fork()) == 0) {
5. i++;
6. if((pid2 = fork()) == 0) {
7. i++;
8. printf("i: %d\n", ++i);
9. exit(0);
10. }
11. printf("i: %d\n", i);
12. }
13. else {
14. if(waitpid(pid1, NULL, 0) > 0) {
15. printf("i: %d\n", ++i);
16. }
17. }
18. printf("i: %d\n", ++i);
19. exit(0);
20. }
```

Possible outputs of the above program are given as:

i: 1	i: 1	i: 1	i: 3	i: 1	i: 1
i: 3	i: 1	i: 2	i: 3	i: 2	i: 2
i: 2	i: 2	i: 3	i: 3	i: 1	i: 3
i: 1	i: 3	i: 2	i: 1	i: 2	i: 4
i: 2		i: 3	i: 2	i: 3	i: 2
		i: 2			i: 3

Which is the valid output?

A. I B. V C. VI D. A and B

Explanation: Assume P0 is the main process. After first fork(), new process say P1 is created. Instruction 18 will be executed both in P0 and P1. Also, after fork(), if P0 runs then it gets blocked because of wiatpid() function. Thus, P1 gets executed. If P1 is started then, it increments i value to 1 and then calls fork() again. Thus, another child process say P2 will be created. Statement 11 will be executed only in P1. Do remember, P2 is calling exit(), thus it will not execute statement 11. In addition, we cannot say whether P1 or P2 runs first. If P1 runs, statement 11 gets executed, thus we may get 1 as output. Otherwise if P2 runs, we may get 3 as output. Thus, because of P1 and P2 execution we may supposed to get output as 1and 3 or 3 and 1. Thus, some options suchas II,III, IV, can be eliminated. Users can analyse this program further.

85. Consider the following program.

```
int main(){
int status;
int counter = 2;
pid_t pid;
if ((pid = fork()) == 0) {
          counter += !fork();
          printf("%d", counter);
          fflush(stdout);
          counter++;
      }
else {
if (waitpid(pid, &status, 0) > 0) {
                  printf("6");
                  fflush (stdout);
            }
counter += 2;
   }
printf("%d", counter);
fflush(stdout);
exit(0);
}
```

What is the output of the above program?

A. 236434 B. 236433
C. 236443 D. None

Explanation: Assume P0 is the main process. When first fork() is called, a new process (say P1) will be created and P0 gets blocked on this. Inside P1, initial value of counter is 0. When counter += !fork(); statement is executed, counter value will be 2 in P1 and 3 in the new process(P2). The following printf will be executed in both. Thus, we may supposed to get either 23 or 32. After this, counter will be incremented in both P1 and P2. Thus, they will be having 3 and 4 as their respective counter values. Do remember that the last printf statement will be executed all the processes P0, P1 and P2. P0 is blocked on P1. Thus, the last printf statement when executed in P1 and P2, we may get 34 or 43. However, if P0 is started, we get 6 and then 4(because of last printf).Thus, possible results are matching with option A.

86. Consider the following program.

```
int main(){
int status;
int counter = 1;
if (fork() == 0) {
              counter++;
            printf("%d",counter);
       }
else {
              if (fork() == 0) {
                           printf("5",);
                           counter--;
                           printf("%d",counter);
                           exit(0);
                          }
else {
                          if (wait(&status) > 0) {
                                  printf("6");
                                  }
}
}
printf("3");
exit(0);
}
```

Not possible output from this program is:

A. 253063 B. 251633
C. 520633 D. 506323

87. Assuming that the semaphores S and Q are initialized to 1 at the beginning, the following concurrent programs leads to

Code for Procces A	Code for Proccss B
do {	do {
wait (S);	wait (Q;
wait (Q);	wait (S);
// some operations here	// some operations here
signal (Q);	signal (S);
signal (S);	signal (Q);
} while (TRUE);	} while (TRUE);

A. Successful completion
B. Indefinite blocking
C. Process B will be forced to get terminated by Process A.
D. None

88. What are the two possibilities in terms of the address space of a newly created process?

A. The child process has a new program loaded into it.
B. The child process has an exact copy of the address space of the parent process including program
C. The child process has an exact copy of the address space of the parent process including data
D. The child process has an exact copy of the address space of the parent process including program and data

89. Fundamental models of IPC

A. Shared address space
B. Shared code
C. Shared memory
D. None

90. Find incorrect statement regarding the possible reason for excessive thrashing.

A. Running too many processes for too short a time slice
B. Processes that use more memory then you have in their working set.
C. Bad page replacement algorithm that picks bad pages.
D. Global page replacement where processes are cannibalising each other.
E. Optimising your scheduler for high-memory loads.

91. It is proposed to have FAT16 file system created on a disk with size 32MB and 1KB blocks. Size of FAT on the disk

A. 20KB B. 64KB C. 23KB D. None

Explanation: Size of FAT 16 = number of bits per entry * number of blocks

number of blocks = $\frac{32MB}{1KB} = \frac{2^{25}}{2^{10}} = 2^{15}$ block

= 32 K blocks

Size of FAT16 = 16 bits × 2^{15} blocks = 2^{19} bits

= $\frac{2^{19}}{8 \times 1024}$ = 64 KB

92. Find correct statement.

A. Bootstrap program is a part of the kernel of the Unix operating system.

B. Bootstrap program is a small program that loads the Kernel of operating system from hard disk to the computer memory.

C. Bootstrap program is kept in the dynamic RAM memory of the computer.

D. Bootstrap program is the second program that runs after the OS is loaded.

93. Find in-correct statement

A. CPU changes its mode from user to monitor when an I/O instruction is executed.

B. In user mode the user's programs are protected against other users' programs from being accessed or modified.

C. Any system call to the operating system causes the CPU to change its mode to monitor.

D. All the operations of the operating system's kernel are performed in the monitor mode of the CPU.

94. Find in-correct statement

A. When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process.

B. Direct Memory Access (DMA) can be used to reduce the I/O operations during the context switching.

C. Computer hardware and special purpose CPU instructions can be used to reduce the context switching time.

D. During context switching, a process's current instruction is completed (i.e., its fetch, decode, execute and write back cycles are completed) before loading the other process.

95. Find in-correct statement

A. A parent process creates a child process by the means of a copy and paste program within the operating system.

B. The child process is created by copying all the context of the parent process into a new slot of the File Control Block (FCB) table.

C. The parent and child process will be running concurrently and the parent process is in charge of all the resources the child process is using.

D. The child process inherits opened files from its parent.

96. Find in-correct statement.

A. Consistency semantics in a file system specifies how multiple users may access to a shared file simultaneously.

B. Unix primarily supports group based permissions.

C. In Unix file system writing to an open file is visible immediately to other users of the same open file.

D. A File Control Block (FCB) contains all the information about the process that owns this file.

97. Find valid statement

A. When a kernel thread is created, process structure is created in swap memory.

B. Communication among threads is difficult since the threads have separate address spaces and do not share memory.

C. In mapping the user-level threads to kernel threads, many-to-many model can be used in a multi-processor system but if a user-level thread is blocked then all user level threads become blocked.

D. The many-to-many model allows the operating system to create a sufficient number of kernel threads.

98. Find invalid statement.

A. Critical section is a global variable that two or more processes can access and modify.

B. Critical section is a part of a program where two or more processes assess a shared resource.

C. Any solution to the critical section problem must satisfy "mutual exclusion", "progress" and "bounded waiting" conditions.

D. Semaphore is a means to provide both mutual exclusion and synchronization of the concurrent processes.

99. Find in-correct statement.

A. A paging MMU with 32-bit logical address and 32Kbytes page size requires a page table with 2^{17} entries.

B. In a two-level page table, we break a large page table into outer and inner tables so that we use table entries for only those pages of a process that need to be in memory.

C. Inverted page table puts both the page number and its frame number into a page table entry. Therefore, we must search the table locations in order to find the logical address and its corresponding memory frame number in one of the table entries.

D. A hashed page table solution is used when the size of the logical address is very large (e.g., 64 bit address). In this mechanism, a logical address is hashed into a hash table address (using a hash function). Different addresses may be hashed into the same hash address.

100. Find which of the following operation which need not be privileged.

A. Clearing memory

B. Turn off interrupts

C. Switch from user mode to kernel mode

D. Set value of timer

101. Find invalid statement

A. Control and status registers are usually not visible to user programs.

B. An interrupt will always lead to an operating system action that corrects a problem and allows the interrupted process to continue running.

C. An interrupt handler will usually disable further interrupts temporarily. Some interrupt handlers must not be interrupted; some can be either way.

D. Management of a process runtime stack usually requires a stack pointer, a stack base, and a stack limit.

102. Find invalid statement

A. One benefit/goal of memory management is the protection of each user process address space from unauthorized manipulation by another user process.

B. Virtual memory divides a program's address space into pages which can be placed in main memory according to some scheduling and allocation policy.

C. One of the goals of a scheduling policy for the processor could be to give each process approximately equal access to the processor.

D. One of the goals of a scheduling policy for resources other than the processor is to give each process approximately equal access to the resource.

103. Find invalid statement.

A. One useful measure of a system's ability to handle a large number of jobs is throughput, defined as the total time required to execute the set of jobs.

B. A process in a Blocked state is waiting for an event, but that event is not related to the virtual memory subsystem.

C. Process state transition diagrams are helpful to understand the ways in which a process and the OS must react to different situations.

D. A process control block would typically contain information about memory allocations, threads and other information used to manage the process.

104. Find invalid statement

A. A process control block would typically contain information about memory allocations, threads and other information used to manage the process.

B. Kernel-level threads are only used within the OS kernel.

C. A symmetric multiprocessor allows some logically concurrent operations to occur simultaneously, and this will require additional care in the design of system and application programs.

D. It is possible for an interrupt handler to execute in its own thread.

105. Find invalid statement

A. A binary semaphore takes on numerical values 0 and 1 only.

B. An atomic operation is a machine instruction or a sequence of instructions that must be executed to completion without interruption.

C. Deadlock is a situation in which two or more processes (or threads) are waiting for an event that will occur in the future.

D. Starvation is a situation in which a process is denied access to a resource because of the competitive activity of other, possibly unrelated, processes.

106. Find valid statement.

A. While a process is blocked on a semaphore's queue, it cannot be called as busywaiting.

B. Circular waiting is a necessary condition for deadlock, but not a sufficient condition.

C. Mutual exclusion can be enforced with a general semaphore whose initial value is greater than 1.

D. External fragmentation can occur in disk systems which does not employ contiguous allocation policy.

107. Find valid statement

A. Pages that are shared between two or more processes can never be swapped out to the disk.

B. The allocated portions of memory using a buddy system are all the same size.

C. Demand paging requires the programmer to take specific action to force the operating system to load a particular virtual memory page.

D. Prepaging is one possibility for the fetch policy in a virtual memory system.

108. Find valid statement

A. The resident set of a process can be changed in response to actions by other processes.

B. The working set of a process can be changed in response to actions by other processes.

C. The translation lookaside buffer is a software data structure that supports the virtual memory address translation operation.

D. In a symmetric multiprocessor, threads can always be run on any processor.

109. A certain architecture supports indirect, direct, and register addressing modes for use in identifying operands for arithmetic instructions. Which of the following cannot be achieved with a single instruction?

A. Specifying a register number in the instruction such that the register contains the value of an operand that will be used by the operation.

B. Specifying a register number in the instruction such that the register will serve as the destination for the operation's output.

C. Specifying an operand value in the instruction such that the value will be used by the operation.

D. Specifying a memory location in the instruction such that the memory location contains the value of an operand that will be used by the operation.

E. Specifying a memory location in the instruction such that the value at that location specifies yet another memory location which in turn contains the value of an operand that will be used by the instruction.

110. The designers of a cache system need to reduce the number of cache misses that occur in a certain group of programs. Which of the following statements is/are true?

I. If compulsory misses are most common, then the designers should consider increasing the cache line size to take better advantage of locality.

II. If capacity misses are most common, then the designers should consider increasing the total cache size so it can contain more lines.

III. If conflict misses are most common, then the designers should consider increasing the cache's associativity, in order to provide more flexibility when a collision occurs.

A. III only B. I and II only

C. II and III only D. I, II, and III

E. None of the above

111. A system designer has put forth a design for a direct-mapped cache C. Suppose that reading a memory address A is anticipated to have an overall expected average latency T (A,C) (including the average cost of cache misses on C). Which of the following statements is/are true?

I. If C contains several words per cache line, then the index of the cache line for A is given by the rightmost bits of A.

II. Suppose C is a unified cache and C' is a comparable split cache with the same total capacity and cache line size as C. Then, generally, T (A,C') > T (A,C).

III. Suppose C" is a two-way set associative cache with the same total capacity and cache line size as C. Then, generally, T (A,C") < T (A,C).

A. III only B. I and II only

C. II and III only D. I, II, and III

E. None of the above

112. We have two caches with the following details.

Cache1 uses a direct-mapped cache containing 2 words per cache line. It would have an instruction miss rate of 3% and a data miss rate of 8%.

Cache2 uses a 2-way set associative cache containing 8 words per cache line. It would have an instruction miss rate of 1% and a data miss rate of 4%.

Assume approximately 0.5 data references on average per instruction. The cache miss penalty in clock cycles is 8 + cache line size in words; for example, the penalty with

1-word cache lines would be 8 + 1 = 9 clock cycles.

Let D1 = cycles wasted by Cache1 on cache miss penalties (per instruction)

Let D2 = cycles wasted by Cache2 on cache miss penalties (per instruction)

On average, how many clock cycles will be wasted by each on cache miss penalties?

A. D1 = 0.45, D2 = 0.48

B. D1 = 0.70, D2 = 0.40

C. D1 = 0.70, D2 = 0.48

D. D1 = 1.10, D2 = 0.40

E. D1 = 1.10, D2 = 0.96

Explanation: Let p be the cache miss penalty, in clock cycles, and let m_i and m_d indicate the instruction and data miss rates, respectively. Then the total time spent on penalties, for an average instruction, is p $(1 * m_i + 0.5 * m_d)$, since there are about 0.5 data references per instruction. Consequently, the total penalty for D1(=10*(0.03+0.5*0.08)) and D2(=16*(0.01+0.5*0.04)), are 0.70 and 0.48, respectively.

113. Consider the following alternative designs of a cache.

Design #1 is a direct-mapped cache of 8 1-word cache lines. The miss penalty is 8 clock cycles.

Design #2 can store the same total number of items as Design #1, but it is a two-way associative cache of 1-word cache lines. Least-recently-used is utilized to determine which items should be removed from the cache. The miss penalty is 10 clock cycles.

Assuming the following eight memory references, cache miss penalties in cycles for both:

Memory References: 0, 3, 14, 11, 4, 11, 8, 0

A. Design #1 spends 56 cycles and Design #2 spends 60 cycles
B. Design #1 spends 56 cycles and Design #2 spends 70 cycles
C. Design #1 spends 48 cycles and Design #2 spends 70 cycles
D. Design #1 spends 64 cycles and Design #2 spends 60 cycles
E. Design #1 spends 64 cycles and Design #2 spends 80 cycles

Explanation:

Design#1: The following table illustrates the miss and hit with given cache and given page reference sequence. We find in total 7 misses. Thus, 8x7=56 cycles as the miss penality.

Memory Reference/ Block No	0	1	2	3	4	5	6	7	
0	*								Miss
3				*					Miss
14							*		Miss
11				*					Miss(page is replaced with 11)
4					*				Miss
11									Hit
8	*								Miss (0 is replaced with 8)
0	*								Miss(8 is replaced with 0)

Design#2: As this is two associative cache, we will be having 4 blocks only. This also gives 7 misses (see the following table). Therefore, miss penalty with this is $10 \times 7 = 70$ cycles.

Memory Reference/Block No	0	1	2	3	
0	*				Miss
3				*	Miss
14			*		Miss
11				*	Miss(page 3 is replaced with 11)
4	*				Miss(page 0 is replaced with 4)
11				*	Hit
8	*				Miss (0 is replaced with 8)
0	*				Miss(8 is replaced with 0)

114. A certain computer has a TLB cache, a one-level physically-addressed data cache, DRAM, and a disk backing store for virtual memory. The processor loads the instruction below and then begins to execute it.

lw R3, 0(R4)

This indicates that the computer should access the virtual address currently stored in register 4 and load that address's contents into register 3. Which of the following is true about what might happen while executing this instruction?

A. If a TLB miss occurs, then a page fault definitely occurs as well.
B. If a data cache miss occurs, then a page fault definitely occurs as well.
C. No more than one data cache miss can occur.
D. No more than one page fault can occur.
E. If a page fault occurs, then a data cache miss definitely does not occur as well.

115. Suppose that a direct-mapped cache has 2^9 cache lines, with 2^4 bytes per cache line. It caches items of a byte-addressable memory space of 2^{29} bytes.

How many bits of space will be required for storing tags?

(Do not include bits for validity or other flags; only consider the cost of tags themselves.)

A. 2^8 bits
B. 2^{11} bits
C. 2^{13} bits
D. 2^{25} bits
E. 2^{32} bits

Explanation: Of the 29 bits of the address space, 9 indicate the cache line index, and 4 indicate the offset within the cache line. That leaves $16 = 2^4$ bits for the

tag. There are 2^9 tag entries, for a total of $2^{9+4} = 2^{13}$ bits.

116. College campuses have wireless ____ on campus that allow students to use their PCs to connect wirelessly to the college network and the Internet from anywhere on campus.
A. Hotspots C. Com spots
B. Hotpoints D. Com points

117. Small programs called ____ are used to communicate with peripheral devices, such as monitors, printers, and scanners.
A. Interfaces C. Drivers
B. Utilities D. Managers

118. A ____ is an area in RAM or on the hard drive designated to hold input and output on their way in or out of the system.
A. Segment C. Page
B. Buffer D. Sector

119. A ____ consists of a series of screens that prompt the user for the necessary information and then creates a particular type of document based on the users input.
A. Worm C. Trojan horse
B. Wizard D. Phish

120. A strategy for speeding up hard drive performance is ____.
A. disk caching C. disk backing up
B. disk tracking D. disk Wiring

121. _____ is responsible for starting I/O operation on a device and processing the completion of an I/O request.
A. OS B. Device driver
C. Read system call D. None

122. Major number is in _____
A. OS/2 B. Windows
C. DOS D. Unix

123. Merging log file & master file is used in _____
A. Unix file system B. FAT system
C. Pile file system D. None

124. Find odd-one out
A. Chmod() B. Umask C. Stat() D. Chown

125. Find odd-one out
A. Chmod B. Umask C. Chown D. Umask()

126. Binary name of a fie in unix is ___
A. Any characters length
B. Major & minor number pair
C. i-node number
D. None

127. PATH is _____
A. Route B. Path name of a file
C. Environment variable D. None

128. Find odd-one out
A. / B. c: C. A: D. /bin

129. Permissions to specific users can be granted in ____
A. DOS B. Linux C. Win NT D. None

130. External fragmentation _____
A. Solved through compaction
B. Contiguous file allocation
C. Chained allocation
D. Indexed allocation

131. Disk allocation table
A. FAT B. Open file table
C. bit map D. None

132. Bit table size on a 16 Gbyte disk with 4 Kbyte block is ___
A. 16 KB B. 4MB C. 2 GB D. none

133. Push-down stack ___
A. Round-robin
B. Free data bloacks management
C. Indexing
D. None

134. Free block list in memory is maintained as
A. Push-down stack B. FIFO queue
C. Both D. None

135. File allocation is done ____
A. Sector basis B. Cluster basis
C. Block basis D. Both B & C

136. I-node contains ___ bytes of address information
A. 64 B. 39
C. both a & b D. None

137. ____ is the largest single file size on a unix file system with 1 KB block size and 4 byte block addresses
A. 128 GB B. 65 MB C. 16 GB D. None

138. Critical file system data is in _____
A. FAT
B. I-node based file system
C. NTFS
D. None

139. _____ can extend multiple disks
A. Partition B. Volume
C. Cluster D. None

140. Maximum volume size in NTFS ____ bytes
A. 2^{32} B. 2^{16} C. 2^{64} D. None

141. In an NTFS system on a disk with partition size > 32 GB, a file of 3200 bytes occupies ____ cluster (sector size is 512 bytes).
A. 7 B. 2 C. 1 D. None

142. Max file size supported by NTFS is ____
A. 2^{32} clusters B. 2^{48} clusters
C. Both A & B D. None

143. NTFS uses _____
A. Super block B. Bit map
C. Both D. None

144. Lazy write/lazy commit employed in ____
A. ext2 B. DOS C. NTFS D. FAT

145. Busy wait is in _____ I/O
A. Interrupt driven B. Programmed I/O
C. DMA D. None

146. Single process dead-lock can be controlled ____
A. Postphoning I/O
B. Re-ordering
C. By making sure that user memory involved in I/O locked into main memory
D. None

147. Find odd-man out
A. Tapes B. Disks C. Terminal D. None

148. Reading ahead employs
A. Single buffer B. Double buffer
C. Circular buffer D. None

149. Line buffered
A. printf B. read C. fread D. None

150. Buffer swapping employs
A. System buffer B. Single buffer
C. Double buffer D. None

151. scroll-mode terminals
A. single buffering B. line buffering
C. double buffering D. None

152. Byte-at-a-time ____ is required
A. Single buffer B. Double buffering
C. Circular buffer D. None

153. In RAID data is
A. In a large disk B. Distributed in the disks
C. Double buffered D. None

154. Redundancy in RAID 0 is ___
A. Two B. One
C. No redundancy D. None

155. Buffer cache in unix is ____
A. disk cache B. double buffer
C. memory cache D. None

156. To copy buffer cache to the process I/O are through DMA ____ required
A. Processor cycles B. Bus cycle
C. Both A & B D. None

157. RAID 4 uses ____ storage
A. bit level B. byte level C. disk level D. None

158. In RAID 6 array if user data requires N disks then it really contains ___ disks
A. 2 B. N C. N+2 D. None

159. In unix devices are
A. files
B. major & minor numbers
C. both A & B D. None

160. Windows 2000 supports
A. HW RAID B. SW RAID
C. both A & B D. None

161. Floppy uses ____ FAT
A. 24-bit B. 16-bit
C. 32 bit D. None

162. In FAT file system directory entry contains filenames whereas in Unix directory entry will not contain file names.
A. Yes B. No C. Probably D. None

163. In FAT, a files last cluster entry contains ___
A. 0xfe B. 0xff C. 0xff8 D. None

164. ROM monitor does ___
A. reads a disk block
B. transfers control to the disk block read
C. Both A & B D. None

165. Berkley fast file system allows ___ block sizes
A. fixed B. variable C. two D. None

166. Find odd-one out
A. /proc B. namefs C. ufs D. ffs

167. vnode
A. FAT B. NTFS C. vfs D. NFS

168. OS/2 supports
A. FAT B. HPFS
C. both A & B D. None

169. OS/2 supports
A. filenames upto 254 characters
B. case sensitive naming
C. Both A & B
D. None

170. Meta-data of file in OS/2
A. Fixed size B. Variable
C. User can add D. None

171. Find odd-one out in terms of FAT
A. DOS B. Windows
C. Mac OS D. None

172. Fork
A. Unix system call B. Mac OS's file data area
C. both A & B D. None

173. Stackable file system
A. DOS B. Unix C. NTFS D. BSD

174. Find Odd-one out in terms of NT system
A. DOS B. HPFS C. YFS D. NTFS

175. Find odd-one out
A. Context switching B. Process switching
C. Bank switching D. None

176. Extended memory
A. Not accessible by 8086
B. Above 1 Mb
C. can be addressed by 80286 and above
D. All

177. In user mode
A. PSW bit 0 value is 1
B. Instructions that modify control registers are not legal and cause a program error
C. All addresses must be less than bound
D. All

178. Binary name of a file ___
A. Usually a unique number
B. Called internal name
C. Both A & B
D. External name
E. None

179. In system mode all addresses are
A. Physical addresses
B. Absolute addresses
C. Logical addresses
D. A & B

180. When a user program calls a system call
A. Mode changes to system mode
B. Interrupts are enabled
C. System call number is sued to find out code in system call interrupt vector area
D. All

181. Find odd-one out in the perspective of open file
A. Dynamic object
B. File descriptor/handle is associated
C. File internal name changes
D. Have an entry in open file table

182. OS/2 uses ___ scheduling
A. preemptive B. non-preemptive
C. round-robin D. None

183. Disk arm scheduling is to reduce effect of __ on disk access
A. horizontal latency B. rotational latency
C. both A & B D. None

184. Latency optimisation
A. seek optimisation
B. rotational optimisation
C. both A & B D. None

185. A scheduling algorithm with small variance
A. show good predictability
B. less chance for indefinite postponement
C. A & B
D. None

186. Interleaving is for
A. seek optimisation
B. latency optimisation
C. no effect under low load
D. B & C

187. Indefinite postponement
A. FCFS B. SSTF C. SCAN D. None

188. Guarantee of service is available in
A. FCFS B. SCAN C. C-SCAN D. All

189. Biased towards inner tracks
A. SSTF B. SCAN C. Both A & B D. None

190. SSTF under moderate loads
A. gives better throughput
B. low mean response times
C. higher variances
D. all

191. starvation
A. FCFS B. SSTF C. SCAN D. C-SCAN

192. For interactive systems, disk arm scheduler
A. good mean response time
B. low variation in response time
C. high variance in response times
D. A & B

193. N-step SCAN algorithm's variance in response time
A. smaller than SSTF B. SCAN
C. Both A & B D. None

194. The scheduling algorithm which optimizes both horizontal & rotational latency effects
A. Interleaving B. SCAN
C. SSTF D. Eschenbach

195. Find odd-one out
A. CSCAN B. SSTF
C. SLTF D. FCFS

196. Disk scheduling is not useful
A. batch processing system with less multiprogramming
B. batch processing system with heavy multiprogramming
C. time sharing with moderate level of multiprogramming
D. None

197. WORM
A. device B. virus like program
C. both A & B D. None

198. Direct access device
A. RAM B. ROM C. Floppy D. None

199. Direct access device
A. Accessing can be at random
B. Access times varies from file to file
C. Both A & B
D. None

200. Device with random accessing ability & with same access time for any unit data
A. cache B. FDD
C. RAM D. A & B

201. If seek time approximates latency time which of the following scheduling algorithms preferable
A. SSTF B. CSCAN
C. SSTF with SLTF D. None

202. Under extremely heavy loads _____ is needed
A. seek optimisation
B. latency optimisation
C. both A & B
D. None

203. Cylinder oriented disk scheduling is
A. FCFS B. SSTF C. SLTF D. None

204. Constant bit density is
A. same sectors/track
B. number of sectors in outer tracks more
C. in ROM
D. None

205. What is the average read/write time for a 512 byte sector with 4 ms average seek time, transfer rate 4MB/s and 7200 RPM, controller overhead is negligible and queing delay is also negligible
A. 4 ms B. 7 ms C. 8.3 ms D. 8.15 ms

206. Disk stripping means
A. RAID
B. Parts of the file is split on separate drives
C. Disk partitioning
D. None

207. Windows NT pipes are
A. uni directional B. bi-directional
C. both A & B D. None

208. Namedpipe is
A. application level communication construct
B. consists both server & client
C. bi-directional
D. All

209. Pipe is
A. |
B. parent's stdout is made child's stdin
C. IPC channel
D. All

210. Anonymous pipe
A. Supported in win NT
B. Same workstation
C. Unix doesn't support
D. All

211. Namedpipe
A. Only server can create
B. Mknod() is used in unix to get created
C. Supported in win NT
D. All

212. Namedpipe
A. frees application writes from low-level details
B. bi-directional
C. both server & client can read & write
D. All

213. In real mode
A. all memory addresses addressable by a process are addressable by all programs
B. 8086/8088 processors always runs
C. traditional MSDOS device drivers, TSR's run in this mode
D. All

214. Protected mode
A. Windows/ Win NT
B. After switching to protected mode protected mode applications are run
C. DPMI is used to run some DOS application in protected mode
D. All

215. Socket verses namedpipe
A. transport and application level things respectively
B. both are bi-directional
C. namedpipe runs on top of socket
D. All

216. File descriptor
A. an integer
B. inheritable
C. row index of descriptor table
D. All

217. Open file table
A. in user space B. per process
C. kernel space D. None

218. Metadata of a file
A. I-node B. FAT C. Both D. None

219. Hard link
A. When created link count increases
B. File I-node is same as original file I-node
C. Only done on same partition
D. All

220. Symbolic link
A. Link count of I-node will not change
B. File I-node will different from original
C. Can be created for a directory
D. All

221. An OS
A. Mediator between user and HW
B. Resource Manager
C. Implementer of virtual computer
D. All

222. Disk sharing can be done by
A. Time-multiplexing I/O channels
B. Space multiplexing disks
C. Both A & B
D. None

223. Virtual printer
A. spooling
B. printer file
C. creates illusion of multiple printers
D. All

224. System call may
A. generate interrupt
B. make OS to gain control of processor
C. not require service routine address
D. All

225. Find odd man out
A. pipe() B. popen()
C. msgget() D. fork()
E. All

226. Find odd one out
A. Read B. Write C. Stat D. Fread

227. What will be the result of the following program

```
#include<stdio.h>
  main()
   {
    fork();
     fork();
     printf("Hello\n");
   }
```

A. two Hello's B. three Hello's
C. four Hello's D. Unpredictable

228. Little language
A. low level B. high level
C. scripting
D. provide interactive access to a collection of functions related to a task

229. Exit status of a unix command if successful
A. positive number B. zero
C. any number D. None

230. File descriptor & file stream pointers are
A. same B. different
C. supported in unix only
D. Both B & C

231. File descriptor
A. meaningful in unix only
B. is an integer
C. not portable like FILE
D. All

232. Memory mapping of a file
A. to share a part of a file by multiple process
B. lets a part of virtual address space to be associated with a section of a file
C. writes by any of the process on mapped file's part/page can be seen by all processes
D. All

233. When we type xyz command in DOS then _____ sequence files are checked to load into memory
A. xyz.com, xyz.exe, and xyz.bat
B. xyz.exe, xyz.bat, and xyz.bat
C. xyz.bat, xyz.exe, and xyz.com
D. xyz.com, xyz.exe, and xyz

234. Disk internal fragmentation
A. depends on block size
B. depends on machine word size
C. depends on degree of multiprogramming
D. None

235. Disk internal fragmentation
A. increases with block size
B. is zero if file is exact multiple of sector size
C. A & B D. None

236. Directory structure in
A. DOS is tree
B. Unix is acyclic graph
C. Tree structured prohibits file sharing
D. All

237. Desktop file is supported in
A. DOS B. Unix C. Macintosh D. None

238. Compressed file systems
A. double drive
B. superstore drive
C. any file is stored in compressed form
D. all

239. ____ called as indirect pointers
A. links B. addresses
C. memory addresses D. None

240. Dangling links occurs
A. symbolic links
B. if the real file is deleted for which symbolic links are existing
C. A & B
D. with hard links

241. Dangling pointers to now-non existing file occurs
A. symbolic links
B. if the real file is deleted for which symbolic links are existing
C. A & B
D. with hard links

242. file reference count is
A. in inode
B. incremented/decremented whenever hardlink is created or deleted for a file
C. A & B
D. None

243. Device driver
A. set of functions B. a program
C. developed exclusively using system calls
D. None

244. Swap space
A. can be a file
B. can be a partition
C. can be on multiple disks
D. All

245. If swap file is used
A. then adding extra space is easy
B. swapping becomes inefficient
C. external fragmentation may increases swap space accessing
D. All

246. Swap partition
A. will not have any file or directory structure
B. may employ contiguous block allocation policy
C. very efficient
D. All

247. If swap partition is used
A. Internal fragmentation increases
B. Swap partition can be shared by multiple OS's one at a time
C. Management of blocks is easy
D. All

248. Normalised turnaround time
A. waiting plus service time
B. elapsed time
C. ratio of turnaround time to service time
D. None

249. Normalised turnaround time
A. min possible value is 1
B. for a good scheduler this parameter to be smaller
C. increase in values of this corresponds to decrease in levels of service
D. All

250. PC supports ____ no of interrupts
A. 15 B. 16 C. 256 D. None

251. PC serial port I/O address is ____
A. 0x3f8 B. 0x2f8 C. A & B D. 0x2d

252. Interrupt number assigned to Hard Disk in PC is ___
A. 11 B. 12 C. 13 D. 14

253. Interrupts mask to enable clock, key board and printer
A. 01111100 B. 11110001
C. 10110111 D. None

254. Device independence
A. is for uniform naming
B. requires files to be modified differently for each device

C. is for common device driver development
D. None

255. A device driver handles
A. one device
B. one class of closely related devices
C. A & B
D. None

256. UART is needed
A. for controlling terminals
B. coverts serial data to character
C. converts character data to serial
D. All

257. Video RAM address in PC
A. 0xB0000 B. 0xB8000
C. both A & B D. None

258. Memory mapped terminals are
A. not interrupt driven B. interrupts driven
C. processor driven D. None

259. Find out correct one
A. L1 cache will be in microprocessor
B. L2 cache will be in outside microprocessor
C. A & B
D. None

260. Cache memory may be constructed through
A. DRAM B. SRAM C. ROM D. None

261. Cache controller
A. copies data from physical memory to cache memory
B. copies data from physical memory to RAM disk
C. copies data from user space to kernel space
D. None

262. Cache memory
A. is needed with fast computers such that computers are not required to wait for RAM
B. improves system performance
C. A & B
D. None

263. Find odd-one out
A. Valid bit B. Used bit
C. Dirty bit D. Presence bit

264. An associative cache for a 64K machine with 8 bit memory word is used then cache word length is
A. 32 bits B. 16 bits C. 25 D. None

265. Compiler generally allocate elements of an array to contiguous memory location. This may
A. aid set associative caching
B. give locality of reference
C. A & B
D. None

266. A computer with 1K cache, 64K RAM with 8 bit word is employing direct mapping, then the size of cache word
A. 10 bits B. 16 bits C. 15 bits D. None

267. Harward architecture
A. separate caches for data & instructions
B. same caches for data & instructions
C. multiple processors
D. None

268. An associative cache for a 64K machine with 8 bit memory word is used then cache word length is
A. 32 bits B. 16 bits C. 24 bits D. None

269. An associative cache for a 64K machine with 8 bit memory word is used. A block size of 4 is used, then cache memory word size is
A. 32 bits B. 46 bits C. 48 bits D. None

270. A two-way set associative memory with 1K is used then the number of locations are
A. 1024 B. 16 C. 512 D. None

271. For I=1 to 1000 do
x=x+I
Assume x is stored in cache currently. Then __ is preferred
A. write-through B. write-back
C. both D. None

272. Intel's Itanium processors L1 cache is
A. Single cache
B. Split cache
C. Split cache with 4-way set associative
D. None

273. Intel's Itanium processor has
A. L1 cache in processor B. L2 cache in processor
C. A & B D. None

274. Intel's Itanium processor L2 cache is
A. split B. unified
C. unified six-way set associative
D. split six-way set associative

275. Itanium supports
A. L1 cache B. L2 cache
C. L3 cache D. L4 cache
E. All

276. A CPU with 32 bit virtual address space, 16K page size will have ___ entries in page table

A. 16536 B. 65535 C. 262144 D. None

277. Alpha microprocessors supports ___ page tables
A. single level B. 4 level
C. three level D. None

278. Intel Pentium processor
A. 16K L1 cache
B. 8K Instruction cache (a part of L1 cache)
C. 8K data cache (a part of L1 cache)
D. All

279. The code part of L1 cache in Pentium processor uses _
A. Direct mapping
B. 2-way set associative
C. 4-way set associative
D. None

280. Pentium's code cache uses _____ replacement strategy
A. FIFO B. LRU
C. Pseudo LRU D. NRU
E. None

281. Data cache in Pentium processor uses line size of
A. 32 bits B. 32 bytes
C. 20 bits D. None

282. Pentium processor the code cache as a TLB of ___ entries
A. 16 B. 32
C. 64 D. None

283. In Pentium, CPU
A. can not write into code cache
B. can write into data cache
C. A & B
D. can not write into data cache
E. All

284. Pentium processor has
A. L1 cache inside microprocessor
B. L2 cache outside
C. A & B
D. All

285. L2 cache size in Pentium processor
A. 256 KB B. 128 KB
C. 1 MB D. None

286. L2 cache in Pentium processor is
A. Unified B. Split C. 4 way D. None

287. L2 cache in Pentium processor is __
A. 2 way set associative
B. 4-way set associative
C. direct mapping
D. All

288. In Pentium processor virtual memory is handled by
A. Pentium processor B. Windows NT
C. Both A & B D. None

289. Associative memory (CAM) contains
A. data register B. mask register
C. match register D. output register
E. All

290. Number of matches in associative memory which is used for cache/virtual memory is ___
A. one B. many C. two D. None

291. A processor with 64K of 8bit bytes of data is having 1 K cache then index bits are
A. 16 B. 6 C. 19 D. 3
E. None

292. Find the wrong one
A. In associative cache any word of physical memory can occupy any word of cache
B. In direct mapped cache any word of physical memory can be mapped to any specific location
C. In direct mapping block size is 1
D. Set associative memory uses CAM

293. LRU cache replacement is used in
A. associative cache B. direct mapped cache
C. set-associative D. None

294. Count value is available in
A. associative cache B. direct mapped cache
C. set associative D. None

295. Count value in set-associative cache
A. is larger then older the data
B. relative order of access
C. is zero for most recently accessed one
D. All

296. TLB does not contain
A. count B. dirty bit
C. every entry of page table
D. All

297. In x86 architecture, segment should start at
A. any address B. at fixed address
C. at an address whose low order bits are 0000
D. All

298. x86 supports
A. segment table B. segments
C. both D. None

299. A CPU can address 1Mx16, cache of 8Kx16 is used through fully associative cache then number of bits per location
A. 16 B. 33 C. 32 D. None

300. Find odd-one out
A. Data bus B. Control bus
C. Address bus D. I/O bus

301. Internal interrupts
A. Occur entirely within the CPU
B. Timer
C. To handle exceptions
D. A, B & C
E. Software interrupts

302. When interrupt occurs
A. Current instruction is completed
B. Context switching may take place
C. Service routine is executed
D. All

303. In burst mode, DMA transfers
A. entire block B. one byte at a time
C. one record at a time D. None

304. _____ mode is recommended to transfer files through DMA
A. input mode B. byte mode
C. burst mode D. character mode
E. transparent mode

305. A controller that must monitor data in real time or near realtime may prefer ___ mode transfer with DMA
A. transparent mode B. burst mode
C. cycle stealing mode D. None

306. In ___ mode DMA controller interleaves instructions and data transfers
A. transparent B. burst
C. cycle stealing D. block
E. None

307. ____ mode DMA controller transfers data when CPU is not using
A. transparent B. block transfer
C. cycle stealing D. None

308. Block address trace
A. required in FIFO page replacement
B. required in optimal page replacement
C. is sequence of virtual block address
D. B & C
E. None

309. Not a stack algorithm
A. LRU B. NRU
C. OPTIMAL D. FIFO

310. Look-through cache
A. separate local bus to cache
B. faster than look-aside
C. I/O operations and caching can be done concurrently
D. All

311. When miss occurs then ___ procedure can respond quicly to CPU
A. look-aside B. look-through
C. can't say D. None

312. Cache coherence
A. stop stale data usage
B. may be found in multiprocessors
C. even found in uniprocessor systems
D. All E. None

313. ___ guarantees no stale information in memory
A. write-through B. write-back
C. copy-back D. None

314. Preferable hit ratio for having good system performance is
A. 0.9 B. almost one
C. about 100 D. None

315. If hit ratio reduces from 99% to 95%
A. access time increases
B. cache hit falls down
C. access time increases by about 23%
D. None

316. Find odd-one out
A. Programmers (assembly) can access all registers in a processor
B. Programmer (assembly) can not access all registers in a processor
C. Some registers are not part of instruction set
D. None

317. Exceptions or traps
A. software interrupts
B. hardware interrupts
C. are triggered when valid instructions perform invalid operation
D. None

318. Usefull addressing mode for short jumps
A. direct B. indirect
C. relative D. None

319. Instructions are orthogonal
A. if they don't overlap
B. if they don't perform same function

C. A & B
D. are perpendicular
E. All

320. Intel Itanium contains
A. 128 general purpose registers for integer data
B. 128 general purpose registers for float point data
C. A & B
D. None

321. In 8085
A. Accumulator is A
B. Register A receives 8-bit arithmetic or logical operations
C. Has 6 general purpose registers
D. Has 16-bit stack pointer register
E. All

322. Interrupts that are masked
A. are lost B. are not lost
C. are delayed D. B & C

323. Relocation register
A. is under the control of OS
B. is not available directly to the user program
C. is involved automatically when a program refers any memory location
D. All

324. For the following code fragment, 2 stage pipeline is proposed; 1st stage for multiplication (10 ns) and 2nd stage for addition (10 ns) is required. Then how much time it takes to complete.

for I=1 to 100 do A[I]=B[I]*C[I]+D[I]

A. 2000 ns B. 1010 ns
C. 1020 ns D. 2010 ns

325. Repeat the above assuming latching in pipeline require 2 ns and fetching time is ignored.
A. 2000 ns B. 1012 ns C. 1212 ns D. None

326. NaN
A. Not a Number
B. Represents illegal operation such as infinity % infinity
C. Is assigned to unitialised variable
D. All

327. Sticky bit
A. can be used for a executable file in unix for performance reasons
B. when set to a unix directory any one can write into it
C. used in rounding algorithms
D. All

328. Gap
A. is a data structure which keeps characters in a large array with a gap in the middle.
B. is the difference between two adjacent values in floating point representations.
C. A & B
D. None

329. Reentrant programs
A. code that can not be changed while in use
B. code can be used or shared by several processes simultaneously
C. code is not serially reusable
D. may not contain static or locale data
E. All

330. Reentrancy
A. is a property of a program
B. is a characteristic of a process
C. makes multiple copies easier
D. is achieved by avoiding global or static variable
E. All

331. Malloc, new etc memory Allocators use
A. per-process memory
B. may call OS only if per-process memory is over or exhausted
C. A & B
D. None

332. Object module produced by a compiler after linking may not contain
A. symbol table B. header information
C. text D. data segment
E. environment variables

333. Find odd man out
A. EXE B. COFF C. ELF D. DLL

334. linker responsibility doesn't include
A. Combining object modules into a load module
B. Relocate object modules when they are joined
C. Link object modules when they are joined
D. Search libraries for external references which are not defined in object modules
E. Attach command line information to load module

335.
```
#define<stdio.h>
int a[10];
int size=100;
static int b=15;
void main()
```

```
{
int sum=0;
}
```

In the above program which is stack variable?.

A. a B. bb C. Size D. Sum

E. None

336. Find odd-one out

A. Symbol table in load module is necessary while running.

B. Symbol table in load module is not needed while running.

C. Strip command can be used in unix to remove symbol table from a load module.

D. None

337. Find odd-one out

A. Object modules are compiled.

B. Load modules are compiled, linked, and are ready for execution.

C. Object modules contain symbol table.

D. Load module contains single symbol table though it is created by joining many object modules.

E. None

338. Find odd-one out

A. Load time dynamic linking

B. Run time dynamic linking

C. Static linking D. A & B

E. Dynamic linking

339. Find odd-man out

A. Process B. Interpretation

C. File descriptor D. Compilation

340. Find odd-one out

A. Program B. I-node number

C. Compilation D. Debugging

341. In ASCII first 32 characters are

A. Control codes

B. Nonprinting characters

C. Characters used in screen & printer control

D. Used in serial communication

E. All

342. In USB ___ number of devices can be connected on a single port

A. 1 B. 2 C. 127 D. None

343. The serial ports of most PC's can transmit data upto ____ bps

A. 9600 B. 2400 C. 1200 D. 115200

E. None

344. RS-422 serial standard is used in

A. PC B. Pentium

C. AppleMac D. None

345. RS-422 standard

A. Uses differential voltage to send data

B. Uses immune to electrical noise

C. Supports longer distances

D. Uses HSKO/HSKi signals

E. All

346. CRC field length in USB data packet

A. 8 bits B. 16 bits C. 1 bit D. None

347. CRC field length in USB token packet

A. 8 bits B. 5 bits C. 1 bit D. None

348. USB supports ___ packets

A. Token B. Data

C. Handshake D. All

349. Find odd-out one

A. USB data packet does not contain device address

B. PID packet is 8 bits

C. USB supports one device

D. None

350. Total bits required for direct mapped cache with 64 KB of data and one-word blocks if addresses are 32 bits is ___

A. 16 KB B. 98KB C. 100KB D. None

351. RISC

A. To simplify compilers

B. To improve performance

C. Will have fewer instructions

D. Tend to emphasise on registers

E. All

352. RISC architectures

A. Executes one instruction per cycle

B. Uses register-to-register operands

C. Simple address modes

D. Supports simple instruction formats

E. All

353. If the instruction length is aligned on boundaries, then

A. Fetching is optimized

B. Single instruction does not cross page boundaries

C. A & B D. None

354. If the instruction length is fixed with fixed field locations, then

A. Control unit becomes less complex

B. Opcode decoding, register operand accessing can be done concurrently

C. A & B D. None

355. Machine cycle in RISC can be defined as

A. Fetching two operands from registers

B. Perform ALU operation

C. A & B

D. None

356. Kerberos

A. Disk B. Network protocols

C. Processor D. None

357. Kerberos

A. Network protocols

B. Does not assumes both side machines are secure

C. Employs authentication server

D. All

358. In Pentium the data cache as a TLB of ____ entries

A. 16 B. 32

C. 64 for page size of 4KB

d. None

359. File descriptor

A. A integer >= 0

B. Of stdin is 0

C. Inheritable to chaild

D. All

360. A paging system is employing HW cache as TLB of 20ns access time (search time) and physical memory access time is 100 ns. It is observed that the hit ratio is 80%. The effective memory access time is

A. 120ns B. 220ns C. 140ns D. None

361. A paging system is employing HW cache as TLB of 20ns access time (search time) and physical memory access time is 100 ns. It is observed that the hit ratio is 98%. The effective memory access time is

A. 120ns B. 122ns

C. 220ns D. None

362. In a paging system with TLB it is observed that for every 10 memory accesses 9 accesses are successfully found in TLB. Find out effective access time if memory access is 200 ms and cache access is 10ms while finding page base address

A. 20ms B. 30ms C. 40ms D. None

363. A given computer has a maximum of 3 page frames which are allocated to a process. Page reference string (FIFO order)is 2 3 2 1 5 2 4 5 3 2 5 2, then how many page faults occurred if LRU is employed and initially all the three frames are free.

A. 10 B. 11

C. 13 D. None of the above

364. From the following data find out when process 3 is completed if shortest job next is employed.

Process	Arrival Time	Expected CPU Time
1	0	14
2	3	12
3	5	7
4	7	4
5	19	7

A. 7 B. 21 C. 25 D. None

365. Repeat for the above data for Round Robin algorithm with time slice value is 5. Assume if a quantum expires at the same time of a process arrival then the process joins the queue first. Also assume over head for context switching and other administration responsibilities is negligible.

A. 24 B. 31 C. 44 D. None

366. Given the following data and for time slice value of 2 find out turn around time for process 3.

Process	Arrival	Service time	Priority
1	1	8	2
2	2	2	4
3	3	1	3
4	4	2	4
5	5	5	1

A. 9 B. 3 C. 10 D. None

367. Calculate for the above data average wait time if FIFO policy is employed

A. 5 B. 6 C. 6.4 D. None

368. Calculate average wait time for the data in problem 46 if round robin with priorities are employed

A. 4 B. 5 C. 6.4 D. None

369. When very few disk accesses are occurring every disk scheduling algorithm tends to approximate which specific disk scheduling algorithm?

A. FCFS B. SCAN C. C-SCAN D. SSTF

370. When only one request is available in the queue at any time then all algorithms behave similar to

A. FCFS B. SCAN C. C-SCAN D. SSTF

371. Given that it takes 1 ms to travel from one track to the next and that the arm is originally positioned at track 15 moving towards the low-numbered tracks then time required to serve requests 4, 40, 11, 35, 7, and 14 is

A. 40ms B. 47ms C. 30ms D. None

372. In single-user, single tasking environment the following disk scheduling algorithm is adequate

A. FCFS B. SCAN C. C-SCAN D. SSTF

373. When a process is created
A. A free PCB is obtained
B. PCS is initialised
C. Obtains necessary resources such as memory, I/O devices
D. All

374. Spreading disk activity among multiple disks & controllers
A. Makes possible to transfer multiple blocks
B. Improves system response time
C. A & B
D. None

375. Which of the following need not be saved during context switching.
A. General purpose registers
B. TLB's
C. PC D. All

376. Suppose the time to serve a page fault is on an average 10ms, memory access is 1μs, hit ratio is 99.99% then average memory access time
A. 1.9999ms B. 1ms
C. 9.999μs D. 1.9999μs

377. Which of the following disk scheduling algorithms is likely to give better throughput?
A. FCFS B. SCAN C. C-SCAN D. SSTF

378. If an instruction takes 'i' μs, page fault μs, the effective instruction time on an the average if a page fault occurs every 'k' instructions
A. i+j/k B. i+j*k C. (i+j)/k D. (i+j)*k

379. Locality of reference implies that the page reference being made a process
A. Will always be the page referred earlier
B. Is likely to be to one of the pages used in the last few page reference
C. Will always be to one of the pages existing in memory
D. Will always lead to a page fault

380. Thrashing
A. Reduces page I/O
B. Implies excessive page I/O
C. Decrease the degree of multi programming
D. Improve system performance

381. A Imbyte memory is managed using variable partition with no compaction. It currently has two partitions 200Kb, and 260 Kb. The samlles allocation request in K bytes that could be denied is for
A. 151 B. 181 C. 231 D. 541

382. In which of the following page replacement strategies, Belody anomaly may arise
A. FIFO B. LRU C. LIFO D. A & C

383. The address sequence generated by tracing a particular program execution in a pure demand paging with 100 records per page with 1 free main memory frame is recorded as follows. What is the number of page faults?
Page reference sequence: 1,0,2,3,1,3,4
A. 13 B. 5 C. 7 D. None

384. Which of the following statements is false?
A. VM implements the translation of a programs address space to physical address space
B. VM allows each program to exceed the size of the primary memory
C. VM increases degree of multiprogramming
D. VM reduces the context switching overhead

385. If RAM is 64MB with 32 bit virtual address space and page size is 4KB then approximate size of the page table
A. 16MB B. 8MB C. 2MB D. 24MB

386. A CPU has 32-bit memory address, 256KB cache. The cache is organised as a 4-way set associative with block size of 16 bytes then the number of sets in the cache
A. 64K B. 128K C. 32K D. None

387. How long does it take to load a 64K program from disk whose average seek time is 30ms, whose rotation time is 20ms and track hold 32K, Page size or block size is 2K? Assume the blocks are spread randomly on the disk.
A. 640 ms B. 100 ms
C. Information is not adequate
D. None

388. Given a machine with only a stack whose top can be output and on which POP and PUSH are allowed. Which of the following strings can be sorted in ascending order?
A. 4312 B. 3421 C. 2134 D. 1243
E. 3142

389. What is the time about C, the cost of replacement policy which is given as the number of page faults for a particular reference string?
A. C(LRU) always < C(FIFO)
B. C(LRU) always > C(FIFO)
C. C(LRU) always = C(FIFO)
D. Can't be answered

390. A system has 3 page frames in main memory and uses LRU replacement policy with the follwing reference string. What is the state of main memory (the pages existing) after the 5th page fault?

1223413121

A. 321 B. 124 C. 234 D. None

391. In a memory scheme, the address of a location is specified by a page address and a displacement within a page, in hexadecimal.

# of pages = 16.

# of words per page = 256.

The address of the 11th page, 94th word is :

A. B5E B. A5D C. 5EB D. E9C

392. A system has 4 tape drives. The following set of jobs have demand for tape drives as shown:

W → 1 X → 2 Y → 3 Z → 4

Which of the following combinations may give rise to deadlock?

1) Y 2) WY 3) WYZ

A. 1 B. 2 C. 3 D. 2 & 3

E. None

393. The following processes exist in the process queue. Completion times required are

P1:2 P2:3 P3:4 P4:1 P5:6

What is the mean completion time?

A. 5.4 B. 7.2 C. 12 D. None

394. The following sequence of operations is valid:

1. Increment address pointer and enable read.
2. Enable write and decrement address pointer.

What does this implement?

A. Stack B. Queue C. List D. Deque

395. 5 processes are in a queue. The times for completion of each are 6, 3, 4, 3 and 2 respectively. Find the minimum average turn around time

A. 18/5 B. 9 C. 62/5 D. 63/5

E. 18

396. A disk has the following parameters : Tracks-35 ,Sectors-10, Data transfer rate- 250,000 bits per sec ,speed of rotation-300 rev/min .What is the total storage in bytes?

A. 300ms B. 500ms

C. 1750000 × 6250 D. None

397. Interrupt is used for a byte transfer request from the above disk. If the interrupt overhead is 10 microsecs, 4 byte transfer time is 8 microsecs, how much time is available during byte transfers for other work?

A. 13ms B. 14micro seconds

C. 33ms D. None

398. A stack is initially empty. A program is executed. It is either a PUSH or a POP instruction. Which of the following cannot represent the stack's contents after HLT?

program: I1, I2, I3, I4, I5, I6, HLT

A. <2> B. <3> C. <5> D. <1,2,5>

E. <3,4,5>

399. Two computers A,B exist with machine languages Ma and Mb; Assembly languages La and Lb; Cross Assembler for B running A is

A. La to Ma B. La to Mb

C. Lb to Ma+La D. La to Ma+Mb

E. Lb to Ma

400. In a system using buffered I/O, which of the following are needed for ensuring a consistesnt state on recovery from a crash:

I. Allocate more main memory for buffers

II. Duplication of buffers in memory

III. Atomic write of multiple buffers

A. I B. III C. II, III D. I,II

E. I, II, III

401. Which of the following are true:

I. Set associative cache is cheaper than a direct mapped cache.

II. Set associative cache has higher hit ratio than a Fully associative cache

III. Set associative cache has higher hit ratio than a direct mapped cache

A. I B. I,III C. III D. I,II

E. I,II,III

402. Policy 1: Allocate a file on disk contiguously

Policy 2: Allocate a file on arbitrary blocks on disk

Which of the following is/are the advantage of Policy 1:

I. It is good for reading large files sequentially

II. Files are easily expandable in this method

III. Random access is faster in this method

A. I B. I,III C. II D. III

E. I,II,III

403. Which of the following is false about RISC architectures?

A. All arithmetic operations deal with registers

B. The only instructions accessing memory are load and store

C. The compiled code is shorter for RISC than for CISC

D. None

404. Which of the following is not done when an interrupt occurs?

A. Save the starting address of the executing procedure
B. Save the address of the current instruction
C. Detect the cause of the interrupt
D. Save the values of the registers
E. Make a call to the kernel

405. 2 level cache. A main memory, 2 caches, 4 processors connected to each cache. 1 level hit ratio is 0.95, 2nd level hit ratio is half of the remaining requests.

If accesing the 1st level cache takes 1 cycle. The 2^{nd} level cache takes 10 cycles and the main memory takes 100 cycles. What are the average number of cycles taken to access an element.

A. 2 B. 4 C. 6 D. 8
E. 100

406. Two processes have serial execution if instructions are executed in some order and instructions of a particular process are in order. Two processes share variable r1 and r2 and have local variable x, y r1 = r2 = 0

Process 1	Process 2
x = 1	y = 1
r1 = y	r2 = x

which of the following is not possible after serial execution of above process?

A. r1 = 0, r2 = 0 B. r1 = 1, r2 = 0
C. r1 = 1, r2 = 1 D. r1 = 0, r2 = 1
E. All of the above are possible

407. An overlay is

A. A part of OS
B. A single memory location
C. Swapping
D. Overloading the system with many user files

408. A computer system has 6 tapes with n processes competing for them. Each process needs 3 tapes. The maximum value of n for which the syetm is guaranteed to be deadlock free is

A. 2 B. 3 C. 4 D. 1

409. In round robin algorithm if time quantum is increased then the average turn around time

A. Increases B. Decreases
C. Remains constant D. None

410. In a paged memory page hit ratio is 0.35, swap disk access time is 100ns, the time for accessing page in RAM is 10ns. The average time required to access a page is

A. 3ns B. 68ns C. 68.5ns D. 78.5ns

411. Memory protection is done by

A. Processor and HW B. The OS
C. Compiler D. The user program

412. The difference between the time you get results and to the time of submission is

A. Elapsed B. Turnaround
C. System D. None

413. Time command in Unix gives

A. User time B. System time
C. Elapsed time D. All

414. A process is said to be in ___ state if it is awaiting for an event that will never occur.

A. Safe B. Unsafe
C. Starvation D. Deadlock

415. ___ is used to know timing details of a program at clock tick level.

A. Time command B. Profiling
C. Make D. None

416. Zombie

A. State of a process B. Game
C. Virus D. None

417. PID of ___ process is 1 in unix.

A. Init B. Page daemon
C. Lp daemon D. Kernel

418. Number of bits used for PID in Unix is __.

A. 8 B. 16 C. 32 D. None

419. Distributed operating systems are

A. Based on Networks
B. Multiple computers are available
C. Designed to have max throughput
D. All

420. Array processors will have

A. OS
B. Not have OS
C. Their MCU runs under some OS
D. None

421. What is the average time required to read or write 512-byte sector for a disk with 5400 RPM with the average seek time of 12ms, transfer rate of 5MB/sec.? Assume controller overhead is 2ms and disk is idle initially.

A. 10ms B. 12ms C. 19.7ms D. 19.2ms
E. None

422. Number of IDE controllers available in normal PC motherboards are

A. 2 B. 3 C. 4 D. 1
E. None

423. Device controller
A. Will have limited instruction set
B. Does not have any stored programs
C. Contains interrupt service routines
D. A & B
E. None

424. I/O processors
A. Called as channels
B. May know how to execute more than one instruction
C. Device controller
D. All of the above
E. None

425. _____ is the terminal device controller.
A. OS B. Serial port
C. Modem D. None
E. Terminal device driver SW

426. New line command
A. Line feed in Unix B. CR in Mac
C. CR & line feed in PC D. All
E. None

427. A character terminal
A. Will have frame buffer
B. Will not have frame buffer
C. Will have character generation memory
D. None

428. In a character terminal
A. Each pixel is addressable
B. Each pixel is not addressable
C. Will contain character bit map stored
D. B & C
E. None

429. Colour map
A. Maps 8 bit colours to 24 bit colour specification
B. Also called as palette
C. Is a resource in a windowing system
D. Indicates the colours displayable at any time
E. All

430. X terminals
A. Supports graphics
B. Command oriented graphic terminal
C. Does not access to the frame buffer
D. All
E. None

431. RAID uses
A. Flat file system
B. Log structured file system
C. Compressed file system
D. None

432. RAID
A. Is to improve transfer rate
B. Is to increase reliability
C. Employs redundant disks
D. All

433. Disk performance can be improved by
A. Overlapped seeks on disks which are connected to same controller
B. Connecting disks to separate controllers
C. Storing files among multiple disks
D. All

434. In ___ system a part of a file is stored in a disk and another part is stored in another disk and vice versa.
A. FAT B. I-node based FS
C. RAID D. None

435. SCSI
A. May have separate SCSI bus
B. Controller connects directly to computer bus
C. Gives higher data rates
D. All
E. None

436. If a program is reading a file and writing modified version to another file and using double buffering then number of buffers it uses are ___.
A. 2 B. 1 C. 4 D. None

437. If a program is reading from a file and writing into a file after modifying and employing double buffering then number of outstanding disk requests are __.
A. 1 B. 2 C. 4 D. None

438. Number of outstanding disk requests in single buffering
A. 1 B. 0 C. 2 D. None

439. Greedy disk arm scheduling algorithm
A. FCFS B. Elevator C. SSTF D. None

440. All disk arm scheduling algorithms except FCFS has bias
A. Yes B. No

441. When does starvation occur in elevator algorithm?
A. Never
B. Steady stream of requests for the same cylinder in which currently head is located

C. Never occurs D. None

442. Batching
A. Is an arm scheduling algorithm
B. Can be used to stop indefinite postponement
C. Is variant of Elevator algorithm
D. None

443. Disk caching is better for
A. Reading B. Writing
C. Reading than writing D. None

444. Advantage of accessing memory through the file interface
A. Improved speed B. Flexibility
C. Trusted processes access to main memory
D. None

445. File pointers in most Unix systems
A. 4 bytes B. 32 bits
C. 4 bytes D. B & C
E. None

446. File system descriptor
A. Boot block B. Super block
C. I-node blocks D. None

447. Number of files that go into MSDOS root directory
A. 336 B. 256 C. 512 D. None

448. Mounting
A. To see multiple file systems as a single one
B. For uniform naming
C. Can not be done by users
D. All

449. Really ASCII is a 7-bit code. But most computers consider it as a 1-byte character. How they consider most significant bit?
A. Zero
B. 1 to support either symbols
C. Parity for error detection
D. All

450. I/O bus consists of
A. Data lines B. Address lines
C. Control lines D. All

451. Number of circuits in null modem
A. Null B. 2 C. 3 D. None

452. In strobe based data transfer
A. Source does not have any idea whether data is received or not
B. First strobe signal is sent then data is sent
C. To indicate end of data strobe signal is removed
D. All

453. A terminal is having transfer rate of 10 CPS. Each character is having 1 start bit, 8 data bits and two stop bits then band rate is ___.
A. 10 CPS B. 110 bps C. 110 D. None

454. Bus speed is not limited by
A. Length of bus B. Number of devices
C. Buffering D. None

455. High speed bus
A. Processor-memory B. I/O
C. Back plane D. None

456. PCI bus is __
A. Processor-memory B. I/O
C. Backplane D. None

457. Number of tapes in SCSI bus are
A. More than backplane bus
B. Less than backplane bus
C. Can not say
D. None

458. An asynchronous bus
A. Is not clocked
B. Is clocked
C. Employs handshaking
D. A & C
E. None

459. A synchronous bus with clock cycle of 50 ns and 40ns per handshake can do one transmission per clock cycle. Then its bandwidth _____. The data portion of bus is 32 bits wide and memory access time is 200 ns.
A. 10Mbps B. 11Mbps
C. 13.3MB/sec D. None

460. A asynchronous bus with clock cycle of 50 ns and 40ns per handshake can do one transmission per clock cycle. Then its bandwidth is_____. The data portion of bus is 32 bits wide and memory access time is 200 ns.
A. 10MB/sec B. 11.MB/sec
C. 13.3MB/sec D. None

461. Asynchronous bus is preferred
A. As they can be scaled
B. Can support wide variety of devices with different latencies
C. As they can support long physical distances
D. All

462. Bus bandwidth can be increased by
A. By increasing data bus width
B. Separate line
C. Block transfers D. All

463. A memory and bus system supports block access of four 32-bit words. A 64-bit synchronous bus has clocked at 200MHz, with each 64-bit transfer taking one-clock cycle, to send an address memory. Two clock cycles needed between each bus operation. A memory access time for the first four words of 200ns; each additional set of four words can be read in 20ns. Assume that a bus transfer of the most recently read data and a read of the next four words can be overlapped. The bandwidth for a read of 256 words for is ___.

A. 4MB/sec B. 71.11 MB/sec
C. 177.11MB/sec D. None

464. A memory and bus system supports block access of four 32-bit words. A 64-bit synchronous bus has clocked at 200MHz, with each 64-bit transfer taking 1-clock cycle, send an address memory. Two clock cycles needed between each bus operation. A memory access time for the first four words of 200ns; each additional set of four words can be read in 20ns. Assume that a bus transfer of the most recently read data and a read of the next four words can be overlapped. The bandwidth for a read of 256 words for is ___.

A. 4MB/sec B. 71.11MB/sec
C. 224.56MB/sec D. None

465. Split transaction protocol

A. Is used to increase effective bus bandwidth
B. Uses protocol in which bus is released while memory access takes place
C. Time to complete one transfer will take little more time
D. Complex to implement
E. All

466. PCI standard backplane bus uses

A. Daisy chain arbitration
B. Centralised, parallel arbitration
C. Distributed arbitration by self-selection
D. Distributed arbitration by collision-detection

467. Can a 100MB/sec bus not transfer 100MB data in real time?

A. Yes B. No

468. Hit time and miss penality

A. All same
B. Hit time is lower than miss penality
C. Hit time is greater than miss penality
D. Not appropriate to compare

469. Find odd-one out

A. Paging is used to exploit spatial locality
B. Caching is used to exploit temporal locality
C. A & B
D. None

470. If a cache size is M=2 ^n words, addresses is m bits long (m >n) then ___ of lower order bits of address m is adequate for direct mapping.

A. log2(M) B. log2(n) C. log2(m) D. None

471. If a cache size is M=2 ^n words, addresses is m bits long (m >n) then ___ number of tag bits adequate in direct mapping

A. log2(M) B. m-n
C. M-log2(M) D. A & B
E. None

472. ____ exploits only temporal mapping

A. Direct mapping B. Set associative
C. CAM D. None

473. Race condition

A. A can occur if two processes are running parallel to each other
B. Can occur if there exists some sort of communication across two processes such as shared memory
C. A & B
D. None

474. Serially reusable resource

A. Memory
B. Disk
C. Is the resource that can only be used by one process at a time and can be retruned soon after
D. All

475. A process can have

A. Only one critical section
B. Any number of critical sections
C. OS may put some restrictions on number of critical sections
D. None

476. Does a machine language instruction is interruptible

A. No
B. Yes
C. Commonly No. But on some machines they are interruptible
D. None

477. In producer & consumer problems buffering is needed

A. To take care of bursty producer
B. To take care of bursty consumer
C. Both A & B
D. None

478. Threads
A. Are more efficient than processes
B. Are more cheaper to create and destroy as they don not require allocation of real addresses
C. Switching is faster
D. Will have their own stack, registers
E. All

479. Find odd-one out when thread is created
A. Memory mapping will not be done
B. Open files are duplicated
C. Address translation cache is not initialised
D. None

480. Every thread contain
A. Thread control block B. User stack
C. Kernel stack D. None

481. Number of PCB's for a process in multithreaded environment
A. One B. One per process
C. One per thread D. None
E. A & B

482. As threads belongs to same process they can communicate without invoking the kernel
A. Yes B. No
C. None

483. Threads are
A. Synchronous B. Asynchronous
C. Fast in getting started and exiting compared to processes
D. B & C
E. A & C

484. Find odd one out with respect to thread.
A. Ready B. Running
C. Blocked D. Suspended
E. None

485. If process is swapped out
A. All of its threads gets swapped out
B. Currently running thread continue to run
C. Files are closed D. None

486. User level threads can not
A. Take advantage of multiprocessing
B. Run without changing to kernel
C. Executed without going to kernel mode
D. None

487. Jacketing
A. Is used to convert blocking system call to non-blocking
B. To avoid blocking threads
C. Contains jacket routine code which checks I/O device is busy or not
D. All

488. Can a thread migrate from one processor to another in all systems?
A. No B. Yes
C. Valid in Emerald D. None

489. In distributed OS
A. One thread for one process may be effective
B. Many threads for one process
C. One thread many processors
D. None

490. SMP
A. Tightly coupled MIMD machine
B. Kernel can run on any processor
C. Each processor does self scheduling from the poll of processes
D. All

491. Are user level threads and light weight processes same?
A. Yes B. No

492. A paging system is equipped with a TLB and page fault rate is 20% and hit rate of TLB is 80%. TLB access takes 10ns and RAM access takes 100ns. Calculate average time required to access page address. Assume service time required to load page and making a entry in page & TLB when page fault occurs is 10ms.
A. 10.5 ms B. 11.3 ms
C. 12.1 ms D. None

493. The context switching of process in a multi tasking OS is done by
A. Round robin scheduler
B. Time quantum
C. Dispatcher
D. Medium term scheduler

494. Which of the following is not a valid process state transition?
A. Running → ready
B. Ready → running
C. Blocked → running
D. None

495. The main goal of multiprogramming
A. Maximise device utilisation
B. Minimise response time
C. Increase CPU throughput
D. None

496. In Unix, if n fork() system cAlls are made then 2^n processes are created.
A. Yes
B. No

497. background process
A. Supported by unix
B. Can not take interactive input
C. Outputs on to the terminal from which it is invoked
D. All

498. When a process consumed t1 amount of time in its time slice (t) currently when an interrupt arrived for which t2 time units are taken. Then, the amount of time it is going to run for when it is started again is ___
A. t-t1 B. t-t1-t2 C. t D. None

499. Non-Preemptive
A. SRTF B. FCFS
C. Round-robin D. None

500. Smaller time slice and round robin results in the maximastion of
A. Throughput B. Efficiency
C. Fairness D. Context switching

501. Which of the following may not be the criteria for scheduling?
A. Arrival time B. Priority
C. Response time D. Process size

502. Can a thread be created without first creating a process?
A. Yes
B. No

503. More than one word are put in one cache block to
A. Exploit the temporal locality of reference program
B. Exploit the spatial locality of reference in a program
C. Reduce miss penality
D. None

504. Which of the following statements is false?
A. VM implements the translation of program address space to physical memory.
B. VM allows each program to exceed the size of primary memory.
C. VM increases the degree of multiprogramming.
D. VM reduces the context switching.

505. A processor needs SW interrupt to
A. Test the interrupt system of the processor
B. Implement co-routines
C. Obtain system services which need execution of privileged instructions
D. Return to subroutine

506. A CPU has two modes, privileged and non-privileged. In order to change the mode from privileged to non-privileged
A. An HW interrupt is needed
B. An SW interrupt is needed
C. A privileged instruction is needed
D. A non-privileged instruction is needed

507. Consider a set of n tasks with know runtimes r1,r2,… rn to be run on a uniprocessor. Which of the following results in maximum throghput?
A. Round-robin
B. SJF
C. Highest response ratio next
D. FCFS

508. Where does the swap space reside?
A. RAM B. ROM
C. Disk D. On-chip cache

509. Swap space is
A. Disk partition B. Disk file
C. A & B D. None

510. Consider a VM system with FIFO page replacement policy. For an arbitrary page access pattern increasing the number of page frames in main memory will
A. Always decrease the number of page faults
B. Always increase the number of page faults
C. Sometime increase the number of page faults
D. Never affect the number of page faults

511. Which of the following required a device driver?
A. Register B. Cache
C. Main memory D. Disk

512. Size command in Unix
A. Displays text area size of the executable file
B. Displays data area size of the executable file
C. A & B
D. None

513. Job & Process are same
A. No
B. Yes
C. Job may become as a process at a later stage
D. None

514. Suppose the time to serve a page fault is on average 10 ms while a memory access takes 1ms. Then 99.99% hit ratio results in average memory access time of

A. 1.9999ms
B. 1ms
C. 9.999ms
D. 1.9999 microseconds

515. I/O redirection
A. Implies changing the name of a file
B. Can be employed to use an existing file as input file for a program
C. Implies connecting two programs through a pipe
D. None

516. When an interrupt occurs, an OS
A. Ignores the interrupt
B. Always changes state of interrupted process to be blocked and schedules another process
C. Always resumes execution of interrupted process after processing the interrupt
D. May change state of interrupted process to be blocked and schedule another process

517. Dirty bit for a page in a page table
A. Helps avoid unnecessary writes on a paging device
B. Helps maintain LRU information
C. Allows only read on a page
D. None

518. To continue a background process to continue to run
A. Nice B. Nohup
C. Bg D. None

519. In multiple queue scheduling
A. Any scheduling policy can be used in each queue
B. Time slice value may increase as it goes downwords to down queue
C. Is used in Unix
D. All

520. The scheduling policy used in real time systems
A. FCFS B. Deadline
C. SJF D. Round robin

521. If a cache access requires one clock cycle and handling cache misses stalls the processor for an additional five cycles, which of the following cache hit rates comes closest to achieving an average memory access of 2 cycles?
A. 75% B. 80% C. 83% D. 86%
E. 98%

Explanation: 2 = (1 cycle for cache) + (1 - hit rate)(5 cycles stall) ⇒ hit rate = 80%

522. LRU is an effective cache replacement strategy primarily because programs
A. Exhibit spatial locality
B. Exhibit temporal locality
C. Usually have small working sets
D. Read data much more frequently than write data
E. Can generate addresses that collide in the cache

Explanation: Temporal locality implies that the probability of accessing a location decreases as the time since the last access increases. By choosing to replace locations that haven't been used for the longest time, the least-recently-used replacement strategy should, in theory, be replacing locations that have the lowest probability of being accessed in the future.

523. If increasing the associativity of a cache improves performance it is primarily because programs
A. Exhibit spatial locality
B. Exhibit temporal locality
C. Usually have small working sets
D. Read data much more frequently than write data
E. Can generate addresses that collide in the cache

Explanation: Increasing cache associativity means that there are more cache locations in which a given memory word can reside, so replacements due to cache collisions (multiple addresses mapping to the same cache location) should be reduced.

524. If increasing the block size of a cache improves performance it is primarily because programs
A. Exhibit spatial locality
B. Exhibit temporal locality
C. Usually have small working sets
D. Read data much more frequently than write data
E. Can generate addresses that collide in the cache

Explanation : Increased block size means that more words are fetched when filling a cache line after a miss on a particular location. If this leads to increased performance, then the nearby words in the block must have been accessed by the program later on, i.e., the program is exhibiting spatial locality.

525. A fully-associative cache using an LRU replacement policy always has a better hit rate than a direct-mapped cache with the same total data capacity.
A. True B. False

Explanation: False. Suppose both caches contain N words and consider a program that repeatedly accesses locations 0 through N (a total of N+1 words). The direct-mapped cache will map locations 0 and N into the same cache line and words 1 through N-1 into separate cache lines. So in the steady state, the program will miss twice (on locations 0 and N) each time through the loop.

Now the fully-associative case: when the program first accesses word N, the FA cache will replace word 0 (the least-recently-used location). The next access is to location 0 and the FA cache will replace word 1, etc. So the FA cache is always chosing the word to be replaced as the word that the program is about to access, leading to a 0%hit rate.

526. Consider the following program:

```
integer A[1000];
for i = 1 to 1000
for j = 1 to 1000
 A[i] = A[i] + 1
```

When the above program is compiled with all compiler optimisations turned off and run on a processor with a 1K byte direct-mapped, write-back data cache with 4-word cache blocks, what is the approximate data cache miss rate? (Assume integers are one word long and a word is 4 bytes.)

A. 0.0125% B. 0.05%
C. 0.1% D. 5%
E. 12.5%

Explanation: Considering only the data accesses, the program performs 1,000,000 reads and 1,000,000 writes. Since the cache has 4-word blocks, each miss brings 4 words of the array into the cache. So accesses to the next 3 array locations won't cause a miss. Since the cache is write-back, writes happen directly into the cache without causing any memory accesses until the word is replaced. So altogether there are 250 misses (caused by a read of A[0], A[4], A[8], ...), for a miss rate of 250/2,000,000 = 0.0125%.

527. In a non-pipelined single-cycle-per-instruction processor with an instruction cache, the average instruction cache miss rate is 5%. It takes 8 clock cycles to fetch a cache line from the main memory. Disregarding data cache misses, what is the approximate average CPI (cycles per instruction)?

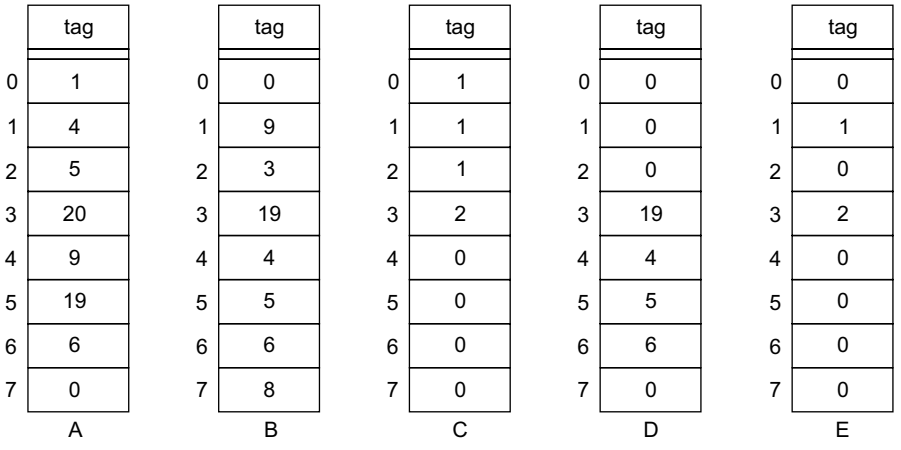
A. 0.45 B. 0.714
C. 1.4 D. 1.8
E. 2.22

Explanation: CPI = (1 inst-per-cycle) + (0.05)(8 cycles/miss) = 1.4

528. Consider an 8-line one-word-block direct-mapped cache initialised to all zeroes where the following sequence of word addresses are accessed:

1, 4, 5, 20, 9, 19, 4, 5, 6, and 9.

Which of the following tables reflect the final tag bits of the cache?



Explanation: First map the addresses to cache line numbers and tags where line number = address mod 8
tag = floor(address / 8)

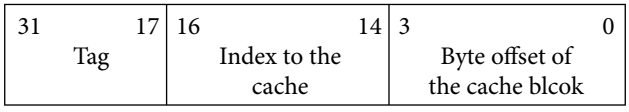
address: 1 4 5 20 9 19 4 5 6 9

line #: 1 4 5 4 1 3 4 5 6 1

tag: 0 0 0 2 1 2 0 0 0 1

So, figure (E) represents the final tag bits of the cache.

529. Consider the following partitioning of a CPU's 32-bit address output which is fed to a direct-mapped write-back cache:



What is the memory requirement for data, tag and status bits of the cache?

A. 8 K bits B. 42 K bits
C. 392 K bits
D. 1,160 K bits
E. 3,200 K bits

Explanation: The tag is 15 bits, cache index is 13 bits, and byte offset 4 bits (i.e, 16 bytes/block). So there are 2^{13} = 8192 cache lines. Each cache line requires

15 tag bits

1 valid bit

1 dirty bit (since this is a write-back cache)

128 data bits (16 bytes/cache line)

===

145 bits per cache line

Total storage required = 8192*145 bits = 1,160K bits.

530. Find odd one out of the following:

A. Process B. Thread

C. Job D. Program

531. ___ is the one which becomes as a process at a later stage of its life.

A. Process B. Thread

C. Job D. Program

532. In Java, for every thread there exists a _____.

A. Class file B. Java file

C. Thread function with the name run()

D. File

533. Every thread will not have its own _____.

A. Thread function B. Page table

C. Stack pointer D. TCB

534. Find the correct one

A. Address space is created only when a process is created

B. Even without address space also, a thread can run

C. In Java where function name overloading is supported, we can have more than one thread function by defining more than one run() methods.

D. A thread can be made to create a new page table.

ANSWER KEY

1. A	**2.** D	**3.** B	**4.** B, C
5. A	**6.** A	**7.** A	**8.** A
9. A	**10.** A	**11.** A	**12.** B
13. B	**14.** B	**15.** A	**16.** A
17. A	**18.** D	**19.** B	**20.** C
21. B	**22.** C	**23.** B	**24.** C
25. D	**26.** C	**27.** B	**28.** B
29. A	**30.** C	**31.** C	**32.** A
33. B	**34.** C	**35.** H	**36.** B
37. C	**38.** B	**39.** C	**40.** B
41. C	**42.** D	**43.** B	**44.** C
45. A	**46.** A,B	**47.** B,C	**48.** C
49. D	**50.** D	**51.** B	**52.** C
53. E	**54.** D	**55.** B	**56.** C
57. D	**58.** E	**59.** B	**60.** C
61. C	**62.** D	**63.** C	**64.** E
65. E	**66.** E	**67.** D	**68.** C
69. E	**70.** D	**71.** C	**72.** A
73. D	**74.** A	**75.** D	**76.** C
77. A,C	**78.** C	**79.** D	**80.** B
81. A	**82.** D	**83.** C	**84.** D
85. A	**86.** B	**87.** B	**88.** A,D
89. C	**90.** E	**91.** B	**92.** B
93. B	**94.** B	**95.** A	**96.** D
97. D	**98.** A	**99.** C	**100.** C
101. B	**102.** D	**103.** A	**104.** B
105. C	**106.** B	**107.** D	**108.** A
109. C	**110.** D	**111.** A	**112.** C
113. B	**114.** C	**115.** C	**116.** A
117. C	**118.** B	**119.** B	**120.** A
121. B	**122.** D	**123.** C	**124.** C
125. D	**126.** C	**127.** C	**128.** D
129. C	**130.** B	**131.** C	**132.** B
133. B	**134.** C	**135.** D	**136.** B
137. C	**138.** C	**139.** B	**140.** C
141. C	**142.** C	**143.** B	**144.** C
145. B	**146.** C	**147.** C	**148.** A
149. A	**150.** C	**151.** B	**152.** A
153. B	**154.** C	**155.** A	**156.** B
157. A	**158.** C	**159.** C	**160.** C
161. D	**162.** A	**163.** C	**164.** C
165. C	**166.** C	**167.** C	**168.** C
169. C	**170.** C	**171.** C	**172.** C
173. D	**174.** C	**175.** C	**176.** D
177. D	**178.** C	**179.** D	**180.** D
181. C	**182.** A	**183.** A	**184.** B
185. C	**186.** D	**187.** B	**188.** D
189. C	**190.** D	**191.** B	**192.** D
193. C	**194.** D	**195.** C	**196.** A
197. C	**198.** C	**199.** C	**200.** D
201. C	**202.** C	**203.** C	**204.** B
205. C	**206.** B	**207.** B	**208.** D
209. D	**210.** D	**211.** D	**212.** D
213. D	**214.** D	**215.** D	**216.** D
217. C	**218.** C	**219.** D	**220.** D
221. D	**222.** C	**223.** D	**224.** D

225. D **226.** C **227.** C **228.** D
229. B **230.** D **231.** D **232.** D
233. A **234.** A **235.** C **236.** D
237. C **238.** D **239.** A **240.** C
241. C **242.** C **243.** A **244.** D
245. D **246.** D **247.** D **248.** C
249. D **250.** B **251.** C **252.** C
253. A **254.** A **255.** C **256.** D
257. C **258.** A **259.** C **260.** B
261. A **262.** C **263.** C **264.** C
265. C **266.** C **267.** A **268.** C
269. B **270.** C **271.** B **272.** C
273. C **274.** C **275.** E **276.** C
277. D **278.** D **279.** B **280.** C
281. B **282.** B **283.** C **284.** C
285. A **286.** A **287.** B **288.** C
289. E **290.** A **291.** C **292.** D
293. C **294.** C **295.** D **296.** D
297. C **298.** B **299.** B **300.** C
301. D **302.** D **303.** A **304.** C
305. B **306.** C **307.** A **308.** D
309. D **310.** D **311.** A **312.** D
313. A **314.** B **315.** C **316.** D
317. C **318.** C **319.** C **320.** C
321. E **322.** D **323.** D **324.** B
325. C **326.** D **327.** D **328.** C
329. E **330.** E **331.** C **332.** E
333. D **334.** E **335.** D **336.** A
337. E **338.** C **339.** D **340.** D
341. E **342.** D **343.** D **344.** C
345. E **346.** B **347.** B **348.** E
349. C **350.** B **351.** E **352.** E
353. C **354.** C **355.** C **356.** B
357. D **358.** C **359.** D **360.** B
361. B **362.** B **363.** D **364.** C
365. B **366.** A **367.** C **368.** A
369. A **370.** A **371.** D **372.** A
373. D **374.** C **375.** B **376.** D
377. D **378.** A **379.** B **380.** B
381. D **382.** A **383.** D **384.** D
385. ? **386.** A **387.** D **388.** D
389. D **390.** A **391.** A **392.** C
393. B **394.** A **395.** B **396.** C
397. B **398.** A **399.** B **400.** B
401. E **402.** A **403.** D **404.** A
405. B **406.** A **407.** B **408.** A
409. C **410.** D **411.** A **412.** A
413. D **414.** C **415.** B **416.** A
417. A **418.** B **419.** D **420.** C
421. C **422.** A **423.** D **424.** D
425. B **426.** D **427.** C **428.** D
429. E **430.** D **431.** B **432.** D
433. D **434.** C **435.** D **436.** C
437. B **438.** B **439.** C **440.** A
441. B **442.** D **443.** C **444.** C
445. D **446.** B **447.** C **448.** D
449. D **450.** D **451.** C **452.** D
453. C **454.** C **455.** A **456.** C
457. B **458.** D **459.** C **460.** B
461. D **462.** D **463.** B **464.** C
465. E **466.** B **467.** B **468.** B
469. D **470.** A **471.** D **472.** A
473. C **474.** D **475.** C **476.** C
477. C **478.** E **479.** B **480.** D
481. A **482.** A **483.** D **484.** D
485. A **486.** A **487.** D **488.** C
489. C **490.** D **491.** B **492.** D
493. C **494.** C **495.** C **496.** A
497. D **498.** B **499.** B **500.** C
501. D **502.** B **503.** B **504.** D
505. C **506.** C **507.** B **508.** C
509. C **510.** C **511.** D **512.** C
513. C **514.** B **515.** B **516.** C
517. A **518.** B **519.** D **520.** B
521. B **522.** B **523.** E **524.** A
525. B **526.** A **527.** C **528.** E
529. D **530.** D **531.** C **532.** C
533. B **534.** A

Previous Years' GATE Questions

1. A scheduling algorithm assigns priority proportional to the waiting time of a process. Every process starts with priority zero (the lowest priority). The scheduler re-evaluates the process priorities every T time units and decides the next process to schedule. Which one of the following is true if the processes have no I/O operations and all arrive at time zero? **(GATE 2013)**

A. The algorithm is equivalent to the first cum first serve algorithm

B. The algorithm is equivalent to the round robin algorithm
C. The algorithm is equivalent to the shortest job first algorithm
D. The algorithm is equivalent to the shortest remaining job first algorithm

2. Three concurrent processes X,Y,Z executes three different code segments that access and update certain shared variables. Process X executes the P operation (i.e wait) on semaphores a, b, and c; process Y executes the P operation on semaphores b, c, and d; process z executes P operation on c, d and a before entering the respective code segments. After completing the execution of its code segment, each process invokes the V operation (i.e., signal) on its three semaphores. All semaphores are binary semaphores initialised to one. Which one of the following represents dead-lock free order of invoking the P operations by the processes?

(GATE 2013)

A. X:P(a),P(b),P(c), Y:P(b), P(c), P(d), Z:P(c),P(d), P(a)
B. X:P(b),P(a),P(c), Y:P(b), P(c), P(d), Z:P(c),P(d), P(a)
C. X:P(b),P(a),P(c), Y:P(c), P(b), P(d), Z:P(a),P(c), P(d)
D. X:P(a),P(b),P(c), Y:P(c), P(b), P(d), Z:P(c),P(d), P(a)

3. In a k-way set associative cache, the cache is divided into v sets, each of which consists of k lines. The lines of a set are placed in sequence one after another. The lines in a sets are sequenced before the lines of s+1. The main memory blocks are numbered from 0 onwards. The main memory block numbered j must be mapped to any one of the cache lines from

(GATE 2013)

A. (j mod v)*k to (j mod v)*k+k-1
B. (j mod v) to (j mod v)+k-1
C. (j mod k) to (j mod k)+v-1
D. (j mod k)*v to (j mod k)*v+v-1

4. Consider a hard disk with 16 recording surfaces (0-15) having 16384 cylinders (0-16383) and each cylinder contains 64 sectors (0-63). Data storage capacity in each sector is 512 bytes. Data are organised cylinder-wise and the addressing format <cylinder no, surface no., sector no>. A file of capacity 42797KB is stored in the disk and the starting disk location of the file <1200,9,40>. What is the cylinder number of the last sector of the file, if it is stored in a contiguous manner?. **(GATE 2013)**

A. 1281 B. 1282 C. 1283 D. 1284

Explanation: Capacity of cylinder= 16×64×512 bytes = 512KB

Number of cylinders needed to store the file= 42797/512 = 83.58

As starting address of the file=<1200,9,40>, last cylinder becomes 1283.

5. A shared variable x, initialised to zero, is operated on by four concurrent processes W, X, Y, Z as follows. Each of the processes W and X reads x from memory, increments by one stores it to memory and terminates. Each of the processes Y and Z reads x from memory, decrements by two and stores to memory and terminates. Each process before reading x invokes P operation (i.e wait) on a counting semaphore S and invokes V operation (i.e signal) on the semaphore S after storing x in memory. Semaphore S is initialised to two. What is the maximum possible value of x after all processes complete execution. **(GATE 2013)**

A. –2 B. –1 C. 1 D. 2

Explanation: We know least possible value of a counting semaphore is 0. Thus, if we assume Y and Z are executed first then value of x becomes 0. After that when W and X are executed then value of x will become 2. This is the largest possible value for x.

6. A certain computation generates two arrays a and b such that a[i]=f(i) for 0<i<n and b[i]=g(i) for 0<i<n. Suppose this computation is decomposed into two concurrent processes X and Y such that X computes the array a and Y computes array b. The processes employ two binary semaphores R and S, both initialised to 0. The array a is shared by the two processes. The structure of the processes are shown below.

(GATE 2013)

```
Process X:                  Process Y:
private i;                  private i;
for (i=0; i<n; i++) {       for (i=0; i<n; i++) {
   a[i] = f(i);                EntryY(R, S);
   ExitX(R, S);                b[i] = g(a[i]);
}                           }
```

Which one of the following represents the correct implementations of ExitX and EntryY?

A.
```
Exit X (R,  S) {
    P (R) ;
    V (S);
}
EntryY (R,  S) {
    P(S) ;
    V(R);
}
```

B.
```
Exit X (R,  S) {
    V(R) ;
    V(S);
}
EntryY (R,  S) {
    P(R) ;
    P(S);
}
```

C.
```
Exit X (R, S) {
    P(S) ;
    V(R);
}
EntryY (R, S) {
    V(S) ;
    P(R);
}
```

D.
```
Exit X (R, S) {
    V(R) ;
    P(S);
}
EntryY (R, S) {
    V(S) ;
    P(R);
}
```

7. A process executes the code **(GATE 2012)**

```
fork();
fork();
fork();
```

The total number of child processes created is

A. 3 B. 4 C. 7 D. 8

8. Consider the 3 processes, P1, P2 and P3 shown in the table. **(GATE 2012)**

Process	Arrival Time	Time Units Required
P1	0	5
P2	1	7
P3	3	4

The completion order of the 3 processes under the policies FCFS and RR2 (round robin scheduling with CPU quantum of 2 time units) are

A. FCFS: P1, P2, P3 RR2: P1, P2, P3

B. FCFS: P1, P3, P2 RR2: P1, P3, P2

C. FCFS: P1, P2, P3 RR2: P1, P3, P2

D. FCFS: P1, P3, P2 RR2: P1, P2, P3

Explanation: Options B and C can be eliminated based on the fact that in FCFS, processes are executed in first come first serve basis. Now, we decide out of A and C. See the following chart which indicates how processes are executed under RR2.We find P1, P3 and P2 are completed in the order. Thus, option C is valid.

	Time
P1	0
P1	1
P2	2
P2	3
P1	4
P1	5
P3	6
P3	7
P1	8
P2	9
P2	10
P3	11
P3	12
P2	13
P2	14
P2	15

9. Fetch_And_Add(X,i) is an atomic Read-Modify-Write instruction that reads the value of memory location X, increments it by the value i, and returns the old value of X. It is used in the pseudocode shown below to implement a busy-wait lock. L is an unsigned integer shared variable initialised to 0. The value of 0 corresponds to lock being available, while any non-zero value corresponds to the lock being not available. **(GATE 2012)**

```
AcquireLock(L){
while (Fetch_And_Add(L,1))
        L = 1;
}
ReleaseLock(L){
L = 0;
}
```

This implementation

A. Fails as L can overflow

B. Fails as L can take on a non-zero value when the lock is actually available

C. Works correctly but may starve some processes

D. Works correctly without starvation

10. A file system with 300 GByte disk uses a file descriptor with 8 direct block addresses, one indirect block address and one doubly indirect block address. The size of each disk block is 128 Bytes and the size of each disk block address is 8 Bytes. The maximum possible file size in this file system is **(GATE 2012)**

A 3 KBytes

B. 35 KBytes

C. 280 KBytes

D. Dependent on the size of the disk

Explanation: In one block we can store 128/8=16 addresses. Thus, maximum file size=(8+16+16*16) blocks=280x128bytes=35KB

11. Consider the virtual page reference string **(GATE 2012)**

1, 2, 3, 2, 4, 1, 3, 2, 4, 1

on a demand paged virtual memory system running on a computer system that has main memory size of 3 page frames which are initially empty. Let LRU, FIFO

and OPTIMAL denote the number of page faults under the corresponding page replacement policy. Then

A. OPTIMAL < LRU < FIFO

B. OPTIMAL < FIFO < LRU

C. OPTIMAL = LRU

D. OPTIMAL = FIFO

12. A computer has a 256 KByte, 4-way set associative, write back data cache with block size of 32 Bytes. The processor sends 32 bit addresses to the cache controller. Each cache tag directory entry contains, in addition to address tag, two valid bits, one modified bit and one replacement bit. **(GATE 2012)**

The number of bits in the tag field of an address is

A. 11 B. 14 C. 16 D. 27

13. The size of the cache tag directory is **(GATE 2012)**

A. 160 Kbits B. 136 Kbits

C. 40 Kbits D. 32 Kbits

14. Let the page fault service time be 10ms in a computer with average memory access time being 20ns. If one page fault is generated for every 10^6 memory accesses, what is the effective access time for the memory? **(GATE 2011)**

A. 21ns B. 30ns C. 23ns D. 35ns

Explanation: If p is page fault rate, effective memory access time=p*page fault service time + (1-p)*memory access time= $=1/10^6*10*10^6+(1-1/10^6)*20 = 29.9ns = 30ns$

15. Let the time taken to switch between user and kernel modes of execution be t1 while the time taken to switch between two processes be t2. Which of the following is true? **(GATE 2011)**

A. t1 > t2

B. t1 = t2

C. t1 < t2

D. Nothing can be said about the relation between t1 and t2

16. An 8KB direct mapped write-back cache is organised as multiple blocks, each of size 32-bytes. The processor generates 32-bit addresses. The cache controller maintains the tag information for each cache block comprising of the following. **(GATE 2011)**

1 Valid bit

1 Modified bit

As many bits as the minimum needed to identify the memory block mapped in the cache.

What is the total size of memory needed at the cache controller to store meta data (tags) for the cache?

A. 4864 bits B. 6144bits C. 6656bits D. 5376bits

Explanation: Answer is (D)

Cache size = 8KB = 2^{13}Bytes

Block size = 32B = 2^5B

Number of cache blocks = $2^{13}/2^5 = 2^8$

RAM uses 32 bit addresses.

Therefore, tag bits = (32–13) + 1 + 1 = 21

Total size of memory needed for storing tags

=Number blocks*number of bits used for tag

$=256 \times 21 = 5376$bits

17. An application loads 100 libraries at startup. Loading each library requires exactly one disk access. The seek time of the disk to a random location is given as 10ms. Rotational speed of disk is 6000rpm. If all 100 libraries are loaded from random locations on the disk, how long does it take to load all libraries? (The time to transfer data from the disk block once the head has been positioned at the start of the block may be neglected) (GATE 2011)

A. 0.50s B. 1.50s C. 1.25s D 1.00s

Explanation:

Rotational latency =half of time for one rotation = 1/2*60/6000*1000= 5ms

Therefore, time needed for one disk access(time needed to load one library) = 15 ms

Time to load all libraries = 15 ×100 = 1500ms = 1.5sec

18. On a non-pipelined sequential processor, a program segment, which is a part of the interrupt service routine, is given to transfer 500 bytes from an I/O device to memory.

Initialise the address register

Initialise the count to 500

LOOP: Load a byte from device

Store in memory at address given by address register

Increment the address register

Decrement the count

If count != 0 go to LOOP

Assume that each statement in this program is equivalent to a machine instruction which takes one clock cycle to execute if it is a non-load/store instruction. The load-store instructions take two clock cycles to execute.

The designer of the system also has an alternate approach of using the DMA controller to implement the same transfer. The DMA controller requires 20 clock cycles for initialisation and other overheads. Each DMA transfer cycle takes two clock cycles to transfer one byte of data from the device to the memory.

What is the approximate speedup when the DMA

controller based design is used in place of the interrupt driven program based input-output?

(GATE 2011)

A 3.4 B 4.4 C 5.1 D 6.7

Explanation: Number of clock cycles required by using load-store approach = 2 + 500 × 7 = 3502 and that of by using DMA = 20 + 500 × 2 = 1020

Required speed up=3502/1020=3.4

19. Consider the following table of arrival time and burst time for three processes P0, P1 and P2. **(GATE 2010)**

Process	Arrival time	Burst Time
P0	0 ms	9 ms
P1	1 ms	4ms
P2	2 ms	9ms

The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the three processes?

A. 5.0 ms B. 4.33 ms C. 6.33 ms D. 7.33 ms

Explanation:

At 0 P0 will be started.

At 1 P0 is preempted and P1 is initiated. It will be completed at time 5. In the mean time P2 also arrives. However, it will not be preempting P1. After P1, P0 starts and completes by 13. Then P2 will be loaded and completed. Thus, waiting times for P0, P1 and P2 are 0,4,(13–2=11). Therefore average waiting= (0+4+11)/3=5ms

20. Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared Boolean variables S1 and S2 are randomly assigned.

Method used by P1	Method used by P2
while (S1 = = S2) ;	while (S1 ! = S2) ;
Critica1 Section	Critica1 Section
S1 = S2;	S2 = not (S1);

Which one of the following statements describes the properties achieved? **(GATE 2010)**

A. Mutual exclusion but not progress

B. Progress but not mutual exclusion

C. Neither mutual exclusion nor progress

D. Both mutual exclusion and progress

21. A system uses FIFO policy for page replacement. It has 4 page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages but now in the reverse order. How many page faults will occur? **(GATE 2010)**

A 196 B. 192 C. 197 D 195

Explanation: Answer is A. First time, every page reference makes a page fault. At the end of first access last four pages will be in RAM and with them no page fault occurs during the reverse pass. Thus, 196 page faults will be seen in total.

22. Which of the following statements are true? **(GATE 2010)**

I. Shortest remaining time first scheduling may cause starvation

II. Preemptive scheduling may cause starvation

III. Round robin is better than FCFS in terms of response time

A. I only B. I and III only

C. II and III only D. I, II and III

23. The following program consists of 3 concurrent processes and 3 binary semaphores. The semaphores are initialised as S0=1, S1=0, S2=0.

Process P0	Process P1	Process P2

while (true) {
wait (S0);
print '0'
release (S1);
release (S2);
}
	wait (S1);
	Release (S0);
		wait (S2);
		release (S0);

How many times will process P0 print '0'? **(GATE 2010)**

A. At least twice B. Exactly twice

C. Exactly thrice D. Exactly once

24. A system has n resources R0,…,Rn-1, and k processes P0,…..Pk-1. The implementation of the resource request logic of each process Pi. is as follows:

```
if (i% 2==0) {
if (i<n) request Ri ;
if (i+2<n)request Ri+2 ;
}
else {
if (i<n) request Rn-i ;
if (i+2<n)request Rn-i-2 ;
}
```

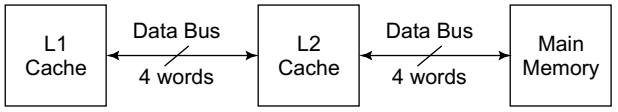
In which one of the following situations is a deadlock possible? **(GATE 2010)**

A n = 40,k = 26 B. n = 21,k = 12

C n = 20,k = 10 D. n = 41,k = 19

25. A computer system has an L1 cache, an L2 cache, and a main memory unit connected as shown below. The block size in L1 cache is 4 words. The block size in L2 cache is 16 words. The memory access times are 2

nanoseconds. 20 nanoseconds and 200 nanoseconds for L1 cache, L2 cache and main memory unit respectively.



When there is a miss in L1 cache and a hit in L2 cache, a block is transferred from L2 cache to L1 cache. What is the time taken for this transfer? **(GATE 2010)**

A. 2 nanoseconds B. 20 nanoseconds
C. 22 nanoseconds D. 88 nanoseconds

26. When there is a miss in both L1 cache and L2 cache, first a block is transferred from main memory to L2 cache, and then a block is transferred from L2 cache to L1 cache. What is the total time taken for these transfers? **(GATE 2009)**

A 222 nanoseconds B. 888 nanoseconds
C 902 nanoseconds D. 968 nanoseconds

27. A CPU generally handles an interrupt by executing an interrupt service routine **(GATE 2009)**

A. As soon as an interrupt is raised
B. By checking the interrupt register at the end of fetch cycle.
C. By checking the interrupt register after finishing the execution of the current instruction.
D. By checking the interrupt register at fixed time intervals.

28. In which one of the following page replacement policies, Belady's anomaly may occur? **(GATE 2009)**

A. FIFO B. Optimal C. LRU D. MRU

29. The essential content(s) in each entry of a page table is / are **(GATE 2009)**

A Virtual page number
B Page frame number
C Both virtual page number and page frame number
D Access right information

30. Consider a 4-way set associative cache (initially empty) with total 16 cache blocks. The main memory consists of 256 blocks and the request for memory blocks is in the following order:

0, 255, 1, 4, 3, 8, 133, 159, 216, 129, 63, 8, 48, 32, 73, 92, 155.

Which one of the following memory block will not be in cache if LRU replacement policy is used? **(GATE 2009)**

A. 3 B 8 C 129 D 216

Explanation: As cache is 4 way set associative cache with 16 blocks, each set has 4 blocks numbered 0,1,2,3. We take each memory block and use the last 2 bits to place it into proper cache following LRU replacement scheme. 216 would not be in cache as it is followed by 8, 48, 32 and 92.

31. Consider a system with 4 types of resources R1 (3 units), R2 (2 units), R3 (3 units), R4 (2 units). A non-preemptive resource allocation policy is used. At any given instance, a request is not entertained if it cannot be completely satisfied. Three processes P1, P2, P3 request the sources as follows if executed independently.

Process P1:	Process P2:	Process P3:
t=0: requests 2 units of R2	t=0: requests 2 units of R3	t=0: requests 1 unit of R4
t=1: requests 1 unit of R3	t=2: requests 1 unit of R4	t=2: requests 2 units of R1
t=3: requests 2 units of R1	t=4: requests 1 unit of R1	t=5: releases 2 units of R1
t=5: releases 1 unit of R2 and 1 unit of R1.	t=6: releases 1 unit of R3	t=7: requests 1 unit of R2
t=7: releases 1 unit of R3	t=8: Finishes	t=8: requests 1 unit of R3
t=8: requests 2 units of R4		t=9: Finishes
t=10: Finishes		

Which one of the following statements is true if all three processes run concurrently starting at time t=0? **(GATE 2009)**

A All processes will finish without any deadlock
B Only P1 and P2 will be in deadlock.
C Only P1 and P3 will be in a deadlock.
D All three processes will be in deadlock.

32. Consider a disk system with 100 cylinders. The requests to access the cylinders occur in following sequence:

4, 34, 10, 7, 19, 73, 2, 15, 6, 20

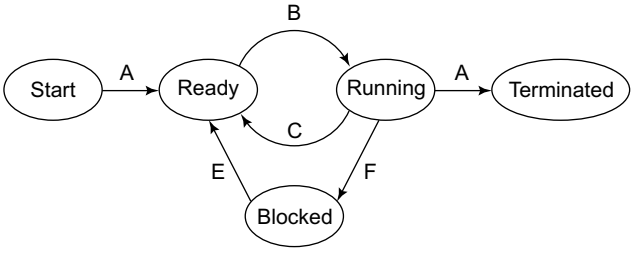
Assuming that the head is currently at cylinder 50, what is the time taken to satisfy all requests if it takes 1ms to move from one cylinder to adjacent one and shortest seek time first policy is used? **(GATE 2009)**

A 95ms B 119ms C 233ms D 276ms

Explanation: As head is at 50, it is going to handle the requests as follows according to SSTF policy: 50,34,20,19,15,10,7,6,4,2,73

Therefore, time taken=[(50–34)+(34–20)+(20–19)+(19–15)+(15–10)+(10–7)+(7–6)+(6–4)+(4–2)+(73–2)]*1ms = 119ms

33. In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state:



Now consider the following statements:

I. If a process makes a transition D, it would result in another process making transition A immediately.

II. A process P2 in blocked state can make transition E while another process P1 is in running state.

III. The OS uses preemptive scheduling.

IV. The OS uses non-preemptive scheduling.

Which of the above statements are true?

(GATE 2009)

A I and II B I and III C II and III D II and IV

34. The enter_CS() and leave_CS() functions to implement critical section of a process are realised using test-and-set instruction as follows:

```
void enter_CS(X)
{
while (test-and-set(X) );
}
void leave_CS(X)
{
X=0;
}
```

In the above solution, X is a memory location associated with the CS and is initialised to 0. Now consider the following statements:

I. The above solution to CS problem is deadlock-free

II. The solution is starvation free.

III. The processes enter CS in FIFO order.

IV More than one process can enter CS at the same time.

Which of the above statements is true?

(GATE 2009)

A. I only B. I and II C. II and III D. IV only

35. A multilevel page table is preferred in comparison to a single level page table for translating virtual address to physical address because **(GATE 2009)**

A It reduces the memory access time to read or write a memory location.

B It helps to reduce the size of page table needed to implement the virtual address space of a process.

C It is required by the translation lookaside buffer.

D It helps to reduce the number of page faults in page replacement algorithms.

36. A hard disk has 63 sectors per track, 10 platters each with 2 recording surfaces and 1000 cylinders. The address of a sector is given as a triple <c,h, s> , where c is the cylinder number, h is the surface number and s is the sector number. Thus, the 0th sector is addressed as <0,0,0> , the 1st sector as <0,0,1> , and so on

The address <400, 16, 29> corresponds to sector number: **(GATE 2009)**

A. 505035 B. 505036 C. 505037 D. 505038

Explanation: 400×2×10×63 + 16×63 + 29 = 505037

37. The address of the 1039th sector is **(GATE 2009)**

A < 0,15,31> B. <0,16,30>

C. < 0,16,31> D. <0,17,31>

Explanation: 16×63 + 31 = 1039

38. Consider a 4-way set associative cache consisting of 128 lines with a line size of 64 words. The CPU generates a 20-bit address of a word in main memory. The number of bits in the TAG, LINE and WORD fields are respectively: **(GATE 2007)**

A. 9, 6, 5 B. 7, 7, 6 C. 7, 5, 8 D. 9, 5, 6

39. Consider a disk pack with 16 surfaces, 128 tracks per surface and 256 sectors per track. 512 bytes of data are stored in a bit serial manner in a sector. The capacity of the disk pack and the number of bits required to specify a particular sector in the disk are respectively:

(GATE 2007)

A 256 Mbyte, 19 bits B. 256 Mbyte, 28 bits

C. 512 Mbyte, 20 bits D. 64 Gbyte, 28 bits

Explanation: Disk capacity = $16 \times 128 \times 256 \times 512 = 2^4 \times 2^7 \times 2^8 \times 2^9 = 2^{28}$ bytes = 256MB

Number of sectors = $16 \times 128 \times 256 = 2^4 \times 2^7 \times 2^8 = 2^{19}$

Therefore, 19 bits are needed to refer to any sector. Thus, correct option is A.

40. Consider the following statements about user level threads and kernel level threads. Which one of the following statements is false? **(GATE 2007)**

A Context switch time is longer for kernel level threads than for user level threads.

B User level threads do not need any hardware support.

C Related kernel level threads can be scheduled on different processors in a multi-processor system.

D Blocking one kernel level thread blocks all related threads.

41. An operating system uses Shortest Remaining Time first (SRT) process scheduling algorithm. Consider the arrival times and execution times for the following processes:

Process	Execution time	Arrival time
P1	20	0
P2	25	15
P3	10	30
P4	15	45

What is the total waiting time for process P2?

(GATE 2007)

A. 5 B. 15 C. 40 D. 55

Explanation: At 0 P1 will be started and completed at 20.

At 20 P2(which is arrived at 15) will be started as it is the only process in the ready queue.

At 30, P3 arrives which needs 10 units to complete. While currently running P2 needs 15 more time units to complete. Thus, P2 is preempted and P3 is started.

P3 completes at 40 and thus P2 is again initiated.

At 45 P4 arrives which needs 15 time units to complete. As this is more than currently running P2, P2 continues to run.

Thus, total waiting time of P2=5+10=15 time units

42. A virtual memory system uses First In First Out (FIFO) page replacement policy and allocates a fixed number of frames to a process. Consider the following statements:

P: Increasing the number of page frames allocated to a process sometimes increases the page fault rate.

Q: Some programs do not exhibit locality of reference.

Which one of the following is true? **(GATE 2007)**

A Both P and Q are true, and Q is the reason for P

B Both P and Q are true, but Q is not the reason for P.

C P is false, but Q is true

D Both P and Q are false.

43. A single processor system has three resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column alloc denotes the number of units of each resource type allocated to each process, and the column request denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish last? **(GATE 2007)**

	alloc			request		
	X	Y	Z	X	Y	Z
P0	1	2	1	1	0	3
P1	2	0	1	0	1	2
P2	2	2	1	1	2	0

A. P0 B. P1

C. P2

D. None of the above, since the system is in a deadlock.

44. Two processes, P1 and P2, need to access a critical section of code. Consider the following synchronisation construct used by the processes:

/* P1 */	/* P2 */
while (true) {	while (true) {
wants1 = true;	wants2 = true;
while (wants2==true);	while (wants1==true);
/* Critical	/* Critical
Section */	Section */
wants1=false;	Wants2=false;
}	}
/* Remainder section */	/* Remainder section */

Here, wants1 and wants2 are shared variables, which are initialised to false.

Which one of the following statements is true about the above construct? **(GATE 2007)**

A. It does not ensure mutual exclusion.

B. It does not ensure bounded waiting.

C. It requires that processes enter the critical section in strict alternation.

D. It does not prevent deadlocks, but ensures mutual exclusion.

45. Consider a machine with a byte addressable main memory of 2^{16} bytes. Assume that a direct mapped data cache consisting of 32 lines of 64 bytes each is used in the system. A 50 × 50 two-dimensional array of bytes is stored in the main memory starting from memory location 1100H. Assume that the data cache is initially empty. The complete array is accessed twice. Assume that the contents of the data cache do not change in between the two accesses.

How many data cache misses will occur in total?

(GATE 2007)

A. 48 B. 50 C. 56 D. 59

Explanation: In direct mapping each physical memory block is mapped to block address % 32(no. of cache lines). Now 1100H = 0001000100 000000 belongs to block 0001000100 which is mapped to cache block

0001000100%32 = cache block 4. So first element is mapped to cache block 4. Since there are 64 elements in each block, 1st 64 elements will be mapped to cache block 4.Access of 1st 64 elements will cause only one miss (1st miss causes entire block to be transferred to cache). 2nd 64 elements will cause one more miss and so on. Since there are 2500 elements access of all elements will cause 40 misses (2500/64).

During second iteration there will be some elements of the array in cache which are already accessed first time. We know total array size is 2500bytes and cache size is 32x64=2048 bytes. Thus, while accessing the array in the first time itself, first 2500-2048=452bytes of the array that are in the cache will be overwritten with last 452 bytes. Thus, while accessing the array second time with these first 452 bytes cache misses occurs as they are not in the cache. That is, we will be getting 452/64=8 (approximately) misses. Also, during second access these 452 bytes will be overwritten with last 452 bytes of the array. Thus, 8 more cache misses are spent. Thus, in total 40+8+8=56 cache misses are needed.

46. Which of the following lines of the data cache will be replaced by new blocks in accessing the array for the second time? **(GATE 2007)**

A. line 4 to line 11 B. line 4 to line 12

C. line 0 to line 7 D. line 0 to line 8

47. A process has been allocated 3 page frames. Assume that none of the pages of the process are available in the memory initially. The process makes the following sequence of page references (reference string): 1, 2, 1, 3, 7, 4, 5, 6, 3, 1

If optimal page replacement policy is used, how many page faults occur for the above reference string?

(GATE 2007)

A. 7 B. 8 C. 9 D. 10

Explanation: Optimal page replacement policy: when a page needs to be swapped in, the OS swaps out the page whose next use will occur farthest in the near future.

page number	1	2	1	3	7	4	5	6	3	1
page fault no.	1	2		3	4	5	6	7		

The pages in the frames are replaced as shown below:

1									
2	7	4	5	6					
3									

The total number of page faults is 7.

48. Least Recently Used (LRU) page replacement policy is a practical approximation to optimal page replacement. For the above reference string, how many more page faults occur with LRU than with the optimal page replacement policy? **(GATE 2007)**

A. 0 B. 1

C. 2 D. 3

Explanation:

page frame	1	2	1	3	7	4	5	6	3	1
page fault no.	1	2		3	4	5	6	7	8	9

The pages in the frames are replaced as shown below:

1	4	3
2	7	6
3	5	1

Total number of page faults is 9.

49. Consider three CPU-intensive processes, which require 10, 20 and 30 time units and arrive at times 0, 2 and 6, respectively. How many context switches are needed if the operating system implements a shortest remaining time first scheduling algorithm? Do not count the context switches at time zero and at the end.

A. 1 B. 2

C. 3 D. 4

Explanation: At time 0, only p1 is available and it is executed. Thus, it cannot be taken as context switch. Processesp2 and p3 arrive at times 2 and 6 respectively. Process p1 executes to completion according to shortest remaining first algorithm, then p2 and then p3. While P2 and P3 are loaded, context switching takes place. Therefore the number of context switches are 2.

50. A CPU has a cache with block size 64 bytes. The main memory has k banks, each bank being c bytes wide. Consecutive c – byte chunks are mapped on consecutive banks with wrap-around. All the k banks can be accessed in parallel, but two accesses to the same bank must be serialized. A cache block access may involve multiple iterations of parallel bank accesses depending on the amount of data obtained by accessing all the k banks in parallel. Each iteration requires decoding the bank numbers to be accessed in parallel and this takes k/2 ns. Latency of one bank access is 80 ns. If c=2 and k = 24, the latency of retrieving a cache block starting at address zero from main memory is:

A. 92 ns B. 104 ns

C. 172 ns D. 184 ns

Explanation: Cache block size=64. Given c=2 and k=24 indicates that we can take 48 bytes in parallel. However, our cache block size 64. Thus, we need two memory accesses as 64 is more than 48 bytes. Thus, latency of reading one cache block=2*(80+24/2)=184ns

51. A computer system supports 32-bit virtual addresses as well as 32-bit physical addresses. Since the virtual address space is of the same size as the physical address space, the operating system designers decide to get rid of the virtual memory entirely. Which one of the following is true?

A. Efficient implementation of multi-user support is no longer possible

B. The processor cache organisation can be made more efficient now

C. Hardware support for memory management is no longer needed

D. CPU scheduling can be made more efficient now

52. Consider three processes (process id 0, 1, 2 respectively) with compute time bursts 2, 4 and 8 time units. All processes arrive at time zero. Consider the longest remaining time first (LRTF) scheduling algorithm. In LRTF ties are broken by giving priority to the process with the lowest process id. The average turn around time is:

A. 13 units B. 14 units

C. 15 units D. 16 units

Explanation: The Gantt chart for the above problem is:

p2	p1	p2	p1	p2	p0	p1	p2	p0	p1	p2

0 4 5 6 7 8 9 10 11 12 13 14

Turn around time of p0 is 12-0 = 12.

Turn around time of p1 is 13-0 = 13.

Turn around time of p2 is 14-0 = 14.

Average turn around time is (12+13+14)/3 = 13.

53. Consider three processes, all arriving at time zero, with total execution time of 10, 20 and 30 units, respectively. Each process spends the first 20% of execution time doing I/O, the next 70% of time doing computation, and the last 10% of time doing I/O again. The operating system uses a shortest remaining compute time first scheduling algorithm and schedules a new process either when the running process gets blocked on I/O or when the running process finishes its compute burst. Assume that all I/O operations can be overlapped as much as possible. For what percentage of time does the CPU remain idle?

A. 0% B. 10.6%

C. 30.0% D. 89.4%

ANSWER KEY

1. B	**2.** B	**3.** B	**4.** C
5. D	**6.** B	**7.** C	**8.** C
9. B	**10.** B	**11.** B	**12.** C
13. A	**14.** B	**15.** C	**16.** D
17. B	**18.** A	**19.** A	**20.** A
21. A	**22.** D	**23.** A	**24.** B
25. C	**26.** A	**27.** C	**28.** A
29. B	**30.** D	**31.** A	**32.** B
33. B	**34.** A	**35.** B	**36.** C
37. C	**38.** B	**39.** A	**40.** D
41. B	**42.** B	**43.** C	**44.** A
45. C	**46.** A	**47.** A	**48.** C
49. B	**50.** D	**51.** C	**52.** A
53. B			

CHAPTER FIVE 5

Entity Relationship Data Model

5.1 Entity Relationship (ER) Model

Entity Relationship Model/ Diagram is a conceptual model.

Entity Relationship Data Model

- It is based on a configuration of our real world perception into object sets called entities and the relationships among these objects.
- A logic tool is used for database scheme design.
- It does not include implementation details.
- It is described by an ER (Entity-Relationship) diagram.

Note

Implementation details are physical and have no place in an ER diagram. The ER diagram displays logical relationship, not physical relationship.

Entity

- An entity is a thing that has an independent existence.
- An entity is described by its attributes.
- An entity is determined by its instantiations.

(Instantiations are particular values for its attributes.)

■ **Example** A customer is an entity with the attributes:
Name
ID No
Address
Telephone No
An account is an entity with the attributes:
Account No
Balance

Entity Set (Entity Type)

- Define a set of entities of the same type (share the same structure)
- Denote by a rectangular box in ER diagram

- Identify entity by a list of attributes placed in ovals
- Identify key attributes (the set of attributes that uniquely identify entity type)

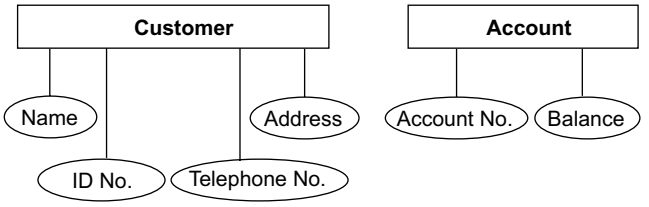


Figure 5.1 An entity representation with its attributes

Relationship and Relationship Set

A relationship is an association among several entities.

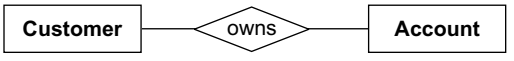
A relationship set is a set of relationships of the same type.

Let $E_1, E_2, \ldots E_n$ be a set of entity sets.

$< e_1, e_2, \ldots, e_n >$ is a relationship, where e_k is contained in E_k.

A subset of $E_1 \times E_2 \times \ldots \times E_n$ is a relationship set.

■ **Example**



Relationship Sets

Defines an association of entity sets.

Is a subset of cartesian product $\mathbf{E}_1 \times \mathbf{E}_2 \times \ldots \times \mathbf{E}_n$.

$\boldsymbol{E}_k$ is said to play a role in the relationship set.

Denoted by a diamond in the ER diagram as shown in Fig. 5.2.

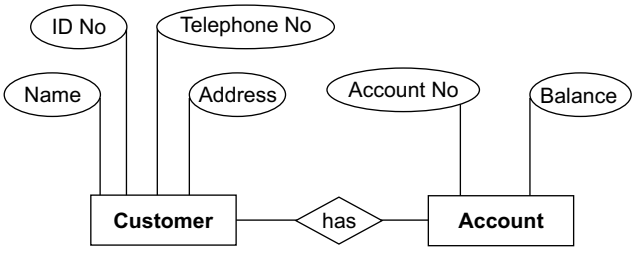


Figure 5.2 A relation

Descriptive Attributes

- Attributes of relationships

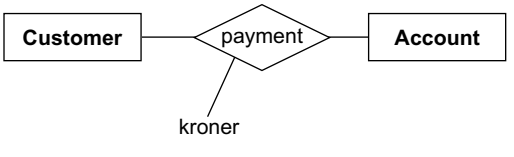


Figure 5.3 Payment relation (Please ask the people to remove the word kroner and the line which joinsit)

- Structural constraints

Degree: number of participating entity sets

Cardinality constraints: {1:1, 1:N, M:N}

Participation constraints: partial or total

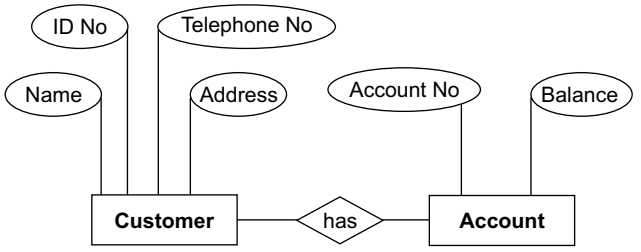


Figure 5.4 Customer and account relation

1:1 each customer has at most one account and each account is owned by at most one customer.
1:N each customer may have any number of accounts but each account is owned by at most one customer.
M:N each customer may have any number of accounts and each account may be owned by any number of customers.

5.1.1 Representing ER Model Using Tables

Basic rules

- One table for one entity set
- One column for one attribute

Superkey
A ***superkey*** is a set of one or more attributes which, taken collectively, allow us to identify uniquely an entity in the entity set.
■ **Example** In the entity set ***customer, customer-name and personal-number*** is a superkey. Note that ***customer-name*** alone is not a superkey, as two customers could have the same name.

Candidate Key
A superkey may contain extraneous attributes, and we are often interested in the smallest superkey. A superkey for which no subset is a superkey is called a ***candidate key.***

■ **Example** ***personal_number*** is a candidate key, as it is minimal and uniquely identifies a customer entity.

Primary Key
Primary key is a candidate key (there may be more than one) chosen by the DB designer to identify entities in an entity set. An entity set that does not possess sufficient attributes to form a primary key is called a ***weak entity set.*** One that does have a primary key is called a ***strong entity set.***

Relation: Customer

Customer No.	Name	Address	Post Code	Gender	Age	Date of Birth
Siddm02	Siddke	48 South St	3070	F	38	04/12/1954
Walsh01	H Walsh	2 Allen Crt	3065	M	44	04/16/1947
Foret13	T Forest	69 Black St	3145	M	24	06/12/1967
Richb76	B Rich	181 Kemp St	3507	M	50	09/12/1941

5.1.1.1 Conceptual Design/Developing the ER Model

- Do not think about the implementation.
- Conceptual model is independent of DBMS.

Basic ER Model Constructs
Entity

- An ***entity*** is an object that exists and is distinguishable from other objects (like a specific person, a company, a book).

Synonyms: Instance (an instantiation), member, individual
An Entity can be uniquely identified as one particular object in the universe. An entity may be **concrete** (like a person, a book) or **abstract** (like a country border, a holiday, etc.)

Entity Sets

- An entity set is a collection of entities of the same type (identified by the chosen properties).

■ **Example** The students at our university.

Synonym: Entity Class

Entity sets need not be disjoint. For example, the entity set ***employee*** and the entity set ***client*** may have members in common.

Entity Types

- **An entity type is defined by its attributes.**

Note that the entity type is **defined** by its attributes, whereas its instantiations are identified by these attributes.

A database is modeled as a collection of entities and relationships among them.

- **An entity set is a collection of entities that share the same properties.**

Synonym: Entity Class

- **An entity type is defined by its properties (attributes).**

A database is modeled as a collection of entities and relationships among them.

Attributes

- **Descriptive properties possessed by all members of an entity set.**

■ **Example** The set of students in a university form an entity set could be named **Student**.

The attributes of the Student could be: **name, ID No, GPA, …**

Formally, an attribute is a function that maps an entity set in a domain.

Every entity is described by a set of pairs in the form (attribute, data value), such as (street, Tröskaregatan), (city, Linköping)

This is akin to Class/Instantiation pattern for objects in OO programming.

Attribute Domain

The domain of an attribute is the set of permitted values for that attribute.

■ **Example** GPA has a range from 1 to 4.

Attribute Species (kinds of)

Simple vs composite
Single valued vs multivalued
Derived
Null valued
Identifier

- **A relationship may also have attributes.**

Defining a Set of ENTITIES and their RELATIONSHIPS

The definition for a set of entities and their relationships depends on how we deal with attributes. Suppose we have an entity set ***employee*** with attributes ***employee-name*** and ***phone-number.*** Is the phone an entity? If so, then we have two entity sets. Also, this new definition allows employees to have 0 or more phones.

This may be a better representation of the way things are.

Does employee have the same status?

Employee-name does NOT have the same status as phone.

The question of what constitutes an entity and what constitutes an attribute depends on how we interpret the world that is being modeled.

Relationships

Relationships associate one or more entities (usually 2).

5.1.1.2. Degree of Relationship

Unary relationship: associates entity with itself
Binary relationship: associates two entities
Ternary relationship: associates three entities
n-ary relationships: associates n entities

Cardinality and Structural Constraint

Relationships between entities have a cardinality associated with them. The [min:max] cardinality together are often referred to as the ***structural constraint.***

Sometimes we need additional attributes for a relationship, which leads to formation of an associative entity type.

Graphical Symbols used in ER diagrams

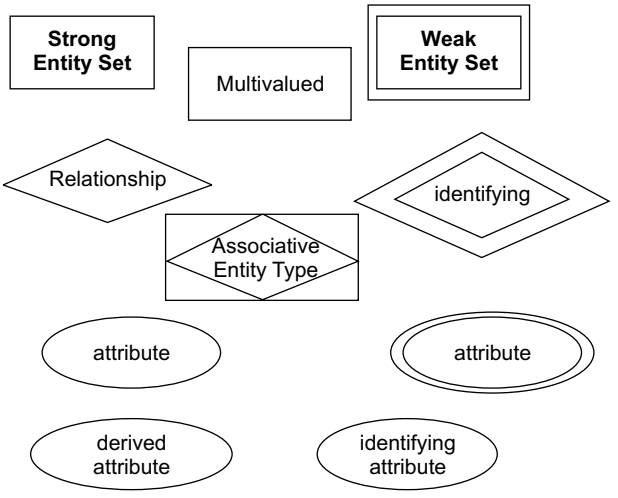


Figure 5.5 Symbols used in representing ER diagrams

Symbols used in representing cardinalities

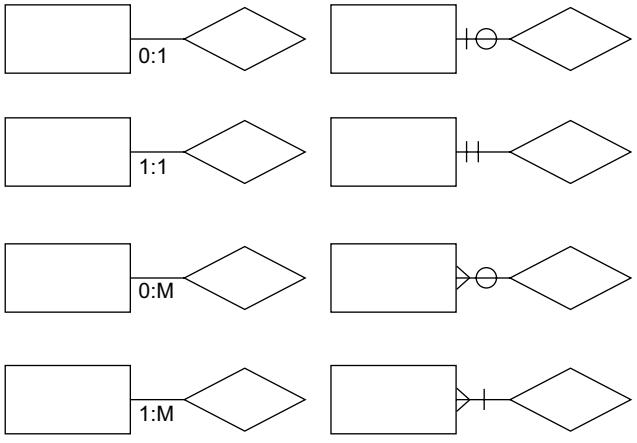


Figure 5.6 Symbolic representation of cardinalities

Existence Dependencies

If the existence of entity X depends on the existence of entity Y then X is said to be existence dependent on Y. (Or we say that Y is the dominant entity and X is the subordinate entity.)

■ **Example** Consider ***account*** and ***transaction*** entity sets, and a relationship ***log*** between them. This is a one-to-many from ***account*** to ***transaction.*** If an account entity is deleted, its associated transaction entities must also be deleted. Thus ***account*** is dominant and ***transaction*** is subordinate.

Strong/Weak Entity Sets

- Weak Entity Set: An entity set whose existence is dependent on one or more other strong entity sets (termed the 'identifying owner(s)')

Thus if an instance of the strong entity set is removed, so must the related instances of the weak entity set.

Strong is dominant.
Weak is subordinate.

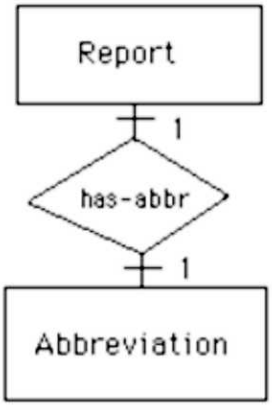
5.1.2 Entity Relationship Data Model

- Based on a configuration of our real world perception into object sets called entities, and the relationships among these objects.
- A logic tool used for database scheme design.
- Does not include implementation details.
- It is described by an ER (Entity-Relationship) diagram.

Note

That implementation details are physical and have no place in an ER diagram. The ER diagram displays logical relationships, not physical relationships.

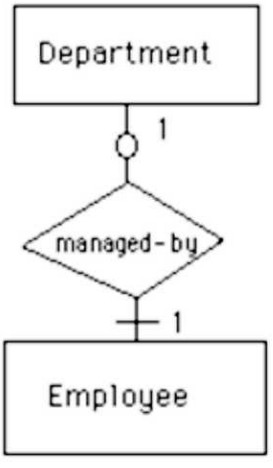
Examples for various cardinalities



(a) one-to-one, both entities mandatory

Every report has one abbreviation, and every abbreviation represents exactly one report.

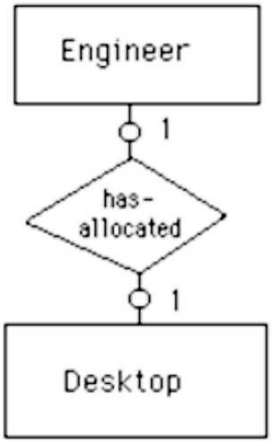
```
create table report
        (report_no  integer,
         report_name  varchar(256),
         primary key(report_no);
create table abbreviation
        (abbr_no  char(6),
         report_no  integer not null unique,
         primary key (abbr_no),
         foreign key (report_no ) references report
           on delete cascade on update cascade);
```



(b) one-to-one, one entity optional, one mandatory

Every department must have a manager, but an employee can be a manager of at most one department

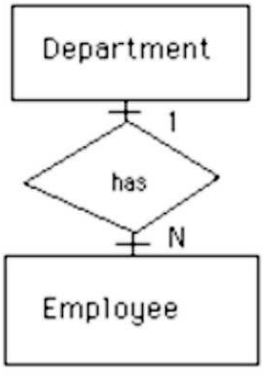
```
create table department
        (dept_no  integer,
         dept_name  char(20),
         mgr_id  char(10) not null unique,
         primary key (dept_no),
         foreign key (mgr_id) references employee
           on delete set default on update cascade);
create table employee
        (emp_id  char(10),
         emp_name  char(20),
         primary key (emp_id));
```



(c) one-to-one,
both entities optional

Some desktop computers are allocated to engineers, but not necessarily to all engineers.

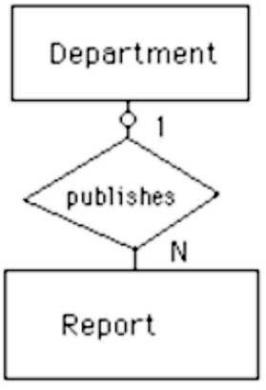
```
create table engineer
        (emp_id  char(10),
         desktop_no  integer,
         primary key (emp_id));

create table desktop
        (desktop_no  integer,
         emp_id  char(10),
         primary key (desktop_no),
         foreign key (emp_id) references engineer
           on delete set null on update cascade);
```



(d) one-to-many, both
entities mandatory

Every employee works in exactly one department, and each department has at least one employee.

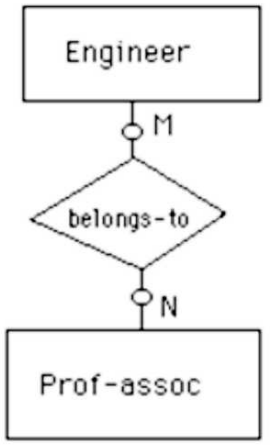
```
create table department
        (dept_no  integer,
         dept_name  char(20),
         primary key (dept_no));
create table employee
        (emp_id  char(10),
         emp_name  char(20),
         dept_no  integer not null,
         primary key (emp_id),
         foreign key (dept_no) references department
           on delete set default on update cascade);
```



(e) one-to-many, one entity
optional, one unknown

Each department publishes one or more reports. A given report may not necessarily be published by a department.

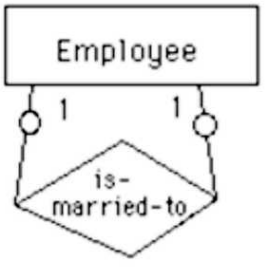
```
create table department
        (dept_no  integer,
         dept_name  char(20),
         primary key (dept_no));
create table report
        (report_no  integer,
         dept_no  integer,
         primary key (report_no),
         foreign key (dept_no) references department
           on delete set null on update cascade);
```



(f) many-to-many, both entities optional

Every professional association could have none, one, or many engineer members. Each engineer could be a member of none, one, or many professional associations.

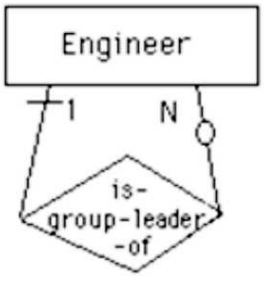
```
create table engineer
       (emp_id  char(10),
        primary key (emp_id));
create table prof_assoc
       (assoc_name  varchar(256),
        primary key (assoc_name));

create table belongs_to
       (emp_id  char(10),
        assoc_name  varchar(256),
        primary key (emp_id, assoc-name),
        foreign key (emp_id) references engineer
          on delete cascade on update cascade,
        foreign key (assoc_name ) references prof-assoc
          on delete cascade on update cascade);
```



(a) one-to-one, both sides optional

Any employee is allowed to be married to another employee in this company.

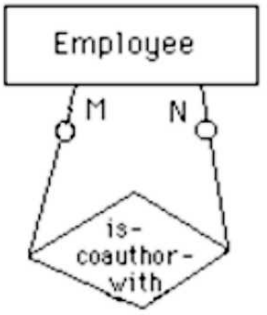
```
create table employee
       (emp_id  char(10),
        emp_name  char(20),
        spouse_id  char(10),
        primary key (emp_id),
        foreign key (spouse_id) references employee
          on delete set null on update cascade);
```



(b) one-to-many, one side mandatory, many side optional

Engineers are divided into groups for certain projects. Each group has a leader.

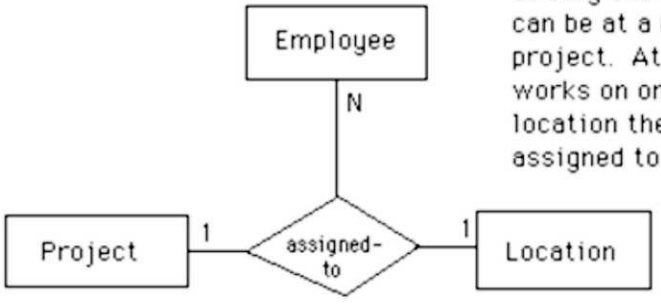
```
create table engineer
       (emp_id  char(10),
        leader_id  char(10) not null,
        primary key (emp_id),
        foreign key (leader_id) references engineer
          on delete set default on update cascade);
```



Each employee has the opportunity to coauthor a report with one or more other employees, or to write the report alone.

```
create table employee
        (emp_id  char(10),
         emp_name  char(20),
         primary key (emp_id));

create table coauthor
        (author_id  char(10),
         coauthor_id  char(10),
         primary key (author_id, coauthor-id),
         foreign key (author_id) references employee
           on delete cascade on update cascade,
         foreign key (coauthor_id) references employee
           on delete cascade on update cascade);
```

(c) many-to-many, both sides optional



Each employee assigned to a project works at only one location for that project, but can be at a different location for a different project. At a given location, an employee works on only one project. At a particular location there can be many employees assigned to a given project.

```
create table employee (emp_id  char(10),
                       emp_name  char(20),
                       primary key (emp_id));
create table project (project_name  char(20),
                      primary key (project_name));
create table location (loc_name  char(15),
                       primary key (loc_name));
create table assigned_to (emp_id  char(10),
                          project_name  char(20),
                          loc_name  char(15) not null,
                          primary key (emp_id, project_name),
                          foreign key (emp_id) references employee
                            on delete cascade on update cascade,
                          foreign key (project_name) references project
                            on delete cascade on update cascade,
                          foreign key (loc_name) references location
                            on delete cascade on update cascade),
                          unique (emp_id, loc_name));
```

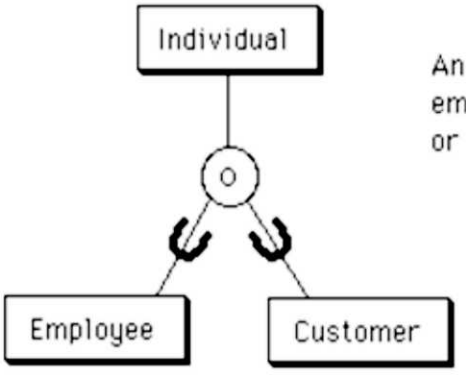
assigned_to

emp_id	project_name	loc_name
48101	forest	B66
48101	ocean	E71
20702	ocean	A12
20702	river	D54
51266	river	G14
51266	ocean	A12
76323	hills	B66

Functional dependencies

emp_id, loc_name -> project_name
emp_id, project_name -> loc_name

(b) one-to-one-to-many ternary relationships



An individual may be either an employee or a customer, or both, or neither.

```
create table individual (indiv_id  char(10),
                         indiv_name  char(20),
                         indiv_addr  char(20),
                         primary key (indiv_id));
create table employee (emp_id  char(10),
                       job_title  char(15),
                       primary key (emp_id),
                       foreign key (emp_id) references individual
                         on delete cascade on update cascade);
create table customer (cust_no  char(10),
                       cust_credit  char(12),
                       primary key (cust_no),
                       foreign key (cust_no) references individual
                         on delete cascade on update cascade);
```

5.1.3 Transforming the Conceptual Data Model to SQL Tables

* **Entity** – directly to a SQL table
* **Many-to-many binary relationship** – directly to a SQL table, taking the 2 primary keys in the 2 entities associated with this relationship as foreign keys in the new table.

* **One-to-many binary relationship** – primary key on "one" side entity copied as a foreign key in the "many" side entity's table.
* **Recursive binary relationship** – same rules as other binary relationships.
* **Ternary relationship** – directly to a SQL table, taking the 3 primary keys of the 3 entities associated with this relationship as foreign keys in the new table.
* A**ttribute of an entity** – directly to be an attribute of the table transformed from this entity.
* **Generalisation super-class (super-type) entity** – directly to a SQL table.
* **Generalisation subclass (subtype) entity** – directly to a SQL table, but with the primary key of its super-class (super-type) propagated down as a foreign key into its table.
* **Mandatory constraint (1 lower bound) on the "one" side of a one-to-many relationship** – the foreign key in the "many" side table associated with the primary key in the "one" side table should be set as "not null" (when the lower bound is 0, nulls are allowed as the default in SQL)

5.1.4 Enhanced ER Model

The following extensions will be added in enhanced-ER model:

1. Class/subclass relationships and type inheritance.
2. Specialisation and generalisation.
3. Constraints on specialisation and generalisation.
4. Union constructs.

5.1.5 File Structures (Sequential files, indexing, B and B + trees)

Read Operating Systems and Data Structures notes

5.2 Introduction to Tuple Relational Calculus (TRC)

Although relational algebra is useful in the analysis of query evaluation, SQL is actualy based on a different query language: relational caclulus. There are two important relational calculus is used in practice namingly 1. Tuple relational calculus and 2. Domain relational calculus. A typical TRC query looks the following:

- **{T/Condition}** - return all tuples T that satisfy the condition **Condition**.
- **{T/R(T)}** - returns all tuples T such that T is a tuple in relation R.
- **{T.name/FACULTY(T) AND T.DeptId = 'CS'}** – returns the values of name field of all faculty tuples with the value 'CS' in their department id field.
  - ⇒ T is the target – a variable that ranges over some relation (its values are tuples of the relation)
  - ⇒ Condition is the body of the query involving T and other variables and evaluates to true or false if a specific tuple is substituted for T.
  - ⇒ The result of a TRC query with respect to a given database is the set of all choices of tuples for the variable T that make the query condition a true statement about the database.
  - ⇒ The variable T is said to be free since it is not bound by a quantifier such as there exists (∃), for all (∀).
  - ⇒ Each variable T ranges over all possible tuples in the universe.
  - ⇒ Tuple variables are also called as range variables.

5.2.1 Relation Algebra vs. Relational Calculus

Although the relational algebra and calculus are equivalent in their expressive power the following important differences can be noted.

- Relational algebra provides a collection of explicit operations – join, union, projection, etc.
- The relational algebriac operations are used to tell the system **how** to build some desired relation in terms of other relations.

- The calculus merely provides a notation for formulating the **definition** of that desired relation in terms of those given relations.
- Relational Algebra is procedural; it is more like a programming language;
- Relational calculus is nonprocedural. it is more close to a natural language.
- The calculus formation is descriptive while the algebraic one is prescriptive.
- SQL does not require the explicit introduction of a tuple variable, it allows the relation name S to serve as an implicit tuple variable.

For example, suppose you want to query:
Get supplier numbers for suppliers who supply part P2.
An algebraic version of this query might follow these steps:

1. Form the natural join of relations SUP (supplier) and P (Part) on S#;
2. Next, restrict the result of that join to tuples for part P2;
3. Finally, project the result of that restriction on S#.

A calculus formulation might look like:
Get S# for suppliers such that there exists a shipment Part with the same S# value and with P# value P2.
That it,
{ t | ∃ s ε SUP(t[S#]=P[S#]) ∧ ∃ u ε P(u[P#]='P2') }

5.2.2 Why it is called relational calculus?

It is founded on a branch of mathematical logic called the predicate calculus.

5.2.3 Relationally Complete Language

A relational query language L is called as relational complete if we can express in L any query that can be expressed in relational calculus.

Formal Definition
Consider a simple relational calculus expression:
{T.name/FACULTY(T) AND T.DeptId='CS'}
which returns the values of name field of all faculty tuples with the value 'CS' in their department id field.
Informally, we need to specify the following information in a TRC expression, namingly:

1. For each tuple variable t, the range relation (for example in the above expression relation FACULTY is the range relation).

   Moreover, variables can be constrained by quantified statements to tuples in a single relation:

- **Esistential Quantifier.** ∃T ε R(cond) will succeed if Cond for atleast one tuple in T is true.
- **Universal Quantifier:** ∃T ε R(Cond) will succeed if Cond succeeds for all tuples in T.
- Any variable that is not bound by a quantifier is said to be free variable otherwise bound variable.
- A TRC expression may contain at most one free variable.

{T.name/FACULTY(T) AND T.DeptId = 'CS'} can be read as: " Find all tuples T such that T is a tuple in the FACULTY relation and the value of DeptId field is 'CS'. Return a tuple with a single field name which is equivalent to the name field of one such T tuple".
The same expression can be alternatively written as:
{R| ∃T ε FACULTY (T.DeptId = 'CS' AND R.name = T.name)} which can be read as " Find all tuples R such that there exists a tuple T in FACULTY with the DeptId field of R is equivalent to the name field of this tuple T}". The same can be alternatively stated as " Find all tuples R that can be obtained by copying the name field of SOME tuple in FACULTY with the value 'CS' in its Dept Id attribute.

Please note that if in the first query instead of T.name if we write only T then the query displays all attributes of the selected tuples.

2. A condition to select particular combinations of tuples. As tuple variables range over their respective range relations, the condition is evaluated for every possible combination of tuples to identify the selected combinations for which

the condition evaluates true. The conditions are also called as functions which are considered to be made of **atoms** (Details are given below).

3. A set of attributes to be retrieved, the requested attributes (in the above example T.name). The values of these attributes are retrieved for each selected combination of tuples.

In addition, the following examples may outline equivalence of TRC and SQL statement.
{T|TEACHING(T) AND T.semister = 'Fall2000'}
is equivalent to
SELECT *
FROM TEACHING T
WHERE T.semister = 'Fall2000';
Here,
Target T corresponds to SELECT list: the query result contains the entire tuple.
TEACHING(T) corresponds to FROM clause, i.e range or domain tuples.
T.semister='Fall2000' corresponds to WHERE clause which is condition.

Atoms

An atom has one one of the following forms

- s ε r where s is a tuple variable and r is a relation.
- s[x] **op** u[y], where s and u are tuple variables, x is an attribute on which s is defined, y is attribute on which u is defined, and op can be >, <, <=, >=, =, ≠.
- s[x] **op** c, where s[x] is as above and c is a constant and op is also same as above.

Formulas are created by joining atoms using the following rules.

- An atom is a formula.
- If P1 is a formula, then so are ¬P1 and (P1)
- If P1 and P2 are formulae then so are P1^P2, P1Vp2, and P1 ⇒ P2.
- If P1(s) is a formula containing a free tuple variable s and r is a relation then ∃ sεr (P1(s)) and ∀s ε r(P1(s)) are also formulae.
- P1^P2 is equivalent to ¬(¬P1, ∨ ¬ P2)
- ∀tε r (P1(t)) is equivalent to ¬ ∃t ε r (¬P1(t))
- P1⇒P2 is equivalent to ¬P1∨P2.

Safety of Expressions

Main drawback of TRC is it may generate an infinite relation. For example, a query like the following may generate infinitley many tuples which may not even appear in the database!.

{t | ¬ (t ε loan)}

Here we are trying to genrate the tuples which are not in the loan relation! Thus, domain of a tuple relational formula is used to solve this problem.

Important Points

- { T| STUDENT(T) AND FAACULTY(T)} will valuate to true if T is a tuple in both the relations. However, this is not possible since the schema of the two relations are different. Two tuples can never be identical.
- If we use attribute which is not available in the tuple then the result is NULL.
- {T|T.A>5} is unbounded expression which is not allowed. All tuple variables should be restricted to the tuples of a specific relation, even if they are not quantified.
- If a tuple variable T is bound to a relation R, then it only has values for the attributes in R. All other attribute values are null.
- A well formed query will have a single unbounded variable. All other variables will have a quantifier over them.
- Bound variables are used to make assertions about tuples in database (used in conditions).
- Free variables designate the tuples to be returned by the query.
- SQL has no quantifiers. Rather it uses some conventions such as:

  - Universal quantifiers are not allowed (but SQL 1999 introduced a limited form).
  - Makes esistential quantifiers implicit: any tuple variable that does not occur in SELECT is assumed to implicitly quantified with ∃.
- Adjacent existential quantifiers and adjacent universal quantifiers commute.
- Adjacent existential and universal quantifiers do not commute.
- A quantifier defines the scope of the quantified variable.
- Relational calculus comes from the first order predicate calculus.
- R(s), where R is a relation name and s is a tuple variable then this atom stands for assertion that s is a tuple in relation R.
- (∃s) (R(s)) says that relation R is not empty. That is, there exists a tuple s in R.
- A free variable is more like a global variable of high level programming languages, that is, a variable defined outside the current procedure. Where as "bound variable" is like a local variable, one that is defined in the procedure at hand and can not be referenced from the outside.
- Relational calculus based languages are higher-level than the algebraic languages. Calculus base languages leaves it to a compiler or interpreter to determine the most efficient order of evaluation.
- Parentheses may be placed around formulas as needed. We assume the order of evaluation of precedence is: arithmetic comparison operators highest, then the quantifiers, then NOT, ∨, ∧, in that order.
- {T| R(T) ∨ S(T)} makes sense if both relations R and S are of same arity.
- {T| R(T) ∧¬S(T)} indicates set difference R-S if both relations are of same arity.
- {T| (∃u) (R(T) ^ R(u) ^ (t[1] ≠ u[1] ≠ t[2] ≠ u[2])) } denotes R if R has two or more members and denotes the empty relation if R is empty or has only one member.
- If E is a relational algebraic expression then there is a safe expression in TRC equivalent to E.
- Let A (branch_name, loan_no, amount), B (cust_name, loan_no) are two relations. The following TRC query displays names of all customers who have loan from "Delhi" branch.

  {T|∃s ε B(T[cust_name] = s[cust_name]

  ∧∃ u ε A (u[loan_no] = s[loan_no] ∧ u[branch_name] = 'Delhi')) }
- The following displays names of faculty who belongs to CS department.

  {T|∃R ε FACULTY (R.DeptId = 'CS' ∧ T.name = R.name) }

  or

  {T.name| FACULTY(T) ∧ T.DeptId = 'CS' }
- The following TRC query may display name, social security number of those people who are staff and simultaneously students.

  {T|STUDENT(T) ∧∃ R ε FACULTY(T[SSN]=R[SSN] ∧ T[Name]=R[Name]) }
- Let A (cust_name, accountno), B (cust_name, loan_no) are two relations. The following TRC query displays names of all customers who have both loan and account

  {T|∃s ε B(T[cust_name]=s[cust_name] ∧ ∃ u ε A (T[cust_name]=u[cust_name])) }
- Let A (cust_name, account no), B (cust_name, loan_no) are two relations. The following TRC query displays names of all customers who have loan or account

  {T|∃s ε B(T[cust_name]=s[cust_name] ∧¬ u ε A (T[cust_name]=u[cust_name])) }
- Let A (cust_name, accountno), B (cust_name, loan_no) are two relations. The following TRC query displays names of all customers who have loan but not account

  {T|∃s ε B(T[cust_name]=s[cust_name] ∧¬∃ u ε A(T[cust_name]=u[cust_name])) }
- Let A(cust_name, accountno), B(cust_name, loan_no) are two relations. The following TRC query displays names of all customers who have account but no loan.

  {T|∃s ε A(T[cust_name]=s[cust_name] ∧¬∃ u ε B(T[cust_name]=u[cust_name])) }
- {T|∃F ε FACULTY(∃C ε CLASS(F.Id=C.Instructor.Id ∧ C.Year='2002' ∧ T.Name=F.Name ∧ T.Course_code=C. Course_Code))}

The above command displays faculty names, Course codes who taught in 2002.

- Write equivalent TRC queries for the following SQL statement :

```
SELECT DISTINCT F.Name
FROM FACULTY F
WHERE NOT EXISTS
  (SELECT * FROM CLASS C
  WHERE F.Id = C.InstructorId AND C.Year = '2002');
```

$\{F.Name|FACULTY(F) \wedge \neg(\exists C \varepsilon CLASS (F.Id=C.InstructorId \wedge C.Year='2002'))\}$

or

$\{F.Name|FACULTY(F) \wedge (\forall C \varepsilon CLASS (F.Id <> C.InstructorId \vee C.Year <> '2002')) \}$

- Find all students who have taken all the courses required by 'CSE432'.

  $\{S.Name \mid STUDENTS(S) \wedge \forall R \varepsilon REQUIRES(R.CrsCode <> 'CSC432' \vee (\exists T \varepsilon TRANSCRIPT(T.StudId=S.StudId \wedge T.CrsCode= R.PreReqCrsCode))\}$

- Find all students (names) who have never taken a course from 'Acorn'.

  $\{S.Name \mid STUDENTS(S) \wedge \forall C \varepsilon CLASS(\exists F \varepsilon FACULTY (F.Id=C.InstructorId \wedge (\neg(F.Name='Acorn' \vee \neg(\exists T \varepsilon TRANSCRIPT(S.Id=T.StudId \wedge C.CrsCode=T.CrsCode \wedge C.Year=T.Year))))) \}$

- Find all course tuples corresponding to all the courses that have been taken by all students.

  $\{E.Name \mid COURSE(E) \wedge \forall S \varepsilon STUDENT(\exists T \varepsilon TRANSCRIPT(T.StudId = S.StudId \wedge E.CrsCode= T. CrsCode)) \}$

- Find all students. Who has registered for course CS308

  $\{S \mid STUDENT(S) \wedge (\exists T \varepsilon TRANSCRIPT(S.Id = TStudId \wedge T.CrsCode= 'CS308')) \}$

5.2.4 Domain Relational Calculus

This is second form of relational calculus which uses domain variables that take on values from attributes domain, rather than values for an entire tuple. This is however, clisely related to TRC.

Formal Definition
An expression in DRC is expressed as

$\{<x1,x2,\ldots xn> | P(x1,x2,\ldots.xn)\}$

$<x1,x2,\ldots,xn>$ represented domain variables and P represents a formula composed of atoms, which are same as TRC.

Atoms
An atom has one one of the following forms

- $<x1,x2,\ldots xn> \varepsilon$ r where r is a relation on n attributes.
- s **op** u, where s and u are domain variables, and op can be >, <, <=, >=, =, ≠. We require that s and u have domains that can be compared by the above operations.
- s **op** c, where s is as domain variable and c is a constant and op is also same as above.

Formulas are created by joining atoms using the following rules.

- An atom is a formula.
- If P1 is a formula, then so are ¬P1 and (P1)
- If P1 and P2 are formulae then so are P1^P2, P1Vp2, and P1⇒ P2.

Find the branch name, loan number and amount for loans over 1300.

$\{<b,l,a> | <b,l,a> \varepsilon$ loan $\wedge$ a>1300$\}$

Find the names all customers who have a loan, an account or both at "Delhi" branch.

$\{<c> \mid \exists l (<c,l> \varepsilon$ borrower
$\wedge \exists b,a (<b,l,a> \varepsilon$ loan $\wedge$b="Delhi"))
$\vee \exists a (<c,a> \varepsilon$ depositor
$\wedge \exists b,n (<b,a,n> \varepsilon$ amount $\wedge$ b="Delhi"))$\}$

Tuple Relational Calculus

RA vs. TRC

- **Selection :**
  Algebra : $\sigma_{Cond}(R)$
  Calculus : $\{T | R(T)$ AND $Cond(T)\}$, i.e. replace attributes A in $Cond$ with $T.A$ to obtain $Cond(T)$.
- **Projection :**
  Algebra : $\Pi_{A_1,....,A_k}(R)$
  Calculus : $\{T | A_1,.... T.A_k | R(T)\}$
- **Cartesion Product:** Given $R(A_1,...., A_n)$ and $S(B_1,...., A_m)$
  Algebra : $R \times S$
  Calculus : $\{T | \exists T1 \in R, \exists T2 \in R ($
  $T.A_1 = T1.A_1$ AND $\cdots$ AND $T.A_n = T1.A_n$ AND
  $T.B_1 = T2.B_1$ AND $\cdots$ AND $T.B_m = T2.B_m)\}$
- **Union:**
  Algebra : $R \cup S$
  Calculus : $\{T | R(T)$ AND $S(T)$
- **Set Difference:**
  Algebra : $R - S$
  Calculus : $\{T | R(T)$ AND $\forall T1 \in S, (T1 <> T)\}$
  where $T <> T1$ is a shorthand for
  $T.A_1 <> T1.A_1$ OR $\cdots$ ORT. $A_n <> T1.A_n$

5.3 Integrity Constraints

- One row of a relation is called as a tuple
- Domain of an attribute is the set of values which it can take
- Domain Integrity Constraints : While developing the database system we can include a integrity constraint such that a specified attribute in a table will be made to accept either a set of range of values or a set of values.

  While we insert a new record, this constraint will be validated by the database system before accepting
- Attribute type Integrity Constraint : Normally while creating the tables every database management system supports freedom to specify the type of the attributes i.e, integer type or date type or string type etc.

  When we propagate (insertions) the database automatically enforces the type.
- Arity of a relation is the no of attributes.
- Cardinality means the no of tuples in that relation instance
- Candidate Key : Normally Candidate keys are the minimal subset of the attribute Set.

  For a relation, more than one candidate key can exist.

  One of the candidate keys is taken as Primary key. Then other keys become the Alternate keys for that relation. The attributes which are members of the candidate keys are called as Prime attributes. Primary Key is the one which is employed during the storage, retrieval of the Records

  If we happen to have more than one candidate key for a table, their selection is based on how they make the logical records physically distributed and what is its consequence on access times.

  * Super Key : is the one which may not be minimal set.

    If we add one attribute to a Candidate key or a primary key, the resulting set will be obviously a key and it is more appropriate to call it as super key.

    For a table if there exists at least one candidate key then, all the attributes of that table makes the largest possible super key.

If for a table there is no candidate key exist for a table then there will not be any super key also.

* Foreign Key :

Dept (Dept_id, Dept_name, Location)

10	Sales	Miami
20	Purchase	AKP

Emp (Empno, ename, emp_mgr, title, Emp_dept)

111	10
113	17
114	NULL

In the above tables, for the first table Dept_id is the primary key and Emp_dept is the foreign key for the second table.

If it is so, first tuple's insertion in the second Table is accepted as the last attribute value is in the domain of the Dept_id of the First table.

The insertion of the second tuple is not accepted as the attr value of employee dept 17 is not in the domain of Dept_id and is also not null.

The third tuples insertion is also accepted as the Emp_dept can take NULL.

This is known as Referential Integrity.

The foreign key can even exist for a single table also. For example, in the above Emp Table, Mgr_Id can be considered as a foreign key as the manager should also be an employee. Thus, its probable values (domain) will be same as the domain of the Emp_id.

5.4 Database Design and Normalisation

During the database design, we will be carrying out analysis of the tables which can be called as Normalisation.

The Normalisation is especially meant to eliminate the following anomalies,

(i) Insertion anomaly
(ii) Deletion anomaly
(iii) Update anomaly
(iv) Join anomaly

Even the objective of the normalisation includes elimination of redundancy in Database tables.

Redundancy may be one reason for some type of anomalies such as update anomalies.

Normalisation is even employed to impose some integrity constraints.

Goals of normalisation

1. Integrity
2. Maintainability

Side effects of normalisation

- Reduced storage space required (usually, but it could increase)
- Simpler queries (sometimes, but some could be more complex)
- Simpler updates (sometimes, but some could be more complex)

5.4.1 Functional Dependencies

If X, Y are two attribute sets, R is the relation then in this relation the FD

$X \rightarrow Y$ is said to be existing if for any two tuples t1, t2 if

T1 [x] = T2 [x] implies T1 [y] = T2 [y]

X functionally determines Y (or) Y is functionally dependent on X.

■ **Example** Emp (ID, Name, Dept, Grade, Sal, Age, Addr)

In the above table the functional dependency ID → Name is very well valid for this relation. However, Name → ID may not be a valid dependency, as there is a possibility that there can be two employees having same names but differ in ID no's.

* If R satisfies X → Y then PI Z (R) also satisfies X → → Y if X, Y subset of Z

■ **Example** Order (Order_no, part, supplier_name, Supplier_addr, Qty, Price)

Order_no Supplier_name → Supplier_addr

In the above table though the functional dependency is valid one. However, the supplier address is very much depends on Supplier name rather than Order no.

Thus, we can say Supplier_addr partially depends on the order no and Supplier Name

- If F is the set of functional dependencies meaningful in the relation and " f " is one FD from F then its lowercase " f " can be said as redundant FD if the set of FD's F – { f } implies the FD " f ".

■ **Example** In the FD set F given by X → Y

Y → Z

X → Z

X → Z is a redundant FD.

- The set of all FD's implied by " F " is called as Closure of F and is denoted as F^+
- If F^+ is same as F then F is called as full family of dependencies.

■ **Example** ID → Name
ID → Dept
ID Grade Age → Salary
ID → Age
ID → Address

{ ID } +
X = { ID }
X = { ID, Name }
X = { ID, Name, Dept }
X = { ID, Name, Dept, Age, Addr }
X = { ID, Name, Dept, Age, Addr, Grade, Salary }

- If X is a key for a database then X^+, contains all the attributes of that relation.

Superkey Rule 1. Any FD involving all attributes of a table defines a super-key on the LHS of the FD.

Given: Any FD containing all attributes in the table R(W, X, Y, Z) i.e., XY → WZ.
Proof:

(1) XY → WZ given
(2) XY → XY by the reflexivity axiom
(3) XY → XYWZ by the union axiom
(4) XY uniquely determines every attribute in table R, as shown in (3)
(5) XY uniquely defines table R, by the definition of a table as having no duplicate rows
(6) XY is therefore a super-key, by the definition of a super-key.

Super-key Rule 2. Any attribute that functionally determines a Super-key of a table, is also a super-key for that table.

Given: Attribute A is a super-key for table R(A, B, C, D, E) and E → A.
Proof:

(1) Attribute A uniquely defines each row in table R, by the def. of a super-key
(2) A → ABCDE by the definition of a super-key and a relational table
(3) E → A given
(4) E → ABCDE by the transitivity axiom
(5) E is a super-key for table R, by the definition of a super-key.

- Algorithm to find out whether a given FD X → Y is Valid in a given relation or not
  (i) Project X, Y from R
  (ii) Sort the tuples of the table using { X }
  (iii) Check every adjacent tuple X values and if they are same then check their Y values if they are matching continue else fail.
  O (n log n) for sorting
  O (n) for comparison where 'n' is the cardinality of the projected relation
  So, Time Complexity is O (n log n)

5.4.2 Armstrong's Axioms

(i) If Y subset of X then X → Y
(ii) If X → Y then XW → YW
(iii) If X → Y, Y→ Z then X → Z
(iv) If X → Y, YW → Z then XW → Z
(v) If X → Z, X → Y then X → YZ
(vi) If X → YZ then X → Y, X → Z

The FD of the first rule type is said to be trivial FD.

A FD X → Y is said to be redundant if the remaining FD's logically implies This FD X → Y.

One way to find that the FD is redundant or not is to prove that by using all of the inference axioms and the remaining FD's, this FD is implied.

The Set of inference axioms are complete and sound. Here the Complete indicates they can be used to enumerate all the possible acceptable FD's. Here the Sound indicates they will not produce or derive non-acceptable Functional dependencies.

■ **Example** F = { X → YW, XW → Z, Z → Y, XY → Z }
Find whether XY → Z is redundant or not ?

G = { X → YW, XW → Z, Z → Y }

T1 = { XY }

T2 = { XYW }

T3 = { XYWZ }

As the dependent of XY i.e, Z belongs to T3.
So, XY → Z is redundant.

- If G = Phi (empty) then we can conclude that XY → Z is not redundant.
- If we go on remove the redundant FD from a FD set till we cannot remove any more FD, the remaining set of FD's is called as Non- Redundant Cover of that relation.
- Though the Cover is Non-Redundant, the FD's may have some extraneous attributes either left side or right side. Removing these extraneous attributes. Also is very much needed. This operation is called as left Reduction, right Reduction, respectively.
- Let F is a Set of FD, XY → W is one FD, then X variable can be said as Extraneous if F – { XY → W } + { Y → W } explains the same cover.
- In order to check X → Y is redundant or not the following steps has to be taken,
  (i) Have G = F – { X → Y }
  (ii) T = { X }
  (iii) For each FD A → B in G do the following
  If A is subset of T T = T union { B }
  If Y is subset of T then X → Y can be said as redundant
  Else Remove A→ B from G.

(iv) if G is NULL then X→ Y can be said as redundant.

* If Covers before and after removal of an attribute X on the left side of the FD are same then X can be said as extraneous otherwise not.

■ **Example** Find Non – Redundant Cover of

$F = \{ X \to Y, Y \to Z, Z \to Y, X \to Z, Z \to X \}$

Take $X \to Y$:

$G = \{ Y \to X, Y \to Z, Z \to Y, X \to Z, Z \to X \}$

As $X^+ = \{ XYZ \}$ contains Y so, $X \to Y$ is redundant.

Take $Y \to X$:

$G = \{ Y \to Z, Z \to Y, X \to Z, Z \to X \}$

As Y+ = { XYZ } contain X so, $Y \to X$ is redundant.

Take $Y \to Z$:

$G = \{ Z \to Y, X \to Z, Z \to X \}$

As Y+ = {Y} do not contain Z. so, $Y \to Z$ is not redundant.

Similar procedure clearly confirms that $Z \to Y$, $X \to Z$, $Z \to X$ are not redundant.

Final FD set = $\{ Y \to Z, Z \to Y, X \to Z, Z \to X \}$

■ **Example** $F = \{X \to Z, XY \to WP, XY \to ZWQ, XZ \to R\}$

Selecting $XY \to WP$:

Choose X as extraneous.

Y+ = { Y }, WP not subset of Y+. So, X is not extraneous.

Choose Y as extraneous.

X+ = { XZR }, WP not subset of X +, Y is also not extraneous.

Selecting $XY \to ZWQ$:

Similarly, X, Y are not extraneous.

Selecting $XZ \to R$:

X is not extraneous and Z is extraneous.

So, left reduced final set = $\{ X \to Z, XY \to WP, XY \to ZWQ, X \to R \}$

- Minimal Cover or Canonical Cover : is the one in which,

(i) Every FD is simple (RHS of any FD should have single attribute)

(ii) It is left reduced

(iii) It is non-redundant

■ **Example** Find the minimal cover for the following set of functional dependencies.

$A \to B\ C$

$AC \to D$

$D \to A\ B$

$A\ B \to D$

In $AB \to D$, B is extraneous

In $AC \to D$, C is extraneous

$A \to B\ C$, $A \to D$, $D \to A\ B$ are left reduced.

$\{A \to D, D \to B, A \to C, D \to A\}$ is the minimal cover.

{A }, {D } are the candidate keys.

In the above set of functional dependencies A,D are prime attributes.

■ **Example** $A \to B\ C$, $B \to C$, $A \to B$, $A\ B \to C$. Find the minimal cover ?

In $A\ B \to C$, B is extraneous.

$\{ A \to B, B \to C \}$ is the minimal cover.

{ A } is the primary key.

* Normalisation : is especially aimed at to make the relations to be free from undesirable anamolies such as insertion anamoly or updation or deletion anamoly.

Normalisation is to obtain powerful relational retrieval algorithms which are based on a collection of relational primitive operators.

Normalisation is also to reduce the need for restructuring of the relations as new datatypes are added.

- If a table contains entries with multi values then the table is said to be unnormalised.
- Normally, such a type of table can be normalized by flattening the table i.e, for each value of an entry which is having multiple values we create a new tuple by simply copying the other attribute values as same.
- We can bring an unnormalised table with entries multiple values to 1 NF by either flattening the table or by decomposing the table.

A table is said to be in 2 NF if

(i) it is in 1 NF

(ii) no non-prime attribute is partially dependent on key or each non-prime attribute should fully dependent on every candidate key.

- A table is said to be in 3 NF if

(i) The relation should be in 2 NF

(ii) No non-prime attribute functionally determines any other non-prime attributes

- Boyce-Codd normal form (BCNF)

A table is in BCNF if, for every nontrivial FD X → A,

(1) attribute X is a super-key.

TABLE PART

PNUM	PNAME	WT
P1	NUT	12
P2	BOLT	17
P3	WRENCH	17
P4	WRENCH	24
P5	CLAMP	12
P6	LEVEL	19

We find the following FD as valid in this relation.

PNUM → PNAME, WT

Also, PNUM is key. Therefore, it is in 3NF.

■ **Example**

Table P1. R (X, Y, Z) is decomposed into R1 (X, Y) R2 (Y, Z)

F D = { X → Y, Z → Y }

	X	Y	Z
R1	A1	A2	B13
R2	B21	A2	A3

This is a lossy decomposition.

- If at all any row contains all A's then it is lossless decomposition.

R (X, Y, Z, W, P, Q) is decomposed into R1 (Z,P,Q) R2 (X, Y, W, P, Q).

FD = { XY → W, XW → P, PQ → Z, XY → Q }

	X	Y	Z	W	P	Q
R1	B1	B12	A3	B14	A5	A6
R2	A1	A2	A3	A4	A5	A6

This is Lossless decomposition.

- This algorithm is an iterative one and you proceed till there is no change in the table and it does not matter with the order of the FD's taken, result is same.

- While carrying out 2 NF Normalisation, we have to first of all find out all possible Candidate keys (Prime attrs) and Non-prime attrs. Also, we have to find out whether there are any partial dependencies of the non-prime attrs. Then, we decompose this table such that all fully dependent attrs along with Prime attrs into one table and partially dependent Non-Prime attrs, the attrs on which they depend into another table. This guarantees 2 NF requirements However, transitive dependencies may exist.
- ABU algorithm is very much suitable to identify whether the decomposition is Lossless or not. Especially, if finally there is no row having all a's in the above tables then we can say that the decomposition is lossy. The reverse is not strong, rather we have to apply extra data dependency analysis.

1. R (A, B, C) FD = { A → B, B → C }
A is the key for the database and all the non-prime attrs depends fully on A.
Thus, it is in 2 NF.
However, A → C is a transitive dependency. Thus, it cannot be in 3 NF.

2. R (X, Y, Z,W) FD = { Y → W, W → Y, X Y → Z }
XY is a Key.
XW is a Key.
So, XYW are Prime attributes.
Z is non- prime attribute
The relation is in 3 NF.

3. R (A,B,C, D, E) FD = { AB → CE, E → AB, C → D }
Keys : { AB }, {E }
This relation is in 2 NF but not 3 NF.

4. Emp (Id, Name, Dept, Hrly_rate)
Dept → Hrly_rate
Id → Dept
Id → Name
This relation is in 1 NF but not in 2 NF, not in 3 NF.

- Every 2 attribute relation is in BCNF.

5. Emp (ID, Name, Dept, Hrly_rate)
ID → Dept, Dept → Hrly_rate
This relation is in 2 NF.

6. Order (SSN, PNO, HRS, ENAME, PNAME, PLOC)
SSN PNO → HRS
SSN → ENAME
PNO → PNAME
PNO → PLOC
Key is (SSN, PNO)
HRS is fully dependent on SSN. ENAME is partially dependent on SSN.
So, this is not in 2 NF.

- Inorder to bring a table to 2 NF, we can decompose the table such that in one table all prime attrs and in another table non-prime attrs and the prime attrs on which they depend.

■ **Example** 2NF decomposition of the above relation is given by
(SSN, PNO, HRS)
(SSN, ENAME)
(PNO, PNAME, PLOC)

7. Emp (SSN, ENAME, BDATE, ADDR, DNO, DNAME, DMGRSSN)
DNO → DNAME
DNO → DMGRSSN

- In order to get the table into 3 NF, we can decompose such that in one table key, all those attrs which fully depends on a key are kept and in other table, we may have the attrs which depends transitively on the key and the attributes via which transitive dependency exists.

■ **Example** The 3 NF decomposition of the above relation is given by

(SSN, ENAME, BDATE, ADDR, DNO)

(DNO, DNAME, DMGRSSN)

- Projection of a set of FD's on to a set of attributes is defined as those set of FD's which are valid with this selected set of attrs.

Let F be projection onto a set of attributes T denoted as PI_T (F)

= { X → Y belongs to F+ / XY subset of T }

Algorithm:

X → Y

X, Y can be composite attributes whose union should be proper subset of T.

To calculate Projection Consider all proper subsets X of T (X subset of T, X !=T)

That appear as the determinant of FD's.

For each Set

1. Calculate X +.
2. For each set of attributes of Y of X+ that satisfies simultaneously the following conditions

A. Y subset of T

b. Y subset of X+

c. Y not equal to T

include X → Y as one of the FD's of Projection of FD's on T.

■ **Example** R (X, Y, Z, W, Q)

FD = { XY → WQ, Z → Q, W → Z, Q → X }

T = { X, Y, Z }

{X }, { Y }, {Z }, {XY }, {XZ }, { YZ }

{X}+ = {X }

{Y}+ = { Y }

{ Z }+ = { ZQX }

so, Z → X is implied.

{ XY } + = { XYZWQ }

so, XY → Z is acceptable

{ YZ } + = { YZWQX }

so, YZ → X is implied

{ XZ } + = { XZQ }

The projected FD's = { XY→ Z, Z → X }

- If a constraint on a relation R states that there cannot be more than one tuple with A given X value then X can be called as a Candidate key.
- If a Primary Key contains a single attribute then the relation can evidently be in 2 NF.

1. R (A, B, C, D)

FD = { AB → C, BC → D }

{AB } is the key.

The largest acceptable normal state is 2 NF.

2. R (A, B, C)

A → B, B → C What are the anomalies you will have?

There is no guarantee always that when we bring a table from 1 NF to 2 NF all the anomalies are eliminated. Only redundancy is reduced.

Under some special conditions in 3 NF also we may find insertion, updation anomalies etc.

■ **Example** Given a set of FDs H, determine a minimal set of tables in 3NF, while preserving all FDs and maintaining only lossless decomposition/joins.

H:	AB →C	DM→NP	D→KL
	A→DEFG	D→M	
	E→G	L→D	
	F→DJ	PR→S	
	G→DI	PQR→ST	

Step 1: Eliminate any extraneous attributes in the left hand sides of the FDs. We want to reduce the left hand sides of as many FDs as possible.

In general: XY→Z and X→Z ⇒ Y is extraneous **(Reduction Rule 1)**

XYZ→W and X→Y ⇒ Y is extraneous **(Reduction Rule 2)**

For this example we mix left side reduction with the union and decomposition axioms:

DM → NP ⇒ D → NP ⇒ D → MNP

D → M D → M

PQR → ST ⇒ PQR → S, PQR → T ⇒ PQR → .T

PR → S PR → S PR → S

Step 2: Find a non-redundant cover H' of H, i.e. eliminate any FD derivable from others in H using the inference rules (most frequently the transitivity axiom).

A → E → G ⇒ eliminate A → G from the cover

A → F → D ⇒ eliminate A → D from the cover

Step 3: Partition H' into tables such that all FDs with the same left side are in one table, thus eliminating any non-fully functional FDs. (Note: creating tables at this point would be a feasible solution for 3NF, but not necessarily minimal.)

R1: AB → C	R4: G → DI	R7: L → D
R2: A → EF	R5: F → DJ	R8: PQR → T
R3: E → G	R6: D → KLMNP	R9: PR → S

Step 4: Merge equivalent keys, i.e., merge tables where all FD's satisfy 3NF.

4.1 Write out the closure of all LHS attributes resulting from Step 3, based on transitivities.

4.2 Using the closures, find tables that are subsets of other groups and try to merge them. Use Rule 1 and Rule 2 to establish if the merge will result in FDs with super-keys on the LHS. If not, try using the axioms to modify the FDs to fit the definition of super-keys.

4.3 After the subsets are exhausted, look for any overlaps among tables and apply Rules 1 and 2 (and the axioms) again.

In this example, note that R7 (L→D) has a subset of the attributes of R6 (D→KLMNP). Therefore, we merge to a single table with FDs D →KLMNP, L→D because it satisfies 3NF: D is a super-key by Rule 1 and L is a super-key by Rule 2.

Final 3NF (and BCNF) table attributes, FDs, and candidate keys:

R1: ABC (AB → C with key AB)	R5:DFJ (F → DJ with key F)
R2: AEF (A → EF with key A)	R6: DKLMNP (D → KLMNP, L → D, w/keys D, L)
R3: EG (E → G with key E)	R7: PQRT (PQR → T with key PQR)
R4: DGI (G → DI with key G)	R8: PRS (PR → S with key PR)

Step 4a. Check to see whether all tables are also BCNF. For any table that is not BCNF, add the appropriate partially redundant table to eliminate the delete anomaly.

5.5 Transactions and Concurrency Control

5.5.1 What is a Transaction?

- A transaction is a logical unit of work –

  It may consist of a simple SELECT to generate a list of table contents, or a series of related UPDATE command sequences.

  A **database request** is the equivalent of a single SQL statement in an application program or transaction.

  - Must be either entirely completed or aborted –

    To sell a product to a customer, the transaction includes updating the inventory by subtracting the number of units sold from the PRODUCT table's available quantity on hand, and updating the accounts receivable table in order to bill the customer later.

  - No intermediate states are acceptable –

    Updating only the inventory or only the accounts receivable is not acceptable.

- **Example Transaction –**

  Consider a transaction that updates a database table by subtracting 10 (units sold) from an already stored value of 40 (units in stock), which leaves 30 (units of stock in inventory).

  - A **consistent database state** is one in which all data integrity constraints are satisfied.
  - At this transaction is taking place, the DBMS must ensure that no other transaction access X.

  Evaluating Transaction Results

- Examine current account balance

```
SELECT ACC_NUM, ACC_BALANCE
FROM CHECKACC
WHERE ACC_NUM = '0908110638';
```

  - SQL code represents a transaction because of accessing the database
  - Consistent state after transaction
  - No changes made to Database

- Register credit sale of 100 units of product X to customer Y for $500:

  Reducing product X's quality on and (QOH) by 100

```
UPDATE PRODUCT
SET PROD_QOH = PROD_QOH - 100
WHERE PROD_CODE = 'X';
```

  Adding $500 to customer Y's accounts receivable

```
UPDATE ACCT_RECEIVABLE
SET ACCT_BALANCE = ACCT_BALANCE + 500
WHERE ACCT_NUM = 'Y';
```

  - If both transactions are not completely executed, the transaction yields an inconsistent database.
  - Consistent state only if both transactions are fully completed
  - DBMS doesn't guarantee transaction represents real-world event but it must be able to recover the database to a previous consistent state. (For instance, the accountant inputs a wrong amount.)

5.5.2 Transaction Properties

All transactions must display **atomicity**, **durability**, **serialisability**, and **isolation**.

Atomicity

The Atomicity property of a transaction implies that it will run to completion as an indivisible unit, at the end of which either no changes have occurred to the database or the database has been changed in a consistent manner.

The basic idea behind ensuring atomicity is as follows. The database keeps a track of the old values of any database on which a transaction performs a write, and if the transaction does not complete its execution, the old values are restored to make it appear as though the transaction never executed.

Ensuring atomicity is the responsibility of the database system itself; it is handled by a component called the **Transaction Management Component.**

Consistency

The consistency property of a transaction implies that if the database was in a consistent state before the start of a transaction, then on termination of the transaction, the database will also be in a consistent state.

Ensuring consistency for an individual transaction is the responsibility of the Application Manager who codes the transaction.

Isolation

The isolation property of a transaction ensures that the concurrent execution of transactions results in a system state that is equivalent to a state that could have been obtained had these transactions executed one at a time in same order.

Thus, in a way it means that the actions performed by a transaction will be isolated or hidden from outside the transaction until the transaction terminates.

This property gives the transaction a measure of relative independence.

Ensuring the isolation property is the responsibility of a component of a database system called the Concurrency Control Component.

Durability

The durability property guarantees that, once a transaction completes successfully, all the updates that it carried out on the database persists even if there is a system failure after the transaction completes execution.

Durability can be guaranteed by ensuring that either:

(a) The updates carried out by the transaction have been written to the disk before the transaction completes.

(b) Information about updates carried out by the transaction and written to the disk is sufficient to enable the database to re-construct the updates when the database system is restored after the failure.

Ensuring durability is the responsibility of the component of the DBMS called the Recovery Management Component.

In a nutshell,:

- Atomicity –
  - All transaction operations must be completed

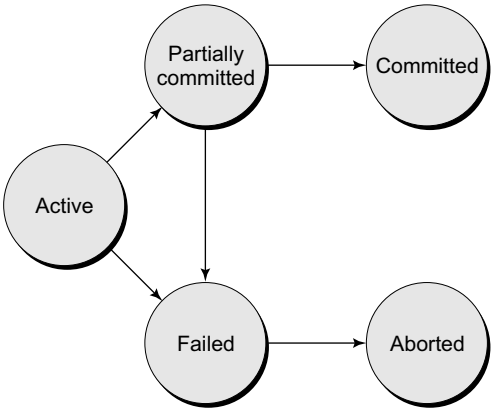
Incomplete transactions aborted

- Durability –
  - Permanence of consistent database state
- Serialisability –
  - Conducts transactions in serial order
  - Important in multi-user and distributed databases
- Isolation
  - Transaction data cannot be reused until its execution complete
- Consistency – (To preserve integrity of data, the database system must ensure: **atomicity, consistency, isolation, and durability (ACID).)**

Execution of a transaction in isolation preserves the consistency of the database.

- A single-user database system automatically ensures serialisability and isolation of the database because only one transaction is executed at a time.
- The atomicity and durability of transactions must be guaranteed by the single-user DBMS.

- The multi-user DBMS must implement controls to ensure serialisability and isolation of transactions – in addition to atomicity and durability – in order to guard the database's consistency and integrity.



5.5.3 Transaction State

- **Active**, the initial state; the transaction stays in this state while it is executing.
- **Partially committed,** after the final statement has been executed.
- **Failed,** after the discovery that normal execution can no longer proceed.
- **Aborted**, after the transaction has been rolled back and the database restored to its state prior to the start of the transaction. Two options after it has been aborted:
  - **Restart** the transaction – only if no internal logical error but hardware or software failure.
  - **Kill** the transaction – once internal logical error occurs like incorrect data input.
- **Committed**, after successful completion. The transaction is terminated once it is aborted or committed.

5.5.4 Transaction Management with SQL

- Defined by ANSI, the standards of SQL transaction support: `COMMIT` & `ROLLBACK`
- User initiated transaction sequence must continue until one of following four events occurs:

1. `COMMIT` statement is reached—all changes are permanently recorded within the database.
2. `ROLLBACK` statement is reached—all the changes are aborted and the database is rolled back to its previous consistent state.
3. End of a program reached—all changes are permanently recorded within the database.
4. Program reaches abnormal termination—the changes made in the database are aborted and the database is rolled back to its previous consistent state.

For example:

```
UPDATE PRODUCT
SET PROD_QOH = PROD_QOH - 100
WHERE PROD_CODE = '345TYX';
UPDATE ACCT_RECEIVABLE
SET ACCT_BALANCE = ACCT_BALANCE + 3500
WHERE ACCT_NUM = '60120010';
COMMIT;
```

In fact, the COMMINT statement used in this example is not necessary if the UPDATE statement is the application's last action and the application terminates normally.

Transaction Log

- The DBMS use transaction log to track all transactions that update database.
  - May be used by ROLLBACK command for triggering recovery requirement.
  - May be used to recover from system failure like network discrepancy or disk crash.
  - While DBMS executes transactions that modify the database, it also updates the transaction log. The log stores:
    - Record for beginning of transaction
    - Each SQL statement
      - The type of operation being performed (update, delete, insert).
      - The names of objects affected by the transaction (the name of the table).
      - The "before" and "after" values for updated fields
      - Pointers to previous and next entries for the same transaction.
    - Commit Statement – the ending of the transaction.

Note

Committed transactions are not rolled back.

1. If a system failure occurs, the DBMS will examine the transaction log for all uncommitted or incomplete transactions, and it will restore (ROLLBACK) the database to its previous state on the basis of this information.
2. If a ROLLBACK is issued before the termination of a transaction, the DBMS will restore the database only for that particular transaction, rather than for all transactions, in order to maintain the durability of the previous transactions.

Concurrency Control

- Coordinates simultaneous transaction execution in multiprocessing database.
- Ensure serialisability of transactions in multiuser database environment.
- Potential problems in multiuser environments.
- Three main problems: **lost updates**, **uncommitted data**, and **inconsistent retrievals**.

Lost updates

- Assume that two concurrent transactions (T1, T2) occur in a PRODUCT table which records a product's quantity on hand (PROD_QOH). The transactions are:

Transaction	Computation
T1: Purchase 100 units	PROD_QOH = PROD_QOH + 100
T2: Sell 30 units	PROD_QOH = PROD_QOH - 30

Normal Execution of Two Transactions

Note

This table shows the serial execution of these transactions under normal circumstances, yielding the correct answer, PROD_QOH=105.

Lost Updates

Note

The addition of 100 units is "lost" during the process.

1. Suppose that a transaction is able to read a product's PROD_QOH value from the table before a previous transaction has been committed.
2. The first transaction (T1) has not yet been committed when the second transaction (T2) is executed.
3. T2 sill operates on the value 35, and its subtraction yields 5 in memory.
4. T1 writes the value 135 to disk, which is promptly overwritten by T2.

Uncommitted Data

- When two transactions, T1 and T2, are executed concurrently and the first transaction (T1) is rolled back after the second transaction (T2) has already accessed the uncommitted data – thus violating the isolation property of transactions. The transactions are:

Transaction	Computation
T1: Purchase 100 units **(Rollback)**	PROD_QOH = PROD_QOH + 100
T2: Sell 30 units	PROD_QOH = PROD_QOH - 30

Correct Execution of Two Transactions

Note

The serial execution of these transactions yields the correct answer.

An Uncommitted Data Problem

Note

The uncommitted data problem can arise when the ROLLBACK is completed after T2 has begun its execution.

Inconsistent Retrievals

- When a transaction calculates some summary (aggregate) functions over a set of data while other transactions are updating the data.
- The transaction might read some data before they are changed and other data after they are changed, thereby yielding inconsistent results.
  1. T1 calculates the total quantity on hand of the products stored in the PRODUCT table.
  2. T2 updates PROD_QOH for two of the PRODUCT table's products.

Retrieval During Update

Note

T1 calculates PROD_QOH but T2 represents the correction of a typing error, the user added 30 units to product 345TYX's PROD_QOH, but meant to add the 30 units to product '123TYZ's PROD_QOH. To correct the problem, the user executes 30 from product 345TYX's PROD_QOH and adds 30 to product 125TYZ's PROD_QOH.

Transaction Results: Data Entry Correction

Note

The initial and final PROD_QOH values while T2 makes the correction—same results but different transaction process.

Transaction Result: Data Entry Correction

Note

- The transaction table demonstrates that inconsistent retrievals are possible during the transaction execution, making the result of T1's execution incorrect.
- Unless the DBMS exercises concurrency control, a multi-user database environment can create chaos within the information system.

5.5.5 The Scheduler – Schedule, Serialisability, Recovery, Isolation

- Previous examples executed the operations within a transaction in an arbitrary order:
  - As long as two transactions, T1 and T2, access unrelated data, there is no conflict, and the order of execution is irrelevant to the final outcome.
  - If the transactions operate on related (or the same) data, conflict is possible among the transaction components, and the selection of one operational order over another may have some undesirable consequences.
- Establishes order of concurrent transaction execution.
- Interleaves execution of database operations to ensure serialisability.
- Bases actions on concurrency control algorithms
  - Locking
  - Time stamping
  - Ensures efficient use of computer's CPU
  - First-come-first-served basis (FCFS) – executed for all transactions if no way to schedule the execution of transactions.
  - Within multi-user DBMS environment, FCFS scheduling tends to yield unacceptable response times.
  - READ and/or WRITE actions that can produce conflicts.

Read/Write Conflict Scenarios: Conflicting Database Operations Matrix

Note

The table below show the possible conflict scenarios if two transactions, T1 and T2, are executed concurrently over the same data.

- **Schedules** – sequences that indicate the chronological order in which instructions of concurrent transactions are executed
  - a schedule for a set of transactions must consist of all instructions of those transactions
  - must preserve the order in which the instructions appear in each individual transaction.
  - Example of schedules (refer right figures 5.7–5.9)
  - Schedule 1 (see Figure 5.7): Let T_1 transfer \$50 from A to B, and T_2 transfer 10% of the balance from A to B. The following is a serial schedule, in which T_1 is followed by T_2.
    - Schedule 2 (see Figure 5.8): Let T_1 and T_2 be the transactions defined previously. The following schedule is not a serial schedule, but it is **equivalent** to Schedule 1.

T_1	T_2
read(A) A := A – 50 write (A) read(B) B := B + 50 write(B)	read(A) temp := A * 0.1 A := A – *temp* write(A) read(B) B := B + *temp* write(B)

Figure 5.7 A sample schedule

T_1	T_2
read(A) $A := A - 50$ write(A)	
	read(A) $temp := A * 0.1$ $A := A - temp$ write(A)
read(B) $B := B + 50$ write(B)	
	read(B) $B := B + temp$ write(B)

Figure 5.8 Concurrent schedule of schedule in Figure 5.7

- Schedule 3 (see Figure 5.9): The following concurrent schedule does not preserve the value of the sum A + B.

T_1	T_2
read(A) $A := A - 50$	
	read(A) $temp := A * 0.1$ $A := A - temp$ write(A) read(B)
write(A) read(B) $B := B + 50$ write(B)	
	$B := B + temp$ write(B)

Figure 5.9 Another Concurrent schudule of schedule in Figure 5.7

5.5.5.1 Serialisability

A (possibly concurrent) schedule is serialisable if it is equivalent to a serial schedule. Different forms of schedule equivalence give rise to the notions of:

1. conflict serialisability
2. view serialisability

■ Conflict Serialisability: Instructions l_i and l_j of transactions T_i and T_j respectively, **conflict** if and only if there exists some item Q accessed by both l_i and l_j, and at least one of these instructions wrote Q.

1. I_i = **read**(Q), I_j = **read**(Q). I_i and I_j don't conflict.
2. I_i = **read**(Q), I_j = **write**(Q). They conflict.
3. I_i = **write**(Q), I_j = **read**(Q). They conflict
4. I_i = **write**(Q), I_j = **write**(Q). They conflict

- If a schedule S can be transformed into a schedule S′ by a series of swaps of non-conflicting instructions, we say that S and S' are **conflict equivalent**.
- We say that a schedule S is **conflict serialisable** if it is conflict equivalent to a serial schedule.

■ View Serialisability: Let S and S′ be two schedules with the same set of transactions. S and S ′ are view equivalent if the following three conditions are met:

1. For each data item Q, if transaction T_i reads the initial value of Q in schedule S, then transaction T_i must, in schedule S', also read the initial value of Q.
2. For each data item Q if transaction T_i executes **read** (Q) in schedule S, and that value was produced by transaction T_j (if any), then transaction T_i must in schedule S' also read the value of Q that was produced by transaction T_j.

3. For each data item Q, the transaction (if any) that performs the final **write**(Q) operation in schedule S must perform the final **write**(Q) operation in schedule S′.

T_3	T_4	T_6
read(Q)		
	write(Q)	
write(Q)		
		write(Q)

- As can be seen, view equivalence is also based purely on reads and writes alone.
- A schedule S is view serialisable if it is view equivalent to a serial schedule.
- Every conflict serialisable schedule is also view serialisable.

T_1	T_5
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$B := A - 10$
	write(B)
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$A := A + 10$
	write(A)

Figure 5.10 View Serializable schedule of schedule in Figure 5.7

- Schedule in Figure 5.10 – a schedule which is view-serialisable but not conflict serialisable.
- Every view serialisable schedule that is not conflict serialisable has blind writes.

■ Other Notions of Serialisability

- Schedule in Figure 5.10 produces same outcome as the serial schedule < T_1, T_5 >, yet is not conflict equivalent or view equivalent to it.
- Determining such equivalence requires analysis of operations other than read and write.

5.5.5.2 Recoverability

Need to address the effect of transaction failures on concurrently running transactions

■ *Recoverable schedule* – if a transaction T_j reads a data items previously written by a transaction T_i, the commit operation of Ti appears before the commit operation of T_j.

- The schedule in Figure 5.11 is not recoverable if T_9 commits immediately after the read.

T_8	T_9
read(A)	
write(A)	
	read(A)
read(B)	

T_{10}	T_{11}	T_{12}
read(A)		
read(B)		
write(A)		
	read(A)	
	write(A)	
		read(A)

Figure 5.11 An example irrecoverable schedule

- If T_8 should abort, T_9 would have read (and possibly shown to the user) an inconsistent database state. Hence database must ensure that schedules are recoverable.

■ *Cascading rollback* – a single transaction failure leads to a series of transaction rollbacks. Consider the following schedule where none of the transactions has yet committed (so the schedule is recoverable)
- If T_{10} fails, T_{11} and T_{12} must also be rolled back.
- Can lead to the undoing of a significant amount of work.

■ *Cascadeless schedules*– cascading rollbacks cannot occur; for each pair of transactions T_i and T_j such that T_j reads a data item previously written by T_i, the commit operation of T_i appears before the read operation of T_j.
- Every cascadeless schedule is also recoverable
- It is desirable to restrict the schedules to those that are cascadeless

➢ Implementation of Isolation

■ Schedules must be conflict or view serialisable, and recoverable, for the sake of database consistency, and preferably cascadeless.

■ A policy in which only one transaction can execute at a time generates serial schedules, but provides a poor degree of concurrency.

■ Concurrency-control schemes tradeoff between the amount of concurrency they allow and the amount of overhead that they incur.

■ Some schemes allow only conflict-serialisable schedules to be generated, while others allow view-serialisable schedules that are not conflict-serialisable.

5.5.6 Concurrency Control with Locking Methods

- Lock guarantees current transaction exclusive use of data item, i.e., transaction T_2 does not have access to a data item that is currently being used by transaction T_1.
- Acquires lock prior to access.
- Lock released when transaction is completed.
- DBMS automatically initiates and enforces locking procedures.
- All lock information is managed by lock manager.

Lock Granularity

- Lock granularity indicates level of lock use: database, table, page, row, or field (attribute).

Database-Level

- The entire database is locked.
- Transaction T2 is prevented to use any tables in the database while T1 is being executed.
- Good for batch processes, but unsuitable for online multi-user DBMSs.
- Figure 5.7 transactions T1 and T2 cannot access the same database concurrently, even if they use different tables. (The access is very slow!)

Table-Level

- The entire table is locked. If a transaction requires access to several tables, each table may be locked.
- Transaction T2 is prevented to use any row in the table while T1 is being executed.
- Two transactions can access the same database as long as they access different tables.
- It causes traffic jams when many transactions are waiting to access the same table.
- Table-level locks are not suitable for multi-user DBMSs.
- Figure 5.8 transaction T1 and T2 cannot access the same table even if they try to use different rows; T2 must wait until T1 unlocks the table.

Page-Level

- The DBMS will lock an entire diskpage (or page), which is the equivalent of a diskblock as a (referenced) section of a disk.

- A page has a fixed size and a table can span several pages while a page can contain several rows of one or more tables.
- Page-level lock is currently the most frequently used multi-user DBMS locking method.

  Shows that T1 and T2 access the same table while locking different diskpages.
- T2 must wait for using a locked page which locates a row, if T1 is using it.

Row-Level

- With less restriction respect to previous discussion, it allows concurrent transactions to access different rows of the same table even if the rows are located on the same page.
- It improves the availability of data, but requires high overhead cost for management.

  For row-level lock.

Field-Level

- It allows concurrent transactions to access the same row, as long as they require the use of different fields (attributes) within a row.
- The most flexible multi-user data access, but cost extremely high level of computer overhead.

Lock Types

- The DBMS may use different lock types: binary or shared/exclusive locks.
- A locking protocol is a set of rules followed by all transactions while requesting and releasing locks. Locking protocols restrict the set of possible schedules.

Binary Locks

- Two states: locked (1) or unlocked (0).
- Locked objects are unavailable to other objects.
- Unlocked objects are open to any transaction.
- Transaction unlocks object when complete.
- Every transaction requires a lock and unlock operation for each data item that is accessed.

Example of Binary Lock Table

Note

The lock and unlock features eliminate the lost update problem encountered. However, binary locks are now considered too restrictive to yield optimal concurrency conditions.

Shared/Exclusive Locks

- Shared (S Mode)
  - Exists when concurrent transactions granted READ access
  - Produces no conflict for read-only transactions
  - Issued when transaction wants to read and exclusive lock not held on item
- Exclusive (X Mode)
  - Exists when access reserved for locking transaction
  - Used when potential for conflict exists.

	S	X
S	true	false
X	false	false

  - Issued when transaction wants to update unlocked data
- Lock-compatibility matrix
  - A transaction may be granted a lock on an item if the requested lock is compatible with locks already held on the item by other transactions
  - Any number of transactions can hold shared locks on an item, but if any transaction holds an exclusive on the item no other transaction may hold any lock on the item.

- If a lock cannot be granted, the requesting transaction is made to wait till all incompatible locks held by other transactions have been released. The lock is then granted.

➢ Reasons to increasing manager's overhead

- The type of lock held must be known before a lock can be granted
- Three lock operations exist: READ_LOCK (to check the type of lock), WRITE_LOCK (to issue the lock), and UNLOCK (to release the lock).
- The schema has been enhanced to allow a lock upgrade (from shared to exclusive) and a lock downgrade (from exclusive to shared).

➢ Problems with Locking

- Transaction schedule may not be serialisable
  - Managed through two-phase locking
- Schedule may create deadlocks
  - Managed by using deadlock detection and prevention techniques

5.5.6.1 Two-Phase Locking

Two-phase locking defines how transactions acquire and relinquish (or revoke) locks.

1. Growing phase – acquires all the required locks without unlocking any data. Once all locks have been acquired, the transaction is in its locked point.
2. Shrinking phase – releases all locks and cannot obtain any new lock.

➢ Governing rules

- Two transactions cannot have conflicting locks
- No unlock operation can precede a lock operation in the same transaction
- No data are affected until all locks are obtained

➢ In the example for two-phase locking protocol the transaction acquires all the locks it needs (two locks are required) until it reaches its locked point.

- When the locked point is reached, the data are modified to conform to the transaction requirements.
- The transaction is completed as it released all of the locks it acquired in the first phase.

➢ Updates for two-phase locking protocols:

- Two-phase locking does not ensure freedom from deadlocks.
- Cascading roll-back is possible under two-phase locking. To avoid this, follow a modified protocol called strict two-phase locking. Here a transaction must hold all its exclusive locks till it commits/aborts.
- Rigorous two-phase locking is even stricter: here all locks are held till commit/abort. In this protocol transactions can be serialized in the order in which they commit.
- There can be conflict serialisable schedules that cannot be obtained if two-phase locking is used.
- However, in the absence of extra information (e.g., ordering of access to data), two-phase locking is needed for conflict Serialisability in the following sense:

  Given a transaction T_i that does not follow two-phase locking, we can find a transaction T_j that uses two-phase locking, and a schedule for T_i and T_j that is not conflict serialisable.

5.5.6.2 Deadlocks

Occurs when two transactions wait for each other to unlock data. For example:

T1 = access data items X and Y

T2 = access data items Y and X

➢ Deadly embrace – if T_1 has not unlocked data item Y, T_2 cannot begin; if T_2 has not unlocked data item X, T_1 cannot continue.

➢ Starvation is also possible if concurrency control manager is badly designed.

- For example, a transaction may be waiting for an X-lock (exclusive mode) on an item, while a sequence of other transactions request and are granted an S-lock (shared mode) on the same item.
- The same transaction is repeatedly rolled back due to deadlocks.

➢ Control techniques

- Deadlock prevention – a transaction requesting a new lock is aborted if there is the possibility that a deadlock can occur.
- If the transaction is aborted, all the changes made by this transaction are rolled back, and all locks obtained by the transaction are released.
- It works because it avoids the conditions that lead to deadlocking.
- Deadlock detection – the DBMS periodically tests the database for deadlocks.
- If a deadlock is found, one of the transactions (the "victim") is aborted (rolled back and restarted), and the other transaction continues.
- Deadlock avoidance – the transaction must obtain all the locks it needs before it can be executed.
- The technique avoids rollback of conflicting transactions by requiring that locks be obtained in succession.
- The serial lock assignment required in deadlock avoidance increase action response times.

➢ Control Choices

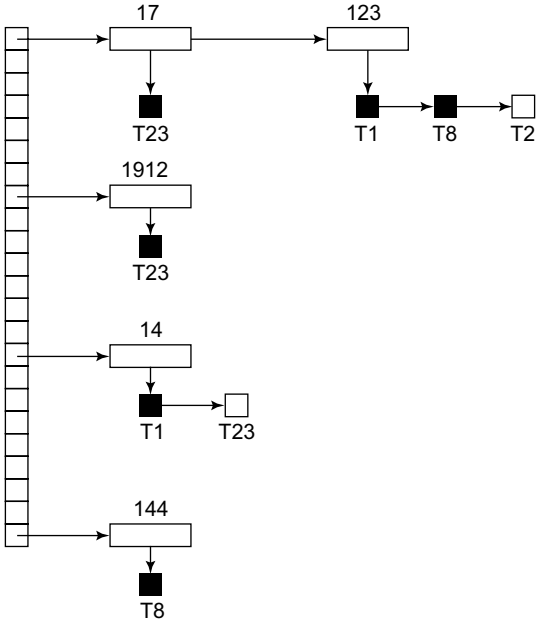
- If the probability of deadlocks is low, deadlock detection is recommended.
- If the probability of deadlocks is high, deadlock prevention is recommended.
- If response time is not high on the system priority list, deadlock avoidance might be employed.

Implementation of Locking

- A Lock manager can be implemented as a separate process to which transactions send lock and unlock requests.
- The lock manager replies to a lock request by sending a lock grant messages (or a message asking the transaction to roll back, in case of a deadlock).
- The requesting transaction waits until its request is answered.
- The lock manager maintains a datastructure called a lock table to record granted locks and pending requests.
- The lock table is usually implemented as an in-memory hash table indexed on the name of the data item being locked.

➢ Lock Table

- Black rectangles indicate granted locks, white ones indicate waiting requests.



- Lock table also records the type of lock granted or requested.
- New request is added to the end of the queue of requests for the data item, and granted if it is compatible with all earlier locks.
- Unlock requests result in the request being deleted, and later requests are checked to see if they can now be granted.
- If transaction aborts, all waiting or granted requests of the transaction are deleted.
  - Lock manager may keep a list of locks held by each transaction, to implement this efficiently.

5.5.7 Concurrency Control with Time Stamping Methods

- Assigns global unique time stamp to each transaction
- Produces order for transaction submission
- Properties
- Uniqueness: ensures that no equal time stamp values can exist.
- Monotonicity: ensures that time stamp values always increase.
- DBMS executes conflicting operations in time stamp order to ensure serialisability of the transaction.
- If two transactions conflict, one often is stopped, rolled back, and assigned a new time stamp value.
- Each value requires two additional time stamps fields
- Last time field read
- Last update
- Time stamping tends to demand a lot of system resources because there is a possibility that many transactions may have to be stopped, rescheduled, and re-stamped.

5.5.7.1 Timestamp-Based Protocols

- Each transaction is issued a timestamp when it enters the system. If an old transaction T_i has time-stamp TS(T_i), a new transaction T_j is assigned time-stamp TS(T_j) such that TS(T_i) < TS(T_j).
- The protocol manages concurrent execution such that the time-stamps determine the serialisability order.
- In order to assure such behavior, the protocol maintains for each data Q two timestamp values:
- **W-timestamp**(Q) is the largest time-stamp of any transaction that executed write(Q) successfully.
- **R-timestamp**(Q) is the largest time-stamp of any transaction that executed read(Q) successfully.
- The timestamp ordering protocol ensures that any conflicting read and write operations are executed in timestamp order.
- Suppose a transaction Ti issues a **read**(Q)
  1. If TS(T_i) ≦ **W-timestamp**(Q), then T_i needs to read a value of Q that was already overwritten. Hence, the read operation is rejected, and T_i is rolled back.
  2. If TS(T_i) ≧ **W-timestamp**(Q), then the read operation is executed, and **R-timestamp**(Q) is set to the maximum of **R-timestamp**(Q) and TS(T_i).
- Suppose that transaction T_i issues **write**(Q).
  1. If TS(T_i) < **R-timestamp**(Q), then the value of Q that T_i is producing was needed previously, and the system assumed that that value would never be produced. Hence, the write operation is rejected, and T_i is rolled back.
  2. If TS(T_i) < **W-timestamp**(Q), then T_i is attempting to write an obsolete value of Q. Hence, this write operation is rejected, and T_i is rolled back.
  3. Otherwise, the **write** operation is executed, and W-timestamp(Q) is set to TS(T_i).

Concurrency Control with Optimistic Methods

- A **validation-based protocol** that assumes most database operations do not conflict.
- No requirement on locking or time stamping techniques.

- Transaction executed without restrictions until committed and fully in the hope that all will go well during validation.
- Two or three Phases:
  - **Read (and Execution) Phase** – the transaction reads the database, executes the needed computations, and makes the updates to a private copy of the database values.
  - **Validation Phase** – the transaction is validated to ensure that the changes made will not affect the integrity and consistency of the database.
  - **Write Phase** – the changes are permanently applied to the database.

➢ The optimistic approach is acceptable for mostly read or query database system that require very few update transactions.

➢ Each transaction T_i has 3 timestamps
  - **Start**(T_i) : the time when T_i started its execution
  - **Validation**(T_i): the time when T_i entered its validation phase
  - **Finish**(T_i): the time when T_i finished its write phase

➢ Serialisability order is determined by timestamp given at validation time, to increase concurrency. Thus TS(T_i) is given the value of **Validation**(T_i).

➢ This protocol is useful and gives greater degree of concurrency if probability of conflicts is low. That is because the serialisability order is not pre-decided and relatively less transactions will have to be rolled back.

5.5.8 Database Recovery Management

➢ Restores a database to previously consistent state, usually inconsistent, to a previous consistent state.

➢ Based on the **atomic transaction property**: all portions of the transaction must be treated as a single logical unit of work, in which all operations must be applied and completed to produce a consistent database.

➢ Level of backup
  - Full backup – dump of the database.
  - Differential backup – only the last modifications done to the database are copied.
  - Transaction log – only the transaction log operations that are not reflected in a previous backup copy of the database.

➢ The database backup is stored in a secure place, usually in a different building, and protected against dangers such as file, theft, flood, and other potential calamities.

➢ Causes of Database Failure
  - Software – be traceable to the operating system, the DBMS software, application programs, or virus.
  - Hardware – include memory chip errors, disk crashes, bad disk sectors, disk full errors.
  - Programming Exemption – application programs or end users may roll back transactions when certain conditions are defined.
  - Transaction – the system detects deadlocks and aborts one of the transactions.
  - External – a system suffers complete destruction due to fire, earthquake, flood, etc.

5.5.8.1 Transaction Recovery

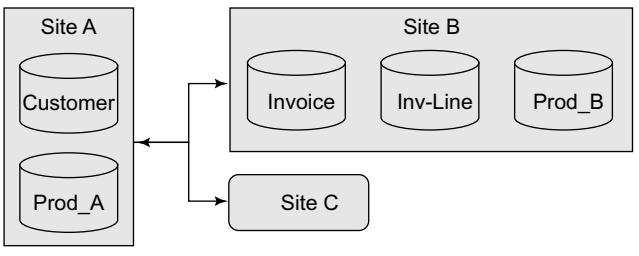
➢ Four important concepts to affect recovery process –
  - Write-ahead-log protocol – ensures that transaction logs are always written before any database data are actually updated.
  - Redundant transaction logs – ensure that a disk physical failure will not impair the DBMS ability to recover data.
  - Database buffers – create temporary storage area in primary memory used to speed up disk operations and improve processing time.
  - Database checkpoint – setup an operation in which the DBMS writes all of its updated buffers to disk and registered in the transaction log.

- Transaction recovery procedure generally make use of deferred-write and write-through techniques.
  - Deferred-write (or Deferred-update)
    - Changes are written to the transaction log, not physical database.
    - Database updated after transaction reaches commit point.
    - Steps:
      1. Identify the last checkpoint in the transaction log. This is the last time transaction data was physically saved to disk.
      2. For a transaction that started and committed before the last checkpoint, nothing needs to be done, because the data are already saved.
      3. For a transaction that performed a commit operation after the last checkpoint, the DBMS uses the transaction log records to redo the transaction and to update the database, using "after" values in the transaction log. The changes are made in ascending order, from the oldest to the newest.
      4. For any transaction with a RP::BACK operation after the last checkpoint or that was left active (with neither a `COMMIT` nor a `ROLLBACK`) before the failure occurred, nothing needs to be done because the database was never updated.
  - Write-through (or immediate update)
    - Immediately updated by during execution
    - Before the transaction reaches its commit point
    - Transaction log also updated
    - Transaction fails, database uses log information to `ROLLBACK`
    - Steps:
      1. Identify the last checkpoint in the transaction log. This is the last time transaction data was physically saved to disk.
      2. For a transaction that started and committed before the last checkpoint, nothing needs to be done, because the data are already saved.
      3. For a transaction that committed after the last checkpoint, the DBMS redoes the transaction, using "after" values in the transaction log. Changes are applied in ascending order, from the oldest to the newest.
      4. For any transaction with a `ROLLBACK` operation after the last checkpoint or that was left active (with neither a `COMMIT` nor a `ROLLBACK`) before the failure occurred, the DBMS uses the transaction log records to `ROLLBACK` or undo the operations, using the "before" values in the transaction log. Changes are applied in reverse order, from the newest to the oldest.

5.6 Solved Questions

1. The following question is based on the DDBMS scenario in the following Figure.

Table	Fragments	Location
Customer	N/A	A
Product	Prod_A	A
	Prod_B	B
INVOICE	N/A	B
INV_LINE	N/A	B



Specify the types of operations the database must support (remote request, remote transaction, distributed transaction, or distributed request) in order to perform the following operations:

To answer the following questions, remember that the key to each answer is in the number of remote data processors that are accessed by each request/transaction. Remember that a distributed request is necessary if a single SQL statement is to access more than one remote DP site.

Use the following summary:

	Number of remote DPs	
Operation	1	> 1
Request	Remote	Distributed
Transaction	Remote	Distributed

Based on this summary, the following questions have to be answered.

At Site C:

A. **SELECT ***
FROM CUSTOMER;

Answer: This SQL sequence represents a *remote request.*

B. **SELECT ***
FROM INVOICE
WHERE INV_TOTAL > 1000;

Answer: This SQL sequence represents a *remote request.*

C. **SELECT ***
FROM PRODUCT
WHERE PROD_QOH < 10;

Answer: This SQL sequence represents a distributed request. Note that the distributed request is required when a single request must access two DP sites. The PRODUCT table is fragmented across two sites, A and B. In order for this SQL sequence to run properly, it must access the data at both sites.

D. **BEGIN WORK;**
UPDATE CUSTOMER
SET CUS_BALANCE = CUS_BALANCE + 100
WHERE CUS_NUM='10936';
INSERT INTO INVOICE(INV_NUM, CUS_NUM, INV_DATE, INV_TOTAL)
VALUES ('986391', '10936', '15-FEB-2002', 100);
INSERT INTO INVLINE(INV_NUM, PROD_CODE, LINE_PRICE)
VALUES ('986391', '1023', 100);
UPDATE PRODUCT
SET PROD_QOH = PROD_QOH – 1
WHERE PROD_CODE = '1023';
COMMIT WORK;

Answer: This SQL sequence represents a *distributed request.*

Note that UPDATE CUSTOMER and the two INSERT statements only require remote request capabilities. However, the entire transaction must access more than one remote DP site, so we also need distributed transaction capability. The last UPDATE PRODUCT statement accesses two remote sites because the PRODUCT table is divided into two fragments located at two remote DP sites. Therefore, the transaction as a whole requires distributed request capability.

E. **BEGIN WORK;**
INSERT CUSTOMER(CUS_NUM, CUS_NAME, CUS_ADDRESS CUS_BALANCE)
VALUES ('34210",'Victor Ephanor', '123 Main St', 0.00);
INSERT INTO INVOICE(INV_NUM, CUS_NUM, INV_DATE, INV_TOTAL)
VALUES ('986434', '34210', '10-AUG-1999', 2.00);
COMMIT WORK;

Answer: This SQL sequence represents a *distributed transaction*. Note that, in this transaction, each individual request requires only remote request capabilities. However, the transaction as a whole accesses two remote sites. Therefore, distributed request capability is required.

A**t Site A:**

F. **SELECT CUS_NUM, CUS_NAME, INV_TOTAL**
FROM CUSTOMER, INVOICE
WHERE CUSTOMER.CUS_NUM = INVOICE.CUS_NUM;

Answer: This SQL sequence represents a *remote request*. Note that the request accesses only one remote DP site; therefore only remote request capability is needed.

G. **SELECT ***
FROM INVOICE
WHERE INV_TOTAL > 1000;

Answer: This SQL sequence represents a *remote request*, because it accesses only one remote DP site.

H. **SELECT ***
FROM PRODUCT
WHERE PROD_QOH < 10;

Answer: This SQL sequence represents a *distributed request.* In this case, the PRODUCT table is partitioned between two DP sites, A and B. Although the request accesses only one remote DP site, it accesses a table that is partitioned into two fragments: PROD A and PROD B. Only if the DBMS supports distributed requests a single request can access a partitioned table.

A**t Site B:**

I. **SELECT ***
FROM CUSTOMER;

Answer: This SQL sequence represents a *remote request.*

J. **SELECT CUS_NAME, INV_TOTAL**
FROM CUSTOMER, INVOICE
WHERE INV_TOTAL > 1000;

Answer: This SQL sequence represents a *remote request.*

K. **SELECT ***
FROM PRODUCT
WHERE PROD_QOH < 10;

Answer: This SQL sequence represents a *distributed request*. (See explanation for Part h.)

2. The following data structure and constraints exist for a magazine publishing company.

- The company publishes one regional magazine each in Florida (FL), South Carolina (SC), Georgia (GA), and Tennessee (TN).
- The company has 300,000 customers (subscribers) distributed throughout the four states listed in Part a.
- On the first of each month an annual subscription INVOICE is printed and sent to all customers whose subscription is due (CUS_SUB_DATE) for renewal. The INVOICE entity contains a REGION attribute to indicate the state (FL, SC, GA, TN) in which the customer resides.

CUSTOMER (CUS_NUM, CUS_NAME, CUS_ADDRESS, CUS_CITY, CUS_STATE, CUS_SUB_DATE)

INVOICE (INV_NUM, REG_CODE, CUS_NUM, INV_DATE, INV_TOTAL)

The company's management is aware of the problems associated with centralised management and has decided that it is time to decentralise the management of the subscriptions in its four regional subsidiaries. Each subscription site will handle its own customer and invoice data. The company's management, however, wants to have access to customer and invoice data to generate annual reports and to issue ad hoc queries, such as:

- List all current customers by region.
- List all new customers by region.
- Report all invoices by customer and by region.

A. Given these requirements, how must you partition the database?

Answer: The CUSTOMER table must be partitioned horizontally by state.

B. What recommendations will you make regarding the type and characteristics of the required database system?

Answer: The Magazine Publishing Company requires a distributed system with distributed database capabilities. The distributed system will be distributed among the company locations in South Carolina, Georgia, Florida, and Tennessee.

The DDBMS must be able to support distributed transparency features, such as fragmentation transparency, replica transparency, transaction transparency, and performance transparency. Heterogeneous capability is not a mandatory feature since we assume there is no existing DBMS in place and that the company wants to standardise on a single DBMS.

C. What type of data fragmentation is needed for each table?

Answer: The database must be horizontally partitioned, using the STATE attribute for the CUSTOMER table and the REGION attribute for the INVOICE table.

D. What must be the criteria used to partition each database?

Answer: The following fragmentation segments reflect the criteria used to partition each database:

Horizontal Fragmentation of the CUSTOMER Table By State

Fragment Name	Location	Condition	Node name
C1	Tennessee	CUS_STATE = 'TN'	NAS
C2	Georgia	CUS_STATE = 'GA'	ATL
C3	Florida	CUS_STATE = 'FL'	TAM
C4	South Carolina	CUS_STATE = 'SC'	CHA

Horizontal Fragmentation of the INVOICE Table By Region

Fragment Name	Location	Condition	Node name
I1	Tennessee	CUS_STATE = 'TN'	NAS
I2	Georgia	CUS_STATE = 'GA'	ATL
I3	Florida	CUS_STATE = 'FL'	TAM
I4	South Carolina	CUS_STATE = 'SC'	CHA

E. Design the database fragments. Show an example with node names, location, fragment names, attribute names, and demonstration data.

Answer:

Fragment C1 Location: Tennessee Node: NAS

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_CITY	CUS_STATE	CUS_SUB_DATE
10884	James D. Burger	123 Court Avenue	Memphis	TN	8-DEC-2000
10993	Lisa B. Barnette	910 Eagle Street	Nashville	TN	12-MAR-2000

Fragment C2 Location: Georgia Node: ATL

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_CITY	CUS_STATE	CUS_SUB_DATE
11887	Ginny E. Stratton	335 Main Street	Atlanta	GA	11-AUG-2000
13558	Anna H. Ariona	657 Mason Ave.	Dalton	GA	23-JUN-2000

Fragment C3 Location: Florida Node: TAM

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_CITY	CUS_STATE	CUS_SUB_DATE
10014	John T. Chi	456 Brent Avenue	Miami	FL	18-NOV-2000
15998	Lisa B. Barnette	234 Ramala Street	Tampa	FL	23-MAR-2000

Fragment C4 Location: South Carolina Node: CHA

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_CITY	CUS_STATE	CUS_SUB_DATE
21562	Thomas F. Matto	45 N. Pratt Circle	Charleston	SC	2-DEC-2000
18776	Mary B. Smith	526 Boone Pike	Charleston	SC	28-OCT-2000

Fragment I1 Location: Tennessee Node: NAS

INV_NUM	REGION-CODE	CUS_NUM	INV_DATE	INV_TOTAL
213342	TN	10884	1-NOV-2000	45.95
209987	TN	10993	15-APR-2000	45.95

Fragment I2 Location: Georgia Node: ATL

INV_NUM	REGION-CODE	CUS_NUM	INV_DATE	INV_TOTAL
198893	GA	11887	15-AUG-2000	70.45
224345	GA	13558	1-JUN-2000	45.95

Fragment I3 Location: Florida Node: TAM

INV_NUM	REGION-CODE	CUS_NUM	INV_DATE	INV_TOTAL
200915	FL	10014	1-NOV-2000	45.95
231148	FL	15998	1-MAR-2000	24.95

Fragment I4 Location: South Carolina Node: CHA

INV_NUM	REGION-CODE	CUS_NUM	INV_DATE	INV_TOTAL
243312	SC	21562	15-NOV-2000	45.95
231156	SC	18776	1-OCT-2000	45.95

F. What type of distributed database operations must be supported at each remote site?

Answer: To answer this question, we must first draw a map of the locations, the fragments at each location, and the type of transaction or request support required to access the data in the distributed database.

	Node				
Fragment	NAS	ATL	TAM	CHA	Headquarters
CUSTOMER	C1	C2	C3	C4	
INVOICE	I1	I2	I3	I4	
Distributed Operations Required	none	none	none	none	distributed request

Given the problem's specifications, we conclude that no interstate access of CUSTOMER or INVOICE data is required. Therefore, no distributed database access is required in the four nodes. For the headquarters, the manager wants to be able to access the data in all four nodes through a single SQL request. Therefore, the DDBMS must support distributed requests.

G. What type of distributed database operations must be supported at the headquarters site?

Answer: See the answer for part f.

1. What is meant by a recursive relationship type? Give some examples of recursive relationship types.

Answer: Recursive relationship is the entity type participates more than once in a relationship type in different role. For example a department manger is the supervisor of the employee work for the department or a supervisee of a general manger of a company.

2. Example of a 3NF table that is not BCNF,

Consder a problem involving students, courses and instructors. For each course, each student is taught by only one instructor. A course may be taught by more than one instructor. Each instructor teaches only one course. Assuming, S = student, C = course, I = instructor. The following FD's are seen valid in the relation.

SC → I

I → C

This table is 3NF with a candidate key SC:

student	**course**	**instructor**
Sutton	Math	Von Neumann
Sutton	Journalism	Murrow
Niven	Math	Von Neumann
Niven	Physics	Fermi
Wilson	Physics	Einstein

Delete anomaly: If Sutton drops Journalism, then we have no record of Murrow teaching Journalism.

How can we decompose this table into BCNF?

Decomposition 1 (bad).....eliminates the delete anomaly

SC (no FDs) and I -> C (two tables)

Problems -

1. lossy join
2. dependency SC -> I is not preserved

SC	student	course IC	instructor	course
	Sutton	Math	Von Neumann	Math
	Sutton	Journalism	Murrow	Journalism
	Niven	Math	Fermi	Physics
	Niven	Physics	Einstein	Physics
	Wilson	Physics		

---------- join SC and IC ----------

SCI'	student	course	instructor
	Sutton	Math	Von Neumann
	Sutton	Journalism	Murrow
	Niven	Math	Von Neumann
	Niven	Physics	Fermi
	Niven	Physics	Einstein **(spurious row)**
	Wilson	Physics	Fermi **(spurious row)**
	Wilson	Physics	Einstein

Decomposition 2 (better).....eliminates the delete anomaly

SI (no FD) and I → C

Advantages – eliminates the delete anomaly, lossless

Disadvantage – dependency SC → I is not preserved

SI	student	instructor IC	instructor	course
	Sutton	Von Neumann	Von Neumann	Math
	Sutton	Murrow	Murrow	Journalism
	Niven	Von Neumann	Fermi	Physics
	Niven	Fermi	Einstein	Physics
	Wilson	Einstein	Dantzig	Math **(new)**
	Sutton	Dantzig **(new)**		

The new row is allowed in SI using unique (student, instructor) in the create table command, and the join of SI and IC is lossless. However, a join of SI and IC now produces the following two rows:

student	**course**	**instructor**
Sutton	Math	Von Neumann
Sutton	Math	Dantzig which violates the FD SC → I.

Decomposition 3 (tradeoff between integrity and performance)

SC → I and I → C (two tables with redundant data)

Problems – extra updates and storage cost

3. What is a composite entity, and when is it used?

Answer: A composite entity, also known as a bridge entity, is one that is used to transform M:N relation-

ships into sets of 1:M relationships. The composite entity's primary key consists of the combination of primary keys from the entities it connects. For a detailed review of the role played by composite entities, refer to the second discussion focus question **(How are M:N relationships handled in the development of an E-R diagram?)**

6. What two courses of action are available to a designer when a multivalued attribute is encountered?

Answer: The designer can split the multivalued attributes into its components and keep these components in the same entity.

The designer may also create a new entity composed of the multivalued attribute's components and link this new entity to the entity in which the multivalued attributes occurred originally. This second option is especially desirable when the number of outcomes in the multivalued attribute is, for all practical purposes, unlimited. For example, employees classified as "technical" may have certifications in many different areas and at many different levels.

7. Consider the following schema for a relational database:

Relation	Attributes
professor	ssn, profname, status,salary
course	crscode, crsname, credits
taught	croscode, semester, ssn

Assumption: (1) Each course has only one instructor in each semester; (2) all professors have different salaries; (3) all professors have different names; (4) all courses have different names; (5) status can take values from "Full", "Associate", and "Assistant".

For each of the following queries, give an expression in *relational algebra:*

(a) Return those professors who have taught both 'csc6710' and 'csc 7710'.

$\pi_{ssn}(\sigma_{crscode='csc6710'}(Taught)) \cap \pi_{ssn}(\sigma_{crscode = 'csc7710'}(Taught))$

(b) Return those professors who have taught 'csc7710'.

$\pi_{ssn}(Professor)-\pi_{ssn}(\sigma_{crscode = 'csc7710'}(Taught))$

(c) Return those professors who taught 'CSC6710' and 'CSC 7710' in the same semester

$\pi_{ssn}(\sigma_{crscode1='csc6710'}(Taught[crscode\ 1, ssn\ semester])) \bowtie$

$\sigma_{crscode2 = 'csc7710'}(Taught[crscode\ 2, ssn\ semester])$

(d) Return those professors who taught 'CSC6710' or 'CSC 7710' but not both

$\pi_{ssn}(\sigma_{crscode='csc6710'v\ crscode = 'csc\ 7710'}(Taught)) - \pi_{ssn}(\sigma_{crscode='csc6710'}(Taught) \cap (\pi_{ssn}(\sigma_{crscode='csc7710'}(Taught)))$

(e) Return those courses that have never been taught.

$\pi_{crscode}(Course)-\pi_{crscode}(Taught)$

(f) Return those courses that have never been taught at least in two semesters.

$\pi_{crscode}(\sigma_{semester\ 1 <> semester\ 2}(Taught[crscode, ssn1\ semester1] \bowtie Taught[crscode, ssn2\ semester2]))$

(g) Return the names of professors who ever taught 'CSC6710'.

$\pi_{profname}(\sigma_{crscode = 'csc6710'}(Taught) \bowtie Professor)$

(h) Return the names of full professors who ever taught 'CSC6710'.

$\pi_{profname}(\sigma_{crscode = 'csc6710'}(Taught) \bowtie \sigma_{status = 'full'}(Professor)$

(i) Return the names of full professors who ever taught at least two courses in one semester.

$\pi_{profname}(\pi_{ssn}(\sigma_{crscode1<>crscode2}(Taught[crscode\ 1, ssn\ semester] \bowtie (Taught[crscode\ 2, ssn\ semester])) \bowtie \sigma_{status = 'full'}(Professor))$

(j) List all the course names that professors 'Smith' taught in fall of 2007.

$\pi_{crsname}(\sigma_{profname='Smith'}(Professor) \bowtie (\sigma_{semester='f2007'}(Taught) \bowtie (Course)$

(k) List the names of those courses that professor Smith have never taught.

$\pi_{crsname}(Course)-\pi_{crsname}(\sigma_{profname='Smith'}(Professor) \bowtie (Taught) \bowtie (Course)$

(l) Return those courses that have been taught by all professors.

$\pi_{crscode,\ ssn}(Taught)/\pi_{ssn}(Professor)$

(m) Return those courses that have been taught in all semesters.

$\pi_{crscode,\ semester}(Taught)/\pi_{semester}(Taught)$

(n) Return those courses that have been taught ONLY by assisitant professors.

$\pi_{crscode}(Course)-\pi_{crscode}(\sigma_{status \neq 'Assistant'}(Professor) \bowtie Taught)$

(o) Return those professors who have taught 'csc6710' but never 'csc7710'.

$\pi_{ssn}(\sigma_{crscode='csc\ 6710'}(Taught))-\pi_{ssn}(\sigma_{crscode='csc7710'}(Taught)$

For each of the following queries, give an expression in SQL:

(p) Return those professors who have taught both 'csc6710' and 'csc7710'.

```
SELECT T1.ssn
From Taught T1, Taught T2,
Where T1.crscode = 'CSC6710' AND
T2.crscode='CSC7710' AND T1.ssn=T2.ssn
```

(q) Return those professors who have never taught 'csc7710'.

```
(SELECT ssn
From Professor)
EXCEPT
(SELECT ssn
From Taught T
Where T.crscode = 'CSC7710')
```

(r) Return those professors who taught 'CSC6710' and 'CSC7710" in the same semester

```
SELECT T1.ssn
From Taught T1, Taught T2, Where T1.crscode = 'CSC6710' AND T2.crscode='CSC7710' AND T1.ssn=T2.ssn
AND T1.semester=T2.semester
```

(s) Return those professors who taught 'CSC6710' or 'CSC7710" but not both.

```
(SELECT ssn
FROM Taught T
WHERE T.crscode='CSC6710' OR T.crscode = 'CSC7710')
Except
(SELECT T1.ssn
From Taught T1, Taught T2,
Where T1.crscode = 'CSC6710') AND T2.crscode ='CSC7710' AND T1.ssn=T2.ssn)
```

(t) Return those courses that have been taught at least in 10 semesters.

```
SELECT crscode
FROM Taught
GROUP BY crscode
HAVING COUNT(*) >= 10
```

(u) Return those courses that have been taught by at least 5 different professors.

```
SELECT crscode
FROM (SELECT DISTINCT crscode, ssn FROM TAUGHT)
GROUP BY crscode
HAVING COUNT(*) >= 5
SELECT crscode
FROM Course C
WHERE (SELECT COUNT(DISTINCT *) FROM Taught T
WHERE T.crscode = C.crscode
) >=5.
```

(v) Return the names of full professors who ever taught at least two courses in one semester.

```
SELECT P.profname
FROM Professor P, Taught T1, Taught T2
WHERE P.status = 'Full' AND P.ssn = T1.ssn AND T1.ssn = T2.ssn
AND T1.crscode <> T2.crscode AND T1.semester = T2.semester
```

(w) In chronological order, list the number of courses that the professor with ssn ssn =
123456789 taught in each semester.

```
SELECT semester, COUNT(*)
FROM Taught
WHERE ssn = '123456789'
GROUP BY semester
ORDER BY semester ASC
```

(x) Delete those professors who taught less than 40 credits.

```
DELETE FROM Professor
WHERE ssn IN(
SELECT T.ssn
      FROM Taught T, Course C
WHERE T.crscode = C.crscode
GROUP BY ssn
HAVING SUM(C.credits) < 40
)
```

(y) Return the professors who taught the largest number of courses in Fall 2001.

```
SE
LECT *
FROM Professor P1
WHERE Not EXISTS
(
SELECT *
FROM Professor P2
WHERE(
(SELECT COUNT(*)
FROM Taught
WHERE Taught.ssn = P2.ssn AND Taught. semester='F2001')
>
(SELECT COUNT(*)
FROM Taught
WHERE Taught.ssn = P1.ssn AND Taught. semester='F2001')
)
```

(z) Change all the credits to 4 for those courses that are taught in f2006 semester.

```
UPDATE Course
SET credits = 4
WHERE crscode IN
(
SELECT crscode
FROM Taught
WHERE semester = 'f2006'
)
```

(aa) Return the name(s) of the professor(s) who taught the most number of courses in S2006.

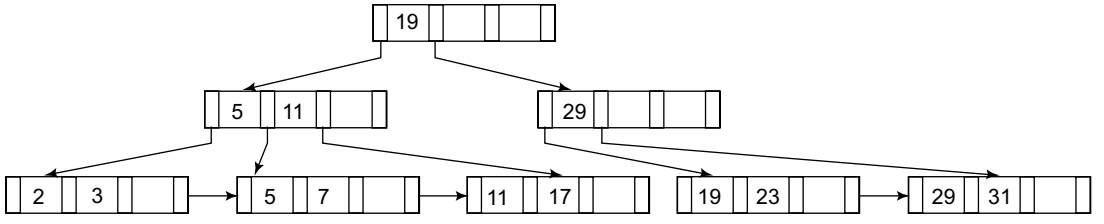
```
SELECT profname
FROM Professor
WHERE ssn IN(
SELECT ssn FROM Taught
WHERE semester = 'S2006'
GROUP BY ssn
HAVING COUNT(*) =
(SELECT MAX(Num)
FROM
(SELECT ssn, COUNT(*) as Num
FROM Taught
WHERE semester = 'S2006'
GROUP BY ssn)
)
    )
```

8. Construct a B+- tree for the following set of key values:
(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

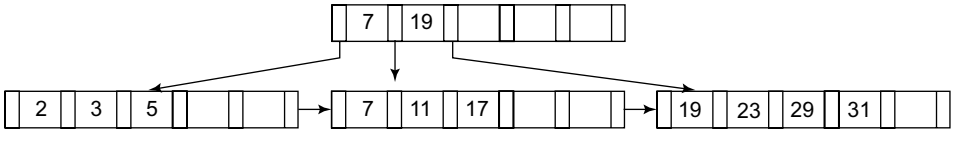
(i) Assume that the tree is initially empty and values are added in ascending order. Construct B+- trees for the cases where the number of pointers that will fit in one node is as follows:

A. Four B. Six C. Eight

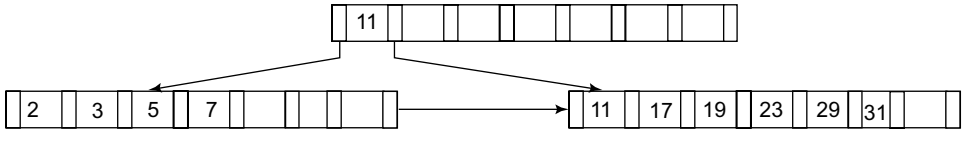
Answer: The following were generated by inserting values into the B+- tree in ascending order. A node (other than the root) was never allowed to have fewer than [n/2] values/pointers.



(a)



(b)



(c)

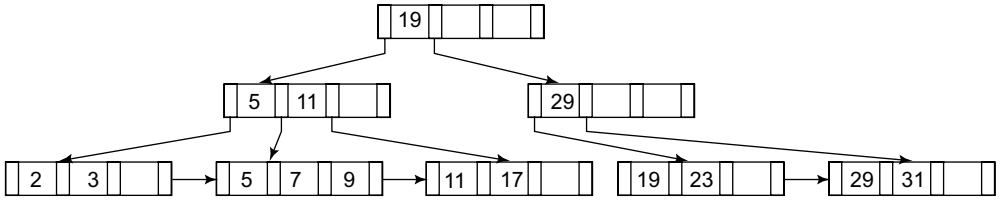
(ii) For each B+- tree in Question i), show the form of the tree after each of the following series of operations:

A. Insert 9 B. Insert 10 C. Insert 8 D. Delete 23

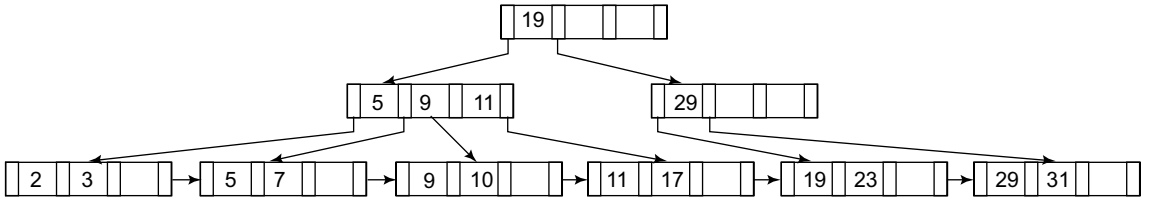
E. Delete 19

Answer:

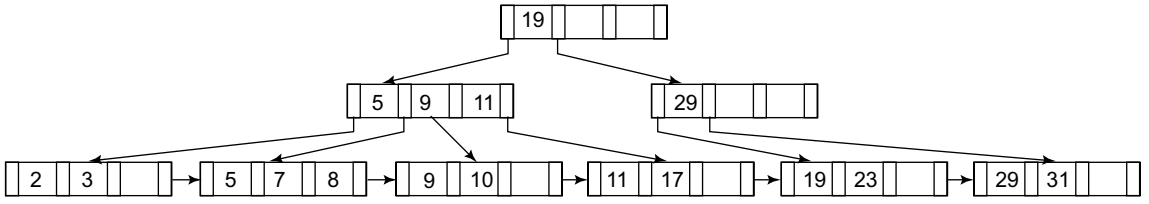
With n = 4



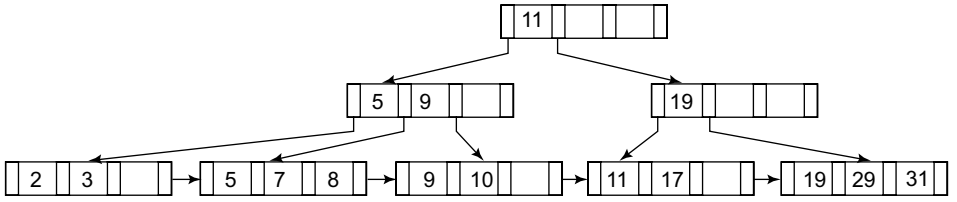
Insert 10:



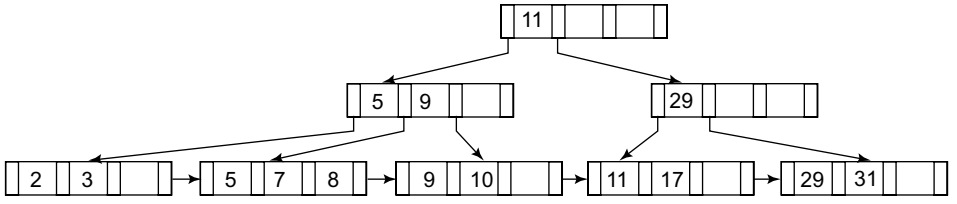
Insert 8:



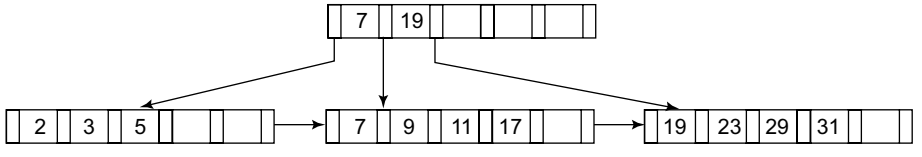
Delete 23:



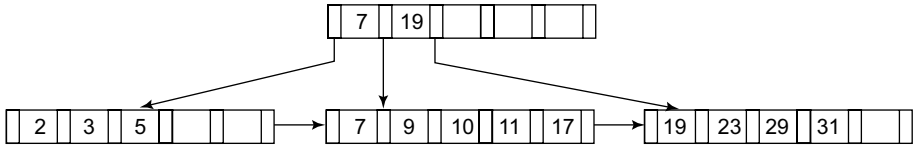
Delete 19:



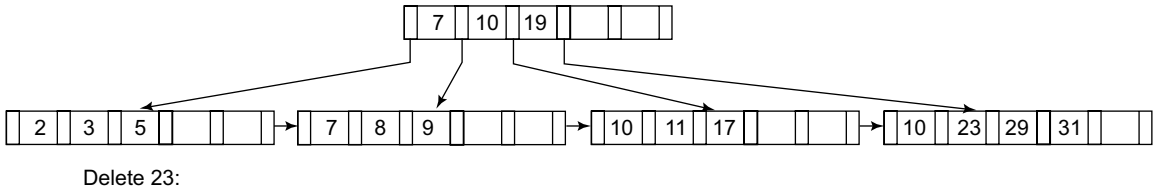
With n = 6



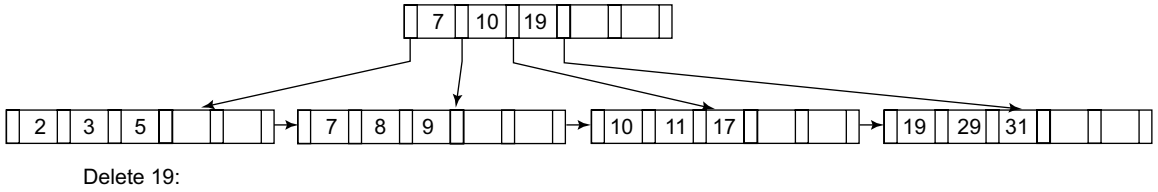
Insert 10:



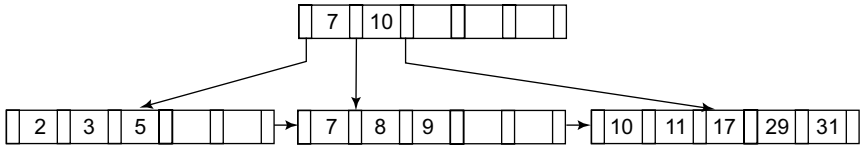
Insert 8:



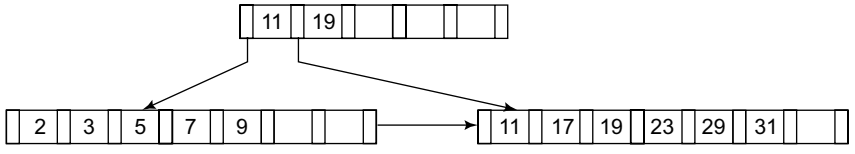
Delete 23:



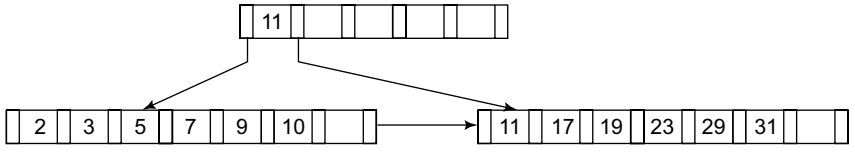
Delete 19:



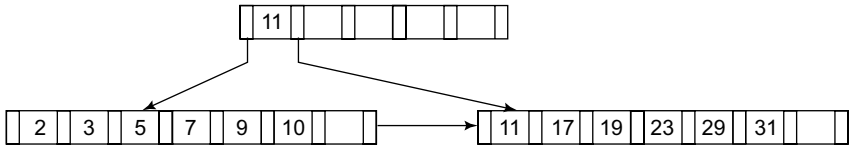
With n=8



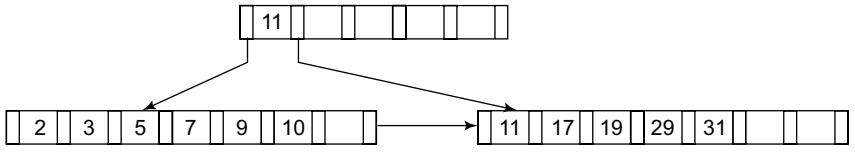
Insert 10:



Insert 8:



Delete 23:



Delete 19:

9. How does Tuple-oriented relational calculus differ from domain-oriented relational calculus?

The tuple-oriented calculus uses a tuple variables i.e., variable whose only permitted values are tuples of that relation. E.g. QUEL

The domain-oriented calculus has domain variables i.e., variables that range over the underlying domains instead of over relation. E.g. ILL, DEDUCE.

10. When is a functional dependency F said to be minimal?

- Every dependency in F has a single attribute for its right hand side.
- We cannot replace any dependency $X \rightarrow A$ in F with a dependency $Y \rightarrow A$ where Y is a proper subset of X and still have a set of dependency that is equivalent to F.

- We cannot remove any dependency from F and still have set of dependency that is equivalent to F.

11. What is Multivalued dependency?

Multivalued dependency denoted by $X \rightarrow\rightarrow Y$ specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation r of R: if two tuples t1 and t2 exist in r such that t1[X] = t2[X] then t3 and t4 should also exist in r with the following properties

- t3[x] = t4[X] = t1[X] = t2[X]
- t3[Y] = t1[Y] and t4[Y] = t2[Y]
- t3[Z] = t2[Z] and t4[Z] = t1[Z]
  where [Z = (R-(X U Y))]

12. What is Lossless join property?

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

13. What is 4NF?

A relation schema R is said to be in 4NF if for every Multivalued dependency $X \rightarrow\rightarrow Y$ that holds over R, one of following is true

- X is subset or equal to (or) XY = R.
- X is a super key.

14. What is 5NF?

A Relation schema R is said to be 5NF if for every join dependency {R1, R2,..., Rn} that holds R, one the following is true

- Ri = R for some i.
- The join dependency is implied by the set of FD, over R in which the left side is key of R.

15. What is Domain-Key Normal Form?

A relation is said to be in DKNF if all constraints and dependencies that should hold on the the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.

16. What do you mean by atomicity and aggregation?

A*tomicity:*

Either all actions are carried out or none are. Users should not have to worry about the effect of incomplete transactions. DBMS ensures this by undoing the actions of incomplete transactions.

A*ggregation*:

A concept which is used to model a relationship between a collection of entities and relationships. It is used when we need to express a relationship among relationships.

17. What is a Phantom Deadlock?

In distributed deadlock detection, the delay in propagating local information might cause the deadlock detection algorithms to identify deadlocks that do not really exist. Such situations are called phantom deadlocks and they lead to unnecessary aborts.

18. What is a checkpoint and when does it occur?

A checkpoint is like a snapshot of the DBMS state. By taking checkpoints, the DBMS can reduce the amount of work to be done during restart in the event of subsequent crashes.

19. How can you find the minimal key of relational schema?

Minimal key is one which can identify each tuple of the given relation schema uniquely. For finding the minimal key it is required to find the closure that is the set of all attributes that are dependent on any given set of attributes under the given set of functional dependency.

A**lgo. I** Determining X^+, closure for X, given set of FDs F

1. Set $X^+ = X$
2. Set Old $X^+ = X^+$
3. For each FD $Y \rightarrow Z$ in F and if Y belongs to X^+ then add Z to X^+
4. Repeat steps 2 and 3 until Old $X^+ = X^+$

A**lgo.II** Determining minimal K for relation schema R, given set of FDs F

1. Set K to R that is make K a set of all attributes in R
2. For each attribute A in K
   A. Compute $(K - A)^+$ with respect to F
   b. If $(K - A)^+ = R$ then set $K = (K - A)^+$

20. What do you understand by dependency preservation?

Given a relation R and a set of FDs F, dependency preservation states that the closure of the union of the projection of F on each decomposed relation Ri is equal to the closure of F. i.e.,

$((\Pi_{R1}(F)) \cup \ldots \cup (\Pi_{Rn}(F)))^+ = F^+$

if decomposition is not dependency preserving, then some dependency is lost in the decomposition.

OBJECTIVE TYPE QUESTIONS

1. Software that defines a database, stores the data, supports a query language, produces reports and creates data entry screens is a:

A. Data dictionary
B. Database management system (DBMS)
C. Decision support system
D. Relational database

2. The separation of the data definition from the program is known as:

A. Data dictionary
B. Data independence
C. Data integrity
D. Referential integrity

3. In the client / server model, the database:
A. Is downloaded to the client upon request
B. Is shared by both the client and server
C. Resides on the client side
D. Resides on the server side

4. The traditional storage of data that is organised by customer, stored in separate folders in filing cabinets is an example of what type of 'database' system?
A. Hierarchical B. Network
C. Object oriented D. Relational

5. The database design that consists of multiple tables that are linked together through matching data stored in each table is called a:
A. Hierarchical database
B. Network database
C. Object oriented database
D. Relational database

6. What is the main limitation of Hierarchical Databases?
A. Limited capacity (unable to hold much data.
B. Limited flexibility in accessing data
C. Overhead associated with maintaining indexes
D. The performance of the database is poor

7. An abstract data type is used to:
A. Link data from remote databases
B. Prevent users from getting to database security information
C. Provide a conceptual view of the data so it is easier to understand
D. Store complex data structure to represent the properties of objects

8. Which component of the database management system (DBMS) most affects the ability to handle large problems (scalability)?
A. Data Storage Subsystem
B. Database Engine
C. Query Processor
D. Security Subsystem

9. The primary difference between the Relational database (RDB) and Object Oriented database (OODB) models is:
A. OODB incorporates methods in with the definition of the data structure, while RDB does not
B. OODB supports multiple objects in the same database while RDB only supports a single table per database
C. RDB allows the definition of the relationships between the different tables, while OODB does not allow the relationships to be defined between objects
D. RDB supports indexes, while OODB does not support indexes

10. Which of the following items is not the advantage of a DBMS?
A. Improved ability to enforce standards
B. Improved data consistency
C. Local control over the data
D. Minimal data redundancy

11. When building a database, the data dealing with an entity is modeled as a:
A. Attribute B. Class
C. Object D. Table

12. Database system modelers use this type of diagram to graphically represent both the data structure and how the different objects are interrelated.
A. Class Diagram B. Data Diagram
C. Object Diagram
D. Table Relationship Diagram

13. In relational database model, after conceptually designing your database, the information contained in a single class would be stored in a:
A. Database B. Field
C. Property D. Table

14. The property (or set of properties) that uniquely defines each row in a table is called the:
A. Identifier B. Index
C. Primary key D. Symmetric key

15. Business rules can be represented in the database through:
A. Associations (or relationships)
B. Attributes C. Properties
D. Secondary keys

16. The association role defines:
A. How tables are related in the database
B. The relationship between the class diagram and the tables in the database
C. The tables that contains each attribute
D. Which attribute is the table's primary key

17. The purpose of an N-Ary association is:
- A. To capture a parent-child relationship
- B. To deal with one to many relationships
- C. To deal with relationships that involve more than two tables
- D. To represent an inheritance relationship

18. A reflexive association is one where one class is:
- A. Broken down into special cases
- B. Combined with multiple other classes
- C. Combined with one other class
- D. Linked back to itself

19. Which of the following statements is not correct?
- A. A primary goal of a database system is to share data with multiple users
- B. It is possible to change a method or property inherited from a higher level class
- C. While companies collect data all the time, the structure of the data changes very often.
- D. In a client / server environment, data independence causes client side applications to be essentially independent of the database stored on the server side.

20. Which of the following statements is not correct?
- A. Data Normalisation is the process of defining the table structure
- B. The purpose of class diagrams is to model the interrelationships between the different classes in the database
- C. Individual objects are stored as rows in a table
- D. Properties of an object are stored as columns in a table.

21. Which of the following statements is not correct?
- A. The primary key must be unique for a given table
- B. Specifying a zero (0) for the lower bound for the association multiplicity on a class diagram indicates that the item is required
- C. Specifying a one (1) for the lower bound for the association multiplicity on a class diagram indicates that the item is required
- D. Most databases allow multiple records that are identical (i.e., records that have the same values for all properties).

22. Which of the following statements is not correct?
- A. All many-to-many relationships must be converted to a set of one-to-many relationships by adding a new entity
- B. In a one-to-one relationship between two classes, the two classes are generally described by one table in relational database model
- C. Encapsulation provides some security and control features
- D. Properties and functions can be protected from other areas of the applications

23. There is a relational schema which has k attributes. The domain of each attribute consists of exactly 2 elements. A table is defined as subset of tuples where in each tuple, a value is defined for each of the k attributes. The minimum value of k needed for the number of distinct tuples to exceed 10^9 is

A. 5 B. 9 C. 17 D. 32

E. 24

24. The relation PAYMENT(Cust-ID, Account, Amount_Paid, Date_Paid, Type_Payment, Discount). Assume that a customer may have more than one account and that he or she can make several payments on any day but not more than one payment per day can be applied to each account. The key for the relation can be:
- A. Cust-ID, Account
- B. Account, Date_Paid
- C. Cust-ID, Account, Date_Paid
- D. None

25. The relation STORE(Location, No-of-Employees, Total-Monthly-Sales, Manager, City)
- A. No key available for this
- B. If we assign STORE_ID as an extra attribute it can act like PK
- C. If we restrict a person to manage only one store then Manager attribute can become as PK
- D. If we restrict a person can manage all the stores in a City then Manager attribute can become as PK
- E. None

26. Candidate keys for a relation R(X, Y, Z, W, P) are

A. 1 B. 2 C. 3 D. 4

E. None

27. The valid FD's in the following relation are

A	B	C
f	e	e
d	e	e
b	c	e
a	c	d
a	b	c

A. $A \rightarrow B$ B. $AB \rightarrow C$

C. $A \rightarrow C$ D. $AC \rightarrow B$

E. None

28. Out of the following FD'S F={ $A \rightarrow BC$, $E \rightarrow C$, $D \rightarrow AEF$, $ABF \rightarrow BD$}. Then the left reduced set is

A. { $A \rightarrow BC$, $E \rightarrow C$, $D \rightarrow AEF$, $ABF \rightarrow BD$}

B. { A→BC, E→C, D→AE, AB→BD}
C. { A→BC, E→C, D→AEF, AF→BD}
D. { A→BC, E→C, D→AEF, AB→BD}
E. None

29. The canonical cover of following FD'S F={ A → BC, E → C, D → AEF, ABF → BD}.
A. {A→B, A→C, E→C, D→A, D→E, E→F, AF→D}
b. {A→B, A→C, E→C, D→A, D→E, D→F, AF→D}
c. {A→B, A→C, E→C, D→A, D→E, E→F, A→D}
d. {A→B, A→C, E→C, D→A, D→E, E→F, AF→D}
E. None

30. If a relation contains only one key (single attribute) and no transitive dependencies then select more valid one.
A. 1NF B. 2NF
C. Every non prime attribute will be fully functionally dependent on key
D. 3NF
E. None

31. The number of candidate keys for a relation (ABCDEFGH) with the given FD's {A→C, B→D,G→H, E→F, C→G}
A. 1 B. 2 C. 3 D. 4
E. None

32. The canonical cover of set of FD's F={ A→BC, B→C, A→B, AB→C}
A. A→C A→B B. A→B A→B
C. A→B B→C D. None

33. List the candidate keys for the relation R(ABCDE) with FD's F={ A→BC, B→D, CD→E, E→A}
A. A B. CD C. E D. None

34. Construct a B-Tree for the following data (2,3,5,7,11,17,19,23,29,31). Assume in a node at most 4 keys can be stored and atleast 2. number of splitting operations required are
A. 1 B. 2 C. 3 D. 4

ANSWER KEY

1. B	**2.** B	**3.** D	**4.** A
5. D	**6.** B	**7.** D	**8.** B
9. A	**10.** C	**11.** A	**12.** A
13. D	**14.** C	**15.** A	**16.** A
17. C	**18.** D	**19.** C	**20.** A
21. B	**22.** B	**23.** D	**24.** C
25. B	**26.** B	**27.** B	**28.** C
29. B	**30.** D	**31.** E	**32.** C
33. A	**34.** B		

Previous Years' GATE Questions

1. An index is clustered, if **(GATE 2013)**
A. It is on a set of fields that form a candidate key.
B. It is on a set of fields that include the primary key.
C. The data records of the file are organised in the same order as the data entries of the index.
D. The data records of the file are organised not in the same order as the data entries of the index.

2. Consider the following relational schema. **(GATE 2013)**

Students(rollno: integer, sname: string)
Courses(courseno: integer, cname: string)
Registration(rollno: integer, courseno: integer, percent: real)

Which of the following queries are equivalent to this query in English?

"Find the distinct names of all students who score more than 90% in the course numbered 107"

(I) SELECT DISTINCT S.sname
FROM Students as S. Registration as R
WHERE R. rollno = S.rollno AND R. Courseno = 107 AND R. percent > 90

(II) $\Pi_{sname}\ (\sigma_{courseno\ =\ 107\ \wedge\ percent\ >\ 90}$ (Registration $\bowtie$ Students)

(III) $\{T \mid \exists S \in$ Students, $\exists R \in$ Registration (S. rollno = R. rollno $\wedge$ R. courseno = 107 $\wedge$ R. percent > 90 $\wedge$ T. sname = S. sname)$\}$

(IV) $\{<S_N> \mid \exists S_R\ \exists R_P\ (<S_R, S_N> \in$ Students $\wedge <S_R, 107, R_P> \in$ Registration $\wedge\ R_P > 90)\}$

A. I, II, III and IV B. I, II and III only
C. I, II and IV only D. II, III and IV only

Relation *R* has eight attributes *ABCDEFGH*. Fields of R contain only atomic values.

F={CH→G, A→BC, B→CFH, E→A, F→EG} is a set of functional dependencies (FDs) so that *F+* is exactly the set of FDs that hold for R.

3. How many candidate keys does the relation R have? **(GATE 2013)**
A. 3 B. 4 C. 5 D. 6

4. The relation R is **(GATE 2013)**
A. In 1NF, but not in 2NF.
B. In 2NF, but not in 3NF.
C. In 3NF, but not in BCNF.
D. In BCNF.

5. Which of the following statements are true about an SQL query? **(GATE 2012)**

P: An SQL query can contain a HAVING clause even if it does not a GROUP BY clause

Q: An SQL query can contain a HAVING clause only if it has a GROUP BY clause

R: All attributes used in the GROUP BY clause must appear in the SELECT clause

S: Not all attributes used in the GROUP BY clause need to apper in the SELECT clause

A. P and R B. P and S

C. Q and R D. Q and S

6. Given the basic ER and relational models, which of the following is INCORRECT? **(GATE 2012)**

A. An attributes of an entity can have more that one value

B. An attribute of an entity can be composite

C. In a row of a relational table, an attribute can have more than one value

D. In a row of a relational table, an attribute can have exactly one value or a NULL value

7. Suppose (A, B) and (C, D) are two relation schemas. Let r1 and r2 be the corresponding relation instances. B is a foreign key that refers to C in r2. If data in r1 and r2 satisfy referential integrity constraints, which of the following is always true? **(GATE 2012)**

A. $\Pi_B(r_1) - \Pi_C(r_2) = \varnothing$ B. $\Pi_C(r_2) - \Pi_B(r_1) = \varnothing$

C. $\Pi_B(r_1) = \Pi_C(r_2)$ D. $\Pi_B(r_1) - \Pi_C(r_2) \neq \varnothing$

Explanation:

B is a foreign key in r1 that refers to C in r2. r1 and r2 satisfy referential integrity constraints. So every value that exists in column B of r1 must also exist in column C of r2.

8. Which of the following is true? **(GATE 2012)**

A. Every relation in 2NF is also in BCNF

B. A relation R is in 3NF if every non-prime attribute of R is fully functionally dependent on every key of R

C. Every relation in BCNF is also in 3NF

D. No relation can be in both BCNF and 3NF

Explanation:

BCNF is a stronger version 3NF. So every relation in BCNF will also be in 3NF.

9. Consider the following transactions with data items P and Q initialised to zero:

T1: read (P) ;
read (Q) ;
if P = 0 then Q : = Q + 1 ;
write (Q) ;

T2: read (Q) ;
read (P) ;
if Q = 0 then P : = P + 1 ;
write (P) ;

Any non-serial interleaving of T1 and T2 for concurrent execution leads to **(GATE 2012)**

A. A serialisable schedule

B. A schedule that is not conflict serialisable

C. A conflict serialisable schedule

D. A schedule for which a precedence graph cannot be drawn

Explanation:

Two or more actions are said to be in conflict if:

1. The actions belong to different transactions.
2. At least one of the actions is a write operation.
3. The actions access the same object (read or write).

The schedules S1 and S2 are said to be conflict-equivalent if the following conditions are satisfied:

1. Both schedules S1 and S2 involve the same set of transactions (including ordering of actions within each transaction).
2. The order of each pair of conflicting actions in S1 and S2 are the same.

A schedule is said to be conflict-serialisable when the schedule is conflict-equivalent to one or more serial schedules.

In the given scenario, there are two possible serial schedules:

1. T1 followed by T2
2. T2 followed by T1.

In both of the serial schedules, one of the transactions reads the value written by other transaction as a first step. Therefore, any non-serial interleaving of T1 and T2 will not be conflict serialisable.

10. Consider the following relations A, B, C. How many tuples does the result of the following relational algebra expression contain? Assume that the schema of A U B is the same as that of A. **(GATE 2012)**

$(A \cup B) \bowtie_{A.Id>40 \text{ V } CId<15} C$

Table A

Id	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11

Table B

Id Name Age

---

15 Shreya 24

25 Hari 40

98 Rohit 20

99 Rohit 11

Table C

Id Phone Area

---

10 2200 02

99 2100 01

A. 7 B. 4 C. 5 D. 9

Explanation :

Result of AUB will be following table

Id Name Age

---

12 Arun 60

15 Shreya 24

99 Rohit 11

25 Hari 40

98 Rohit 20

The result of given relational algebra expression will be

Id Name	Age	Id	Phone	Area
12 Arun	60	10	2200	02
15 Shreya	24	10	2200	02
99 Rohit	11	10	2200	02
25 Hari	40	10	2200	02
98 Rohit	20	10	2200	02
99 Rohit	11	99	2100	01
98 Rohit	20	99	2100	01

11. Consider the above tables A, B and C. How many tuples does the result of the following SQL query contains? **(GATE 2012)**

```
SELECT A.id
FROM A
WHERE A.age > ALL (SELECT B.age
FROM B
WHERE B. name = "arun")
```

A. 4 B. 3

C. 0 D. 1

Explanation:

The meaning of "ALL" is the A.Age should be greater than all the values returned by the subquery. There is no entry with name "arun" in table B. So the subquery will return null. If a subquery returns null, then the condition becomes true for all rows of A. So all rows of table A are selected.

12. Consider a relational table with a single record for each registered student with the following attributes.

1. Registration_Number:< Unique registration number for each registered student
2. UID: Unique Identity number, unique at the national level for each citizen
3. BankAccount_Number: Unique account number at the bank. A student can have multiple accounts or joint accounts. This attributes stores the primary account number
4. Name: Name of the Student
5. Hostel_Room: Room number of the hostel

Which of the following options is incorrect?

(GATE 2011)

A. BankAccount_Number is a candidate key

B. Registration_Number can be a primary key

C. UID is a candidate key if all students are from the same country

D. If S is a superkey such that $S \cap UID$ is null then $S \cup UID$ is also a superkey

Explanation:

A Candidate Key value must uniquely identify the corresponding row in table. BankAccount_Number is not a candidate key. As per the question "A student can have multiple accounts or joint accounts. This attributes stores the primary account number". If two students have a joint account and if the joint account is their primary account, then BankAccount_Number value cannot uniquely identify a row.

13. Consider a relational table r with sufficient number of records, having attributes A1, A2,…, An and let $1 \Leftarrow p \Leftarrow n$. Two queries Q1 and Q2 are given below.

(GATE 2011)

Q1 : $\pi_{A1.....A_n}(\sigma_{A_p = c}(r)$ where c is a const

Q1 : $\pi_{A1.....A_n}(\sigma_{c_1 \le A_p \le c_2}(r)$ where c_1 and c_2 are constants.

The database can be configured to do ordered indexing on Ap or hashing on Ap. Which of the following statements is true?

A. Ordered indexing will always outperform hashing for both queries

B. Hashing will always outperform ordered indexing for both queries

C. Hashing will outperform ordered indexing on Q1, but not on Q2

D. Hashing will outperform ordered indexing on Q2, but not on Q1.

Explanation:

If record are accessed for a particular value from table, hashing will do better. If records are accessed in a range of values, ordered indexing will perform better.

14. Database table by name Loan_Records is given below. **(GATE 2011)**

Borrower	Bank Manager	Loan_Amount
Ramesh	Sunderajan	10000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderajan	7000.00

What is the output of the following SQL query?

```
SELECT Count(*)
FROM ( (SELECT Borrower, Bank_Manager
   FROM Loan_Records) AS S
   NATURAL JOIN (SELECT Bank_Manager,
                Loan_Amount
          FROM Loan_Records) AS T );
```

A. 3 B. 9 C. 5 D. 6

Explanation :

Following will be contents of temporary table S

Borrower	Bank_Manager
Ramesh	Sunderajan
Suresh	Ramgqpal
Mahesh	Sunderjan

Following will be contents of temporary table T

Bank_Manager	Loan_Amount
Sunderajan	10000.00
Ramgopal	5000.00
Sunderjan	7000.00

Following will be the result of natural join of above two tables. The key thing to note is that the natural join happens on column name with same name which is Bank_Manager in the above example. "Sunderjan" appears two times in Bank_Manager column, so their will be four entries with Bank_Manager as "Sunderjan".

Borrower	Bank_Manager	Load_Amount
Ramesh	Sunderajan	10000.00
Ramesh	Sunderajan	7000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderajan	10000.00
Mahesh	Sunderajan	7000.00

15. The table at any point in time. Using MX and MY, new records are inserted in the table 128 times with X and Y values being MX+1, 2*MY+1 respectively. It may be noted that each time after the insertion, values of MX and MY change. What will be the output of the following SQL query after the steps mentioned above are carried out? **(GATE 2011)**

SELECT Y FROM T WHERE X = 7;

A. 127 B. 255 C. 129 D. 257

Explanation:

```
X   Y
-------
1   1
2   3
3   7
4   15
5   31
6   63
7   127
  .....
  .....
```

16. A relational schema for a train reservation database is given below.

Passenger (pid, pname, age)

Reservation (pid, class, tid) **(GATE 2010)**

Table: Passenger

```
pid pname age
-----------------
0 Sachin   65
1 Rahul    66
2 Sourav   67
3 Anil     69
```

Table : Reservation

```
pid class tid
---------------
0 AC   8200
1 AC   8201
2 SC   8201
5 AC   8203
1 SC   8204
3 AC   8202
```

What pids are returned by the following SQL query for the above instance of the tables?

```
SLECT pid
FROM Reservation,
WHERE class 'AC' AND
  EXISTS (SELECT *
    FROM Passenger
    WHERE age > 65 AND
    Passenger. pid = Reservation.pid)
```

A. 1, 0 B. 1, 2 C. 1, 3 D. 1, 5

Explanation :

When a subquery uses values from outer query, the subquery is called correlated. The correlated subquery is evaluated once for each row processed by the outer query. The outer query selects 4 entries (with pids as 0, 1, 5, 3) from Reservation table. Out of these selected entries, the subquery returns Non-Null values only for 1 and 3.

17. Which of the following concurrency control protocols ensure both conflict serialzability and freedom from deadlock? **(GATE 2010)**

I. 2-phase locking

II. Time-stamp ordering

A. I only B. II only

C. Both I and II D. Neither I nor II

Explanation:

Two phase locking is a concurrency control method that guarantees Serialisability. The protocol utilises locks, applied by a transaction to data, which may block (interpreted as signals to stop) other transactions from accessing the same data during the transaction's life. 2PL may be lead to deadlocks that result from the mutual blocking of two or more transactions. See the following situation, neither T3 nor T4 can make progress.

T_3	T_4
lock-X (B)	
read (B)	
$B := B - 50$	
write (B)	
	lock-S (A)
	read (A)
	lock-S (B)
lock-X(A)	

Timestamp based concurrency control algorithm is a non-lock concurrency control method. In Timestamp based method, deadlock cannot occur as no transaction ever waits.

18. Consider the following schedule for transactions T1, T2 and T3: **(GATE 2010)**

T1	T2	T3
Read (*X*)		
	Read (*Y*)	
		Read (*Y*)
	Write (*Y*)	
Write (*X*)		
		Write (*X*)
	Read (*X*)	
	Write (*X*)	

Which one of the schedules below is the correct serialization of the above?

A. T1 T3 T2 B. T2 T1 T3

C. T2 T3 T1 D. T3 T1 T2

Explanation:

T1 can complete before T2 and T3 as there is no conflict between Write(X) of T1 and the operations in T2 and T3 which occur before Write(X) of T1 in the above diagram.

T3 should can complete before T2 as the Read(Y) of T3 doesn't conflict with Read(Y) of T2. Similarly, Write(X) of T3 doesn't conflict with Read(Y) and Write(Y) operations of T2.

Another way to solve this question is to create a dependency graph and topologically sort the dependency graph. After topologically sorting, we can see the sequence T1, T3, T2.

19. The following functional dependencies hold for relations R(A, B, C) and S(B, D, E):

$B \rightarrow A$,

$A \rightarrow C$

The relation R contains 200 tuples and the relation S contains 100 tuples. What is the maximum number of tuples possible in the natural join $R \rhd\lhd S$ (R natural join S) **(GATE 2010)**

A. 100 B. 200

D. 300 D. 2000

Explanation:

From the given set of functional dependencies, it can be observed that B is a candidate key of R. So all 200 values of B must be unique in R. There is no functional dependency given for S. To get the maximum number of tuples in output, there can be two possibilities for S.

(1) All 100 values of B in S are same and there is an entry in R that matches with this value. In this case, we get 100 tuples in output.

(2) All 100 values of B in S are different and these values are present in R also. In this case also, we get 100 tuples.

20. Consider two transactions T1 and T2, and four schedules S1, S2, S3, S4 of T1 and T2 as given below: **(GATE 2009)**

T1 = R1[X] W1[X] W1[Y]

T2 = R2[X] R2[Y] W2[Y]

S1 = R1[X] R2[X] R2[Y] W1[X] W1[Y] W2[Y]

S2 = R1[X] R2[X] R2[Y] W1[X] W2[Y] W1[Y]

S3 = R1[X] W1[X] R2[X] W1[Y] R2[Y] W2[Y]

S1 = R1[X] R2[Y]R2[X]W1[X] W1[Y] W2[Y]

Which of the above schedules are conflict-serialisable?

A. S1 and S2 B. S2 and S3

C. S3 only D. S4 only

Explanation:

There can be two possible serial schedules T1 T2 and T2 T1. The serial schedule T1 T2 has the following sequence of operations

R1[X] W1[X] W1[Y] R2[X] R2[Y] W2[Y]

And the schedule T2 T1 has the following sequence of operations.

R2[X] R2[Y] W2[Y] R1[X] W1[X] W1[Y]

The Schedule S2 is conflict-equivalent to T2 T1 and S3 is conflict-equivalent to T1 T2.

21. Let R and S be relational schemes such that R={a,b,c} and S={c}. Now consider the following queries on the database: **(GATE 2009)**

I. $\pi_{R-S}(r) - \pi_{R-S}(\pi_{R-S}(r) \times S - \pi_{R-S, S}(r))$

II. $\{t | t \epsilon \pi_{R-S}(r) \wedge \forall u \epsilon_r (\exists v \epsilon s (u = v[s] \wedge t = v[R-S]))\}$

III. $\{t | t \epsilon \pi_{R-S}(r) \wedge \forall v \epsilon r (\exists u \epsilon s (u = v[s] \wedge t = v[R-S]))\}$

IV. SELECT R.a, R.b
FROM R,S
WHERE R.c = S.c

Which of the above queries are equivalent?

A. I and II B. I and III

C. II and IV D. III and IV

22. Consider the following relational schema:

Suppliers (sid:integer, sname:string, city:string, street:string)

Parts(pid:integer, pname:string, color:string)

Catalog(sid:integer, pid:integer, cost:real)

Consider the following relational query on the above database: **(GATE 2009)**

```
SELECT S.sname
  FROM Suppliers S
    WHERE S.sid NOT IN (SELECT C.sid
        FROM Catalog C
        WHERE C.pid NOT IN (SELECT P.pid
            FROM Parts P
            WHERE P.color<> 'blue'))
```

Assume that relations corresponding to the above schema are not empty. Which one of the following is the correct interpretation of the above query?

A. To find the names of all suppliers who have supplied a non-blue part.

B. To find the names of all suppliers who have not supplied a non-blue part.

C. To find the names of all suppliers who have supplied only blue parts.

D. To find the names of all suppliers who have not supplied only blue parts.

Explanation :

The subquery "*SELECT P.pid FROM Parts P WHERE P.color<> 'blue'*" gives pids of parts which are not blue. The bigger subquery "*SELECT C.sid FROM Catalog C WHERE C.pid NOT IN (SELECT P.pid FROM Parts P WHERE P.color<> 'blue')*" gives sids of all those suppliers who have supplied blue parts. The complete query gives the names of all suppliers who have not supplied a non-blue part

23. Assume that, in the suppliers relation above, each supplier and each street within a city has a unique name, and (sname, city) forms a candidate key. No other functional dependencies are implied other than those implied by primary and candidate keys. Which one of the following is true about the above schema? **(GATE 2009)**

A. The schema is in BCNF

B. The schema is in 3NF but not in BCNF

C. The schema is in 2NF but not in 3NF

D. The schema is not in 2NF

Explanation :

The schema is in BCNF as all attributes depend only on a superkey (Note that primary and candidate keys are also superkeys).

24. Let R and S be two relations with the following schema **(GATE 2008)**

R (P, Q, R1, R2, R3)

S (P, Q, S1, S2)

Where {P, Q} is the key for both schemas. Which of the following queries are equivalent?

I. $\Pi_P (R \bowtie S)$

II. $\Pi_P (R) \bowtie \Pi_P (S)$

III. $\Pi_P (\Pi_{P,Q} (R) \cap \Pi_{P,Q} (S))$

IV. $\Pi_P (\Pi_{P,Q} (R) - (\Pi_{P,Q} (R) - (\Pi_{P,Q} (S)))$

A. Only I and II B. Only I and III

C. Only I, II and III D. Only I, III and IV

Explanation:

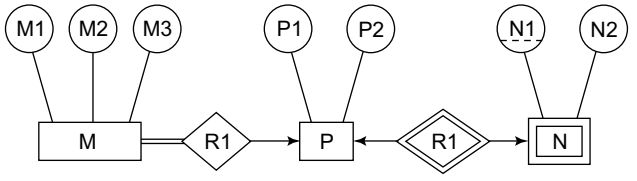
In I, Ps from natural join of R and S are selected.

In III, all Ps from intersection of (P, Q) pairs present in R and S.

IV is also equivalent to III because (R–(R–S)) = R ∩ S.

II is not equivalent as it may also include Ps where Qs are not same in R and S.

25. Consider the following ER diagram. **(GATE 2008)**



The minimum number of tables needed to represent M, N, P, R1, R2 is

A. 2 B. 3 C. 4 D. 5

Explanation:

Many-to-one and one-to-many relationship sets that the total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side. Since R1 is many to one and participation of M is total, M and R1 can be combined to form the table {M1, M2, M3, P1}. N is a week entity set, so it can be combined with P.

26. Which of the following is a correct attribute set for one of the tables for the correct answer to the above question? **(GATE 2008)**

A. {M1, M2, M3, P1} B. {M1, P1, N1, N2}

C. {M1, P1, N1} D. {M1, P1}

27. Consider the following relational schemes for a library database: **(GATE 2008)**

Book (Title, Author, Catalog_no, Publisher, Year, Price)

Collection (Title, Author, Catalog_no)

within the following functional dependencies:

I. Title Author → Catalog_no

II. Catalog_no → Title Author Publisher Year

III. Publisher Title Year → Price

Assume {Author, Title} is the key for both schemes. Which of the following statements is true?

(GATE 2008)

A. Both Book and Collection are in BCNF

B. Both Book and Collection are in 3NF only

C. Book is in 2NF and Collection is in 3NF

D. Both Book and Collection are in 2NF only

Explanation:

Table Collection is in BCNF as there is only one functional dependency "Title Author → Catalog_no" and {Author, Title} is key for collection. Book is not in BCNF because Catalog_no is not a key and there is a functional dependency "Catalog_no → Title Author Publisher Year". Book is not in 3NF because non-prime attributes (Publisher Year) are transitively dependent on key [Title, Author]. Book is in 2NF because every non-prime attribute of the table is either dependent on the key [Title, Author], or on another non prime attribute.

28. Consider the following log sequence of two transactions on a bank account, with initial balance 12000, that transfer 2000 to a

1. T1 start
2. T1 B old=12000 new=10000
3. T1 M old=0 new=2000
4. T1 commit
5. T2 start
6. T2 B old=10000 new=10500
7. T2 commit

Suppose the database system crashes just before log record 7 is written. When the system is restarted, which one statement is true of the recovery procedure? mortagage payment and then apply a 5% interest. **(GATE 2006)**

A. We must redo log record 6 to set B to 10500

B. We must undo log record 6 to set B to 10000 and then redo log records 2 and 3

C. We need not redo log records 2 and 3 because transaction T1 has committed

D. We can apply redo and undo operations in arbitrary order because they are idempotent.

Explanation :

Once a transaction is committed, no need to redo or undo operations.

29. Consider the relation enrolled (student, course) in which (student, course) is the primary key, and the relation paid (student, amount) where student is the primary key. Assume no null values and no foreign keys or integrity constraints. Given the following four queries:

Query1: select student from enrolled where student in (select student from paid)

Query2: select student from paid where student in (select student from enrolled)

Query3: select E.student from enrolled E, paid P where E.student = P.student

Query4: select student from paid where exists

(select * from enrolled where enrolled.student = paid. student)

Which one of the following statements is correct?

(GATE 2006)

A. All queries return identical row sets for any database

B. Query2 and Query4 return identical row sets for all databases but there exist databases for which Query1 and Query2 return different row sets.

C. There exist databases for which Query3 returns strictly fewer rows than Query2.

D. There exist databases for which Query4 will encounter an integrity violation at runtime.

Explanation:

The output of Query2, Query3 and Query4 will be identical. Query1 may produce duplicate rows. But rowset produced by all of them will be same.

Table enrolled

student course

---

abc c1

xyz c1

abc c2

pqr c1

Table paid

student amount

---

abc 20000

xyz 10000

rst 10000

Output of Query 1

abc

abc

xyz

Output of Query 2

abc

xyz

Output of Query 3

abc

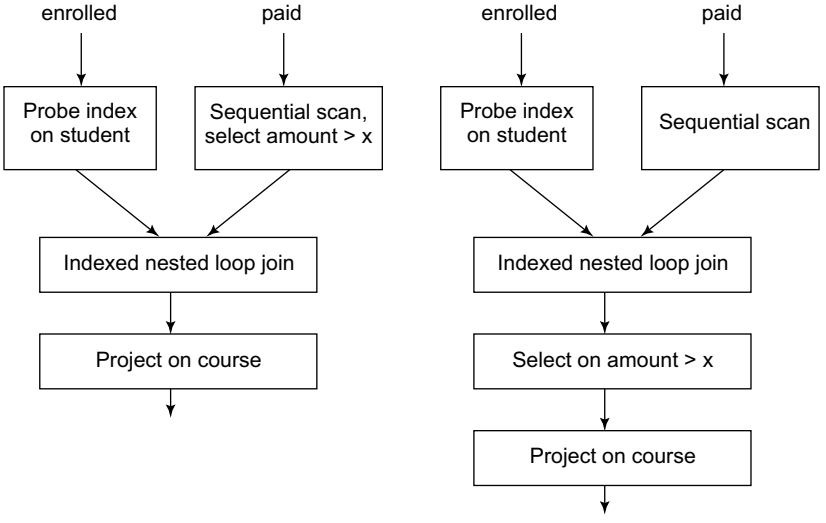
xyz

Output of Query 4

abc

xyz

30. Consider the relation enrolled (student, course) in which (student, course) is the primary key, and the relation paid (student, amount), where student is the primary key. Assume no null values and no foreign keys or integrity constraints. Assume that amounts 6000, 7000, 8000, 9000 and 10000 were each paid by 20% of the students. Consider these query plans (Plan 1 on left, Plan 2 on right) to "list all courses taken by students who have paid more than x". **(GATE 2006)**



A disk seek takes 4ms, disk data transfer bandwidth is 300 MB/s and checking a tuple to see if amount is greater than x takes 10 micro-seconds. Which of the following statements is correct?

A. Plan 1 and Plan 2 will not output identical row sets for all databases.
B. A course may be listed more than once in the output of Plan 1 for some databases
C. For x = 5000, Plan 1 executes faster than Plan 2 for all databases.
D. For x = 9000, Plan I executes slower than Plan 2 for all databases.

Explanation :

Assuming that large enough memory is available for all data needed. Both plans need to load both tables courses and enrolled. So disk access time is same for both plans.

Plan 2 does lesser number of comparisons compared to plan 1.

1. Join operation will require more comparisons as the second table will have more rows in plan 2 compared to plan 1.
2. The joined table of two tables will will have more rows, so more comparisons are needed to find amounts greater than x.

31. The following functional dependencies are given: **(GATE 2006)**

$AB \rightarrow CD$, $AF \rightarrow D$, $DE \rightarrow F$, $C \rightarrow G$, $F \rightarrow E$, $G \rightarrow A$

Which one of the following options is false?

A. CF+ = {ACDEFG}
B. BG+ = {ABCDG}
C. AF+ = {ACDEFG}
D. AB+ = {ABCDFG}

Explanation :

Closure of AF or AF+ = {ADEF}, closure of AF doesn't contain C and G.

Option (D) Also looks correct. AB+ = {ABCDG}, closure of AB doesn't contain F.

32. Which one of the following statements about normal forms is false? **(GATE 2005)**

A. BCNF is stricter than 3NF
B. Lossless, dependency-preserving decomposition into 3NF is always possible
C. Lossless, dependency-preserving decomposition into BCNF is always possible
D. Any relation with two attributes is in BCNF

Explanation :

It is not always possible to decompose a table in BCNF and preserve dependencies. For example, a set of functional dependencies $\{AB \rightarrow C, C \rightarrow B\}$ cannot be decomposed in BCNF. See this for more details.

33. The following table has two attributes A and C where A is the primary key and C is the foreign key referencing A with on-delete cascade. **(GATE 2005)**

A	C
2	4
3	4
4	3
5	2
7	2
9	5
6	4

The set of all tuples that must be additionally deleted to preserve referential integrity when the tuple (2,4) is deleted is:

A. (3,4) and (6,4)
B. (5,2) and (7,2)
C. (5,2), (7,2) and (9,5)
D. (3,4), (4,3) and (6,4)

Explanation :

When (2,4) is deleted. Since C is a foreign key referring A with delete on cascade, all entries with value 2 in C must be deleted. So (5, 2) and (7, 2) are deleted. As a result of this 5 and 7 are deleted from A which causes (9, 5) to be deleted.

34. The relation book (<u>title</u>, price) contains the titles and prices of different books. Assuming that no two books have the same price, what does the following SQL query list? **(GATE 2005)**

```
select title
from book as B
where (select count(*)
    from book as T
    where T.price > B.price) < 5
```

A. Titles of the four most expensive books
B. Title of the fifth most inexpensive book
C. Title of the fifth most expensive book
D. Titles of the five most expensive books

Explanation:

When a subquery uses values from outer query, the subquery is called correlated subquery. The correlated subquery is evaluated once for each row processed by the outer query.

The outer query selects all titles from book table. For every selected book, the subquery returns count of those books which are more expensive than the se-

lected book. The where clause of outer query will be true for 5 most expensive books. For example count (*) will be 0 for the most expensive book and count(*) will be 1 for second most expensive book.

35. Let r be a relation instance with schema R = (A, B, C, D). We define r1 = 'select A,B,C from r' and r2 = 'select A, D from r'. Let s = r1 * r2 where * denotes natural join. Given that the decomposition of r into r1 and r2 is lossy, which one of the following is true? **(GATE 2005)**

A. s is subset of r B. r U s = r
C. r is a subset of s D. r * s = s

Explanation:

Consider the following example with lossy decomposition of r into r1 and r2. We can see that r is a subset of s.

Table r

A	B	C	D
1	10	100	1000
1	20	200	1000
1	20	200	1001

Table r1

A	B	C
1	10	100
1	20	200

Table r2

A	D
1	1000
1	1001

Table s (natural join of r1 and r2)

A	B	C	D
1	10	100	1000
1	20	200	1000
1	20	100	1001
1	20	200	1001

36. Let E1 and E2 be two entities in an E/R diagram with simple single-valued attributes. R1 and R2 are two relationships between E1 and E2, where R1 is one-to-many and R2 is many-to-many. R1 and R2 do not have any attributes of their own. What is the minimum number of tables required to represent this situation in the relational model? **(GATE 2005)**

A. 2 B. 3
C. 4 D. 5

Explanation:

The situation given can be expressed with following sample data.

E1
a
b
c

E2
x
y
z

R1

E1	E2
a	x
a	y
b	z

R2

E1	E2
a	x
a	y
b	y

37. Consider a relation scheme R = (A, B, C, D, E, H) on which the following functional dependencies hold: {A→B, BC → D, E → C, D → A}. What are the candidate keys of R? **(GATE 2005)**

A. AE, BE B. AE, BE, DE
C. AEH, BEH, BCH D. AEH, BEH, DEH

Explanation :

A set of attributes S is candidate key of relation R if the closure of S is all attributes of R and there is no subset of S whose closure is all attributes of R.

Closure of AEH, i.e. AEH+ = {ABCDEH}

Closure of BEH, i.e. BEH+ = {ABCDEH}

Closure of DEH, i.e. DEH+ = {ABCDEH}

ANSWER KEY

1. C	**2.** A	**3.** B	**4.** A
5. B	**6.** C	**7.** A	**8.** C
9. B	**10.** A	**11.** B	**12.** A
13. C	**14.** C	**15.** A	**16.** C
17. B	**18.** A	**19.** A	**20.** B
21. A	**22.** B	**23.** A	**24.** D
25. A	**26.** A	**27.** C	**28.** C
29. A	**30.** C	**31.** C	**32.** C
33. C	**34.** D	**35.** C	**36.** C
37. D			

CHAPTER SIX 6

Information System and Software Engineering

6.1 Software Engineering—A Layered Technology

Software engineering encompasses a process, the management of activities, technical methods, and use of tools to develop software products. Fritz Bauer defined Software engineering as the "*establishment and use of sound engineering principles in order to obtain economical software that is reliable and works efficiently on real machines*."

IEEE definition of software engineering: (1) the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).

We need discipline but we also need adaptability and agility.

Software engineering is a layered technology as shown in Fig. 6.1. Any engineering approach must rest on an organizational commitment to quality.The bedrock that supports software engineering is a quality focus.

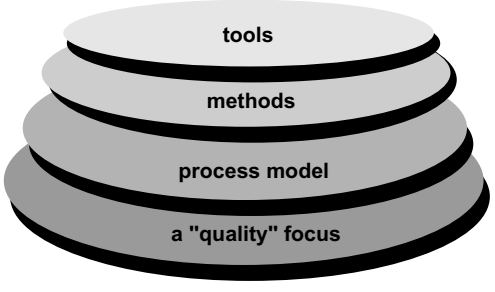


Figure 6.1 Software Engineering: A layered view

The foundation for software engineering is a *process* layer. It is the glue that holds the technology layers together and enables rational and timely development of computer software. Process defines a framework that must be established for effective delivery of software engineering technology.

The software process forms the basis for management control of software projects and establishes the context in which technical methods are applied, work products (models, documents, data, reports, etc.) are produced, milestones are established, quality is ensured, and change is properly managed.

Software engineering *methods* provide the technical "how to's" for building software. Methods encompass a broad array of tasks that include communication, require analysis, design, coding, testing and support.

Software engineering *tools* provide automated or semi-automated support for the process and the methods.

When tools are integrated so that information created by one tool can be used by another, a system for the support of software development called *computer-aided software engineering* is established.

6.1.1 A Process Framework

Software process models can be prescriptive or agile, complex or simple, all-encompassing or targeted, but in every case, five key activities must occur. The framework activities are applicable to all projects and all application domains, and they are a template for every process model.

Software process
Process framework
Umbrella activities
Framework activity #1
Software engineering action

Each framework activity is populated by a set of software engineering actions — a collection of related tasks that produces a major software engineering work product (design is a software engineering action). Each action is populated with individual work tasks that accomplish some part of the work implied by the action.

The following *generic process framework* is applicable to the vast majority of software projects.

Communication: Involves heavy communication with the customer (and other stakeholders) and encompasses requirements gathering.

Planning: Describes the technical tasks to be conducted, the risks that are likely, resources that will be required, the work products to be produced and a work schedule.

Modeling: Encompasses the creation of models that allow the developer and customer to better understand software requirement and the design that will achieve those requirements.

Construction: Combines code generation and the testing required uncovering errors in the code.

Deployment: Deliver the product to the customer who evaluates the delivered product and provides feedback.

Each software engineering action is represented by a number of different task sets – each a collection of software engineering work tasks, related work products, quality assurance points, and project milestones.

The task set that best accommodates the needs of the project and the characteristics of the team is chosen.

The framework described in the generic view of software engineering is complemented by a number of *umbrella activities*. Typical activities include:

Software project tracking and control: Allows the team to assess progress against the project plan and take necessary action to maintain schedule.

Risk management: Assesses the risks that may affect the outcome of the project or the quality.

Software quality assurance: Defines and conducts the activities required to ensure software quality.

Formal technical review: Uncover and remove errors before they propagate to the next action.

Measurement: Defines and collects process, project, and product measures that assist the team in delivering software that meets customers' needs.

Software configuration management: Manages the effect of change throughout the software process.

Reusability management: Defines criteria for work product reuse.

Work product preparation and production: Encompasses the activities required to create work products such as models, documents, etc.

6.2 The Software Engineering Process

6.2.1 The Linear Sequential Model (Waterfall Model)

The linear sequential model is one of the oldest models available and is still thought to be one of the most widely used process models in software development.

The linear sequential model illustrates a sequenced systematic approach to software development, which starts with analysis and progresses through each stage to testing and maintenance (completion). Each stage has a set of defined milestones and results and progress to another stage does not occur until these predefined results are accomplished.

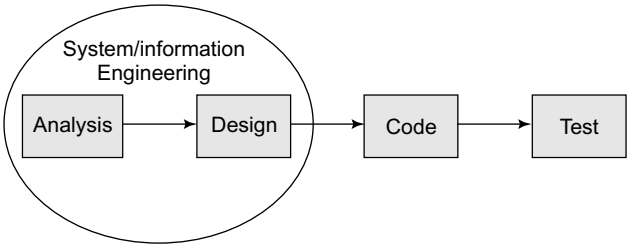


Figure 6.2 The Linear Sequential Model

The first stage, system engineering and modelling involves establishing requirements for all systems elements such as system architecture, hardware, system function and goals, etc. System/Information Engineering and analysis encompasses requirements gathering at the system level and strategic business level with a certain amount of top-level analysis and design. When this stage is complete and software system constraints are identified due to system requirements software requirements analysis begins.

6.2.1.1 Software Requirements Analysis

This activity involves requirements gathering specifically for the software part of the system. The information domain, required function, behaviour, performance, and interfacing need to be fully understood. These requirements are defined during meetings with software developers and customers/users.

6.2.1.2 Software Design

This is a multistage process. The design process translates the requirements defined in last stage into a software representation that can be assessed for quality before coding begins.

6.2.1.3 Testing

Testing begins once code has been generated. Testing uncovers errors, which caused incorrect output or input that do not match the agreed requirements of the system.

6.2.1.4 Maintenance

All software except perhaps embedded software, will go through changes after it has been developed. These changes may arise due to changes in the external environment (e.g., change in hardware infrastructure) or due to errors encountered or changes in customers' requirements (functional or non-functional). The maintenance phase will reapply the above steps to the existing program rather than creating a new one.

6.2.2 The Prototyping Model

This model is often used when customers/users are unable to quantify or describe their requirements accurately for the software system. This is an iterative model with each successive phase becoming more detailed and strictly documented than the first. Therefore, the first iteration is very creative and unconstrained and informal. This model usually begins with requirements gathering. Then a prototype is quickly put together using initial requirements gathering. The quick design focussed on a representation of those aspects of the software that will be viable to customers/users in order to try to improve their requirements. The customers/users then test-drive or evaluate the prototype. Iterations through this loop allow stakeholders to better understand requirements and what task software is to carry out. This model gives all stakeholders a better understanding of software to be built, clarifies and discovers requirements that may not have been discovered until after system was developed.

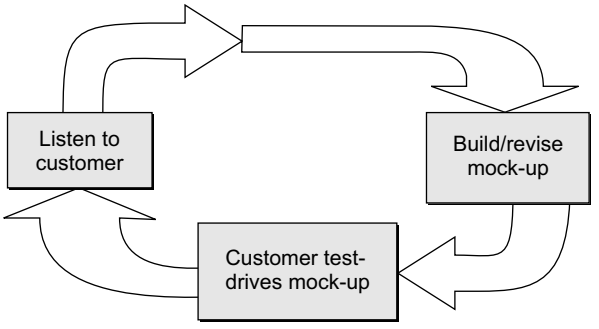


Figure 6.3 The Prototyping Paradigm

6.2.3 Evolutionary Software Process Model

Complex software systems evolve during software development creating new system and software requirements. This makes straight line SDLC impossible to follow. However due to competition and marketing forces, a limited version of the software must be released. This method allows production of increasingly more complete versions of the software product.

6.2.4 The Spiral Model

Boehm developed the spiral model in 1988; it is an iterative prototyping model that uses the systematic and formal

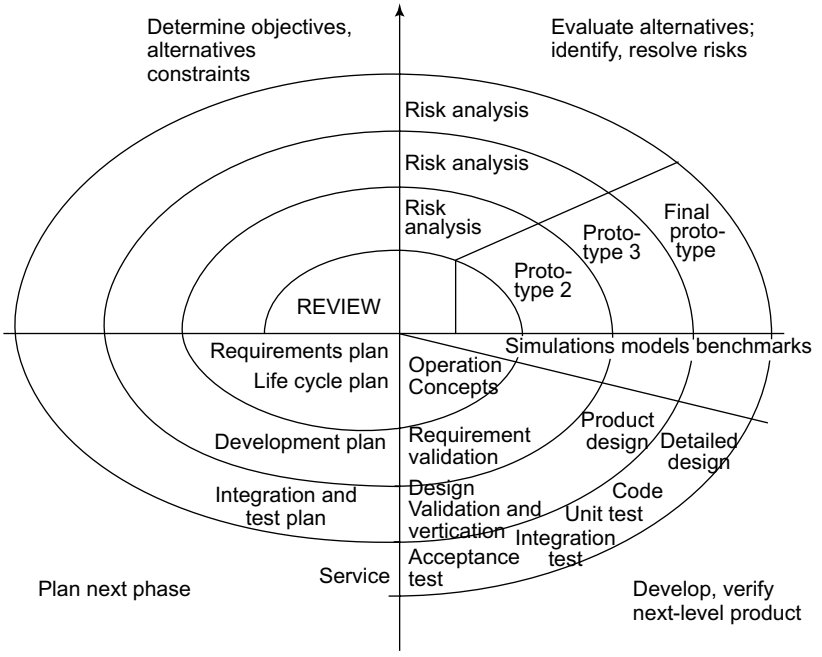


Figure 6.4 Example of a Spiral Development Process [Boehm88]

approaches of the linear model. The model like all the others is broken up into phases from communication, planning, risk analysis, engineering construction and release, and customer evaluation. After each iteration targets deliverables, risk and the number of iterations to be completed are adjusted after discussions between stakeholders. A prototype is usually produced after each iteration starting with paper-based models and working to more complete prototypes. Each of the phases encompasses certain work tasks that are characteristic of the software project. Thus this model can be applied to small and large projects with more complex, comprehensive and numerous tasks can be carried out for each phase. However, it is not the perfect model. It may be difficult to convince customers that the development process is controllable. Also its success relies heavily on the success of the risk analysis expertise used.

6.3 Software Requirement Specification

A system is a collection of elements that are organised to perform a task using some method. They involve elements such as hardware, software, people, procedures and processes. A system analyst defines these elements of system initially.

System Analysis Objectives:

Before preparing SRS (Software Requirement Specification) system should be analysed for following tasks:

(a) To identify customer needs
(b) Feasibility of system
(c) To perform technical and economic analysis
(d) Establish cost

The main aim is to identify customer needs, which lead to success of software system.

Feasibility Study:

This is a study about time and money is enough or not. This involves cost-benefit analysis of the system.

Technical feasibility indicates that what technology to be used for development and tools needed.

Requirements Engineering:

This phase believes – "*A problem well specified is half-solved*".

Requirement engineering is a disciplined application of proven, principals, methods, tools and notations to describe a proposed system's intended behavior and its associated constraints.

This includes following activities:

(1) Identification and documentation of user needs.
(2) Creation of a document describing external behavior and its associated constraints.
(3) Analysis and validation of document to ensure completeness.

The primary output of this stage is requirement specification, which describes both hardware and software. If it describes only software then it is "Software requirement Specification". The document must be understandable by: users, customers, designers and testers. The document includes.

(1) Inputs and Outputs
(2) Functional requirements
(3) Non-functional requirements-performance

Reasons for Poor-Requirement Engineering:

(1) Requirements will change
(2) Difficult to cover
(3) Communication barrier between developers and users
(4) Lack of confidence of developers
(5) Use of in-appropriate methods
(6) Insufficient training

Software Requirement Specification (SRS)

SRS is a means of translating ideas in the minds of clients into a formally specified set of requirements. SRS document is a communication medium between customer and the supplier. The document is initially not to be edited.

This phase includes following activities;

(a) Problem/Requirement Specification
(b) Requirement Specification.

The first step is to understand the problem, goals, and constraints.

Second step is the specification of needs that is identified in first step. This phase terminates with validated requirement specification document.

Why SRS is required?

This may be used in competitive tendering of the company or writing to their own.

Used to capture user requirements and highlight any inconsistencies, conflicting requirements.

The client does not know about software or software development and the developers do not understand client's problem and application area. Hence there is a communication gap between client and developers. Thus, SRS act as a bridge between this communication gaps.

A good SRS should satisfy all parties. SRS also helps clients to understand their needs.

What is contained in SRS? / Components of SRS

The SRS document should contain a list of requirements that has to be agreed by both client and developers.

(1) Functional requirements
(2) Performance requirements
(3) Interface requirements
(4) Operational requirements
(5) Resource requirements
(6) Verification requirements
(7) Acceptance testing requirements

(8) Documentation requirements
(9) Quality requirements
(10) Safety requirements
(11) Reliability and Maintainability requirements

(1) Functional Requirements
This is a subset of overall system requirements. This consider trade-offs (hardware and software) and also describes how the system operates under normal conditions and response to software failures OR invalid inputs to system.

(2) Performance Requirements
This can be stated in measurable value i.e., rate, frequency, speeds and levels. This can be extracted from system specifications.

(3) Interface Requirements
This can be specified separately for hardware and software. These requirements specify any interface standard that is requested. These should be carefully documented.

(4) Operational Requirements
This gives the "in the field" view of the system. The specified details are,

(a) How system will operate and communicate?
(b) What are the operator syntax/notations?
(c) How many operators and their qualification required?
(d) What help is provided by system.
(e) How error messages should be displayed?
(f) What is screen layout look?

(5) Resource Requirements
Specify utilisation of hardware, such as amount, percentage and memory usage. This is very important when extending hardware. The software resources include using specific, certified, standard compilers and databases.

(6) Verification Requirements
This specifies how customer accepts after completion of project. This specifies how functional and performance requirements to be measured. This also states, whether tests are to be staged or only under completion of the project and also whether a representative of client's company should present or not.

(7) Acceptance Testing
This provides details of tests to be performed for customer acceptance in document.

(8) Documentation Requirements
This specifies what documents are to be supplied to client, either through the project OR at the end of the project. The document and other relevant documentation

(9) Quality Requirements
This specifies whether product should meet international or local standards. The quality factors are: Correctness, reliability, efficiency, integrity, usability, maintainability, flexibility, portability and reusability

(10) Safety Requirements
This specifies safety steps to be taken for protection of human, equipments and data. i.e., protecting from moving parts, electrical circuitry and other physical dangers

(11) Reusability Requirements
This states that software must perform function under stated conditions, for a given period of time.

(12) Maintainability Requirements
This is maintenance of software in the site it is used, for hardware and software changes in the system.

Characteristics of Good SRS
The SRS must be clear, concise, consistent, traceable and unambiguous.
Complete: The SRS should include all types of requirements to be specified.
Consistent: There should not be any conflict, there may be following confliction.

- **Multiple descriptors:** Two or more words referring to the same object.
- **Opposing physical requirements:** Description of real world objects clash, i.e., one requirement states warning indicates orange and another states red.
- **Opposing functional requirements:** This is a conflict in functions.

Traceable: Tracing the references, which help in modifications have made to requirement to bring out to its current state. This is an aid in modification in future documents by stating references.

Unambiguous: This means "Not having two or more possible meanings". This means each requirement can have only one interpretation. One way of removing the ambiguity is to use requirement specification language. This is beneficial when detecting ambiguity using lexical syntactic analysers. The disadvantage is time needed for learning and understanding of the system to be built.

Verifiable: The software requirement specification must be verifiable, that it contains all of the requirements of the user. Verifiable requirements are that the software product must use a cost effective method. Non-verifiable requirements are system should have good-interface and work well under most conditions.

How requirement are specified?
The two factors needed for requirement specification are:

1. Notation used to describe requirement
2. How the notations is to be presented to the reader of SRS document

Notations
These are the terms used for specifying the requirements. This is used since the customer does not understand the technical language. Hence, the requirement expert must know the ability of knowledge of customer to understand

modeling. Modeling techniques include Z schema, DFD, ER diagram, state transition diagram, and flow charts.

Benefits of Good SRS

1. Establish basis for agreement between client and supplier.
2. Reduces development cost
3. Helps in removal of inconsistencies, omissions and misunderstandings.
4. Helps in validation of final product.

6.4 Software Cost Estimation

Estimation of cost of the software products is an important task. There are many factors which influence on cost of a software products development and maintenance.

The primary cost factors are:

(a) Programmer ability and familiarity of application area
(b) Complexity of product
(c) Size of the product and available time
(d) Required level of reliability
(e) Level of technology used
(f) Familiarity and availability of technology

6.4.1 Cocomo Model

Constructive Cost Estimation model was proposed by Boehm. He postulates three classes of products:

(1) Organic products – Application programs (Data processing and scientific)
(2) Semi- detached – Utility programs (Compilers, Linkers)
(3) Embedded – System programs

Boehm introduces three forms of COCOMO:

(1) Basic COCOMO

Computes development effort & cost, given program size in estimated lines of code.

(2) Intermediate COCOMO

Computes effort as function of program size and various cost drivers that relates to product, hardware, personnel, and project attributes.

(3) Advanced COCOMO

This model incorporates all intermediate model characteristics with an assessment of cost driver's impact on each step of software engineering process.

COCOMO can be applied to 3 modes of the projects

(a) Organic mode: These are simple, small projects developed by small teams with good experience.

(b) Semi-detached: These are medium sized projects (in size and complexity) developed by the team with mixed experience. This project doesn't meet the rigid requirements of the project.

(c) Embedded project: These projects should be developed within set of tight constraints.

Example: Flight control software.

Basic COCOMO Equation

$E = a_b \times (KLOC)\ exp(b_b)$ [Effort of person – months]

$D = c_b \times (E)\ exp(d_b)$ [development time in months]

Where

E – Effort applied in person – months
D – Development time
KLOC – estimated no of lines of code
a_b, c_b – co-efficients
b_b, d_b – components are in table.

From E and D, we can compute no of peoples required for project N, by equation, N = N / D [People]

Software project	a_b	b_b	C_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.8	1.20	2.5	0.32

Intermediate COCOMO

This is an extended model with a set of cost drivers. The equation is

$E = a_i(LOC)\ exp(b_i) \times EAF$

EAF – effort adjustment factor (range 0.9 – 1.4)
E- Effort applied in person months
a_i, b_i – co-efficients is shown in table below.

Software project	a_i	b_i
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Table of cost drivers

Cost Driver	Description
RELY	
DATA	Required software reliability
CPLX	Database size
TIME	Product complexity
STOR	Execution time constraints
VIRT	Virtual machine volatility – degree to which the computer operating system change
TURN	Computer turn around time
ACAP	Analyst capability
AEXP	Application experience
PCAP	Programmer capability
VEXP	Virtual machine (i.e operating system) experience
LEXP	Programming language experience
MODP	Use of modern programming practices
TOOL	Use of software tools
SCED	Required development schedule.

Staffing level estimation

Modern studied different staffing patterns of different R and D projects and uses Raleigh curve for staffing estimation.

This is the simplest way to determine numbers of software engineers is to divide effort estimation. This is an important since all phases of development doesn't require constant number of engineers. If a constant number of engineers used may cause some phases will be over staffed and some under staffed.

He derived an equation

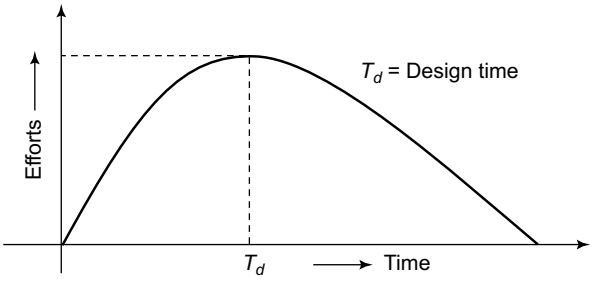
$$E = K/t_d^2 \times t \times \exp[-t^2/2t_d^2]$$

Where E- effort required at time t. This is also an indication of number of engineers at particular time.

K – Area under curve.

td – Time at which curve attains max. value.

$$E = K/t_d^2 \times t \times \exp[-t^2/2t_d^2]$$



6.5 Software Design

This is the step of moving from problem domain to solution domain by taking input as SRS document and output of this stage is architecture design of system to be build.

Design objectives: The major goal of this stage is to provide a best possible design, within limitations by requirements, by social and physical environment constraints. The criteria used to evaluate design are

1. Verifiability
   (a) Completeness
   (b) Consistency
   (c) Efficiency
2. Traceability
   (a) Simplicity / understandability

Software design: Design is a blueprint or a plan of the solution system, representing components, subsystems and their interactions.

Major design activities are:

1. **Architectural design:** Identifying subsystems and their relationships which makeup the system and documented.
2. **Abstract specification**: Each subsystem must be abstract and its services to be provided.
3. **Component design**: Designing services and interfaces of different components.
4. **Data structure design**: Designing detail specification of data structure to be used in system.
5. **Algorithm design**: Designing algorithm used to service to be provided in the system.

6.5.1 Important Points Related Module Level Concepts

A module is a logically separable part of program which is discrete unit. In a programming language, a module is a function or a procedure. A module should support well defined abstractions, solvable and modifiable separately.

The criteria used for modularisation are:

a. Coupling

b. Cohesion

Coupling: Coupling is the strength of interconnections between modules or a measure of interdependence among modules.

If two modules are closely connected means they are "highly coupled". "Loosely coupled" modules are having weak interconnections, where modules have no inter connections are having "no coupling".

Coupling increases as the complexity of interface between modules increases.

Coupling is an abstract concept and it cannot be measured or formulated. Coupling can be minimised by reducing number of interface per module and complexity of interface.

Two components can be dependent on

(a) Reference made between components i.e. module A invokes module B, that means B depends on A.

(b) Amount of data passed between modules, just like passing a block (array) of data from module A to module B.

(c) Amount of control that a component has on another.

(d) The degree of complexity in interface between components.

In practical, it is impossible to build components with no coupling. But if coupling is high, then other components are affected at the time of modification of a component. Hence, a low coupling helps to minimise number of components needing revision.

The types of coupling are:

(1) **No coupling**: The modules are not having interconnections.

(2) **Data coupling**: In this coupling, only data will be passed from one module to another.

(3) **Stamp coupling**: In this, data structure will be passed from one component to another.

(4) **Control coupling**: In this a component passed a parameter to control the activity of another.

(5) **Common coupling:** In this many modules access from a common data store (global data). In this, it is difficult to determine which component is responsible for change in data.

(6) **Content coupling:** One component is completely dependent on another. That is a component uses data or control information maintained within another module.

Components in object-oriented design have low coupling since each object contains functions which defines actions performed by it or on it. Thus, low coupling is an advantage of object-oriented design.

Cohesion

Cohesion of a module represents how tightly bound the internal elements of the module are to one another. This is an intra-module concept.

There are different levels of cohesion, they are

(1) **Co-incidental Cohesion**: This is a condition, where there is no meaningful relationship among the elements of the module.

(2) **Logical Cohesion:** In this, there is some logical relationship between elements of the module.

(3) **Temporal Cohesion:** The elements in a module are executed together. These exit in modules performing "initialisation", "clean-up", and "termination".

(4) **Procedural Cohesion:** A procedurally cohesive module contains elements that belongs to same procedural limit.

(5) **Communicational Cohesion:** In this level, module has elements that are related by a reference to the same input or output data. This level module is performing more than one function.

(6) **Sequential Cohesion:** The elements are together in one module, where output of one forms an input to another.

(7) **Functional Cohesion:** This is the strongest cohesion, in which all elements of a module are related for performing a single function. Functions like compute square are functionally cohesive.

All models in a design must be good when it functionally cohesive and low coupling, the number of modules obtained by the portioning must be minimised. Because large number complexity and lead to a bad design

6.5.2 Design Notations

The representations schemes used in design are called as Design notations. Notations used to specify the external characteristics structure and processing details of the system, the design notations used are

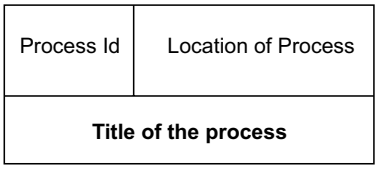
1. Pseudo code
2. Structured English
3. Data flow diagram
4. Structural charts

6.5.2.1 Data Flow Diagrams

DFD is a tool used by system analysts to show the flow of data in an information system. The different symbols or components for representation are:

6.5.2.1.1 *Process*

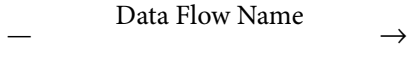
Process transforms a data flow by either changing its structure or by generating new information from it.



A process must have at least one data flow into it and one data flow out of it. A process in DFD having contains a process id, location of process and process title.

6.5.2.1.2 *Data flows*

A data flows can be represented by an arrow depicts data flowing form one process to another. The arrow head shows directions of flow and with a label as identification.



6.5.2.1.3 *Data stores*

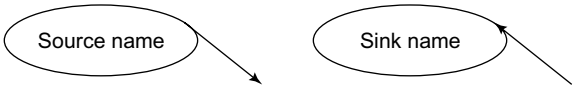
A data store is a computer file, a manual record or a pile of documents. It is a collection of related information a telephone book, patient records student records.

ID	Data Store Name

Each data store is having a unique id that is a letter followed by a number and data store name.

6.5.2.1.4 *Sources and sinks (terminations or external entities)*

External entities are which provide or reactive information form to the system.



A sink is a one which receives information from the information to the system. An external entities are people, place, a manager, a sales department

6.5.2.1.5 *DFD leveling*

DFD's allows analyst or user to look a system at different levels of details. DFD leveling is a practice where a DFD depend on its details into set of DFD's. The DFD depend on its detail representation called as level 2 diagrams. If necessary it is possible to design level 3 and level 4, etc. for more detailed representation.

Context diagrams (level 0 DFD):

A level O DFD is known as context diagram representing high level details of the system. Context diagram comprises a process box for entire system, with external entities and data flows between them.

A current physical DFD is a DFD showing data and operation within current existing system.

Guidelines to draw context diagram

a. Read case study from start to end, until you get a fair idea about system and its processes.
b. Make a list of external entities.
c. Identify data flows from system and to the s/m.

Level 1 DFD

More detailed representation is shown in level 1 DFD. Level 1 DFD contains data flows, data stores, process and external entities. The data flows are connected to and from the actual process which create reactive or change them. Processes are identified by number as 1,2,3,4 and so on.

Guidelines to draw level 1 DFD:

a. Make a sentence of function in system and identify verb as a process in the system
b. Make a list of all potential process
c. Group potential process to form 3 to 10 process. A DFD must contain minimum of 3 and maximum of 10 process
d. Identify data flows
e. Identify list of data stores

6.5.2.1.6 *Advantages of DFD*

1. Easy to understand and validate correctness
2. Since DFD is a pictorial representation, it can be understood quickly than textural narration
3. DFD shows an abstract specification of system. It only shows what system will do? Rather than how it can be done?

6.5.2.2 Structured Charts

This is used in functional oriented design. The structure of a program is made up of modules and their interconnections.

A structured chart is a graphical representation of structure of a problem. In this module is represented by a box, with its name. The flow of data parameters are represented by arrows. The parameters can be shown as data by unfilled circle of the arrow (Fig. 6.5).

The invoking (calling) function is called "superiordinates" and function is called "subordinate"

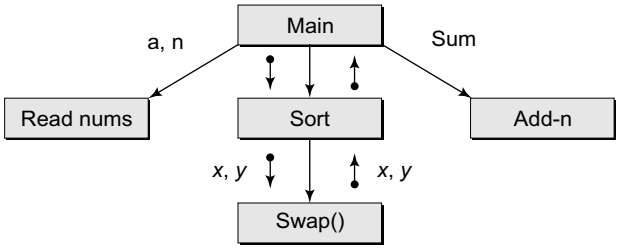


Figure 6.5 Structured Charts showing function calls

There may be situations, where designer wish to communicate procedural information explicitly like loops and decisions, there are represented as shown in Fig. 6.6(a) and (b).

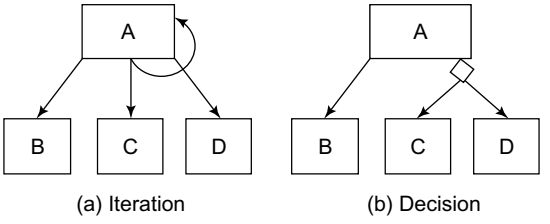


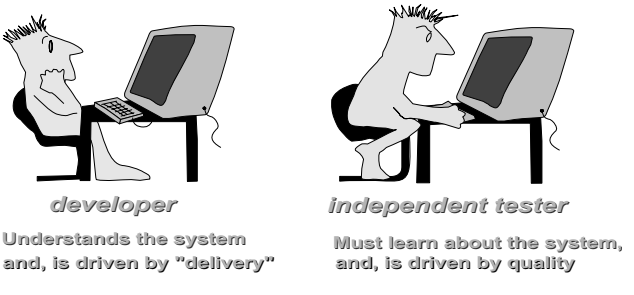
Figure 6.6 Representation of loops and decision making in structured charts

6.6 Software Testing Fundamentals

Testing is the process of exercising a program with the specific intent of finding errors prior to delivery to the end user.

6.6.1 Software Testing Strategies

Who Tests the Software?



All software testing strategies provide the software developer with a template for testing and all have the following generic characteristics:

- Conduct effective formal technique reviews, by doing this, many errors will be eliminated before testing commences.

- Testing begins at the component level and works "outward" towards the integration of the entire computer-based system.
- Different testing techniques are appropriate at different points in time.
- Testing is conducted by the developer of the software and an independent test group.
- Testing and debugging are different activities, but debugging must be accommodated by testing strategy.

6.6.1.1 Verification and Validation

Verification refers to the set of activities that ensure that software correctly implements a specific function.
Validation refers to the set of activities that ensure that the software has been built is traceable to customer requirements.

Verification: Are we building the product right?

Validation: Are we building the right product?

The definition of Verification and Validation encompasses many of the activities that are encompassed by Software Quality Assurance (SQA).

Testing does provide the last fortress from which quality can be assessed and more pragmatically, errors can be uncovered.

Testing should not be viewed as a safety net that will catch all errors that occurred because of weak software engineering practices. Stress quality and error detection throughout the software process.

6.6.1.2 Organising for Software Testing

For every software project, there is an inherent conflict of interest that occurs as testing begins. Programmers that built the software are asked to test it. Unfortunately, these developers have a vested interest in demonstrating that the program is error free and work perfectly according to the customer's requirement.

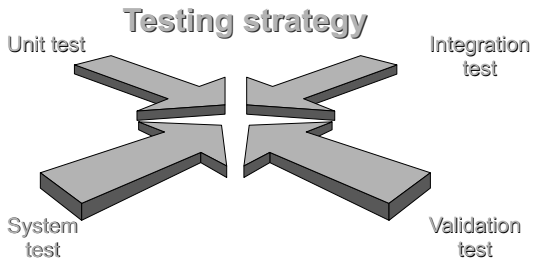
An independent test group does not have the conflict that builders of the software might experience.

There are often a number of misconceptions that can be erroneously inferred from the preceding discussion:

1. That the developer of software should not test.
2. That the software should be tossed over the wall to strangers who will test it mercilessly.
3. That testers get involved only when testing steps are about to begin.

These aforementioned statements are incorrect.

The role of an *Independent Test Group* (ITG) is to remove inherent problems associated with letting the builder test the software that has been built. The ITG and software engineering. Work closely throughout a software project to ensure that thorough tests are conducted.



6.6.1.3 Strategic Issues

Testing Strategy

- We begin by 'testing-in-the-small' and move toward 'testing-in-the-large'
- For conventional software
  - The module (component) is our initial focus
  - Integration of modules follows
- For OO software
  - Our focus when "testing in the small" changes from an individual module (the conventional view) to an OO class that encompasses attributes and operations and implies communication and collaboration.

Specify product requirement in a quantifiable manner long before testing commences. "Portability, maintainability, and usability."

State testing objectives explicitly. "Test effectiveness, test coverage, mean time to failure, etc."

Build use-cases. "*Understand the users of the software and develop a profile for each user category*".

Rapid cycle testing. "*Develop a testing plan that emphasizes*". Feedback generated from rapid-cycle tests can be used to control quality levels and the corresponding test strategies.

Build "robust" software that is designed to test itself.

Use effective formal technical reviews as a filter prior to testing.

Conduct formal technical reviews to assess the test strategy and test cases themselves.

Develop a continuous improvement approach for the testing process. The test strategy should be measured by using metrics.

6.6.1.4 Test Strategies for Traditional Software

6.6.1.4.1 *Unit Testing*

Both black-box and white-box testing techniques have roles in testing individual software modules.

Unit Testing focuses verification effort on the smallest unit of software design.

Unit Test Considerations

Module interface is tested to ensure that information properly flows into and out of the program unit under test.

Local data structures are examined to ensure that data stored temporarily maintains its integrity.

All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once.

All error handling paths are tested.

If data do not enter and exit properly, all other tests are moot.

Comparison and control flow are closely coupled. Test cases should uncover errors such as:

1. comparison of different data types
2. incorrect logical operators or precedence
3. expectation of equality when precision error makes equality unlikely
4. incorrect comparison of variables
5. improper loop termination
6. failure to exit when divergent iterations is encountered
7. improperly modified loop variables

Boundary testing is essential. software often fails at its boundaries. Test cases that exercise data structure, control flow, and data values just below, at, and just above maxima and minima are very likely to uncover errors.

Error handling: When error handling is evaluated, potential errors should be tested:

1. error description is unintelligible
2. error noted does not correspond to error encountered
3. error condition causes O/S intervention prior to error handling
4. exception-condition processing is incorrect
5. error description does not provide enough information to assist the location of the cause of the error.

Unit Test Procedures

Because a component is not a stand-alone program, driver and/or stub software must be developed for each unit test.

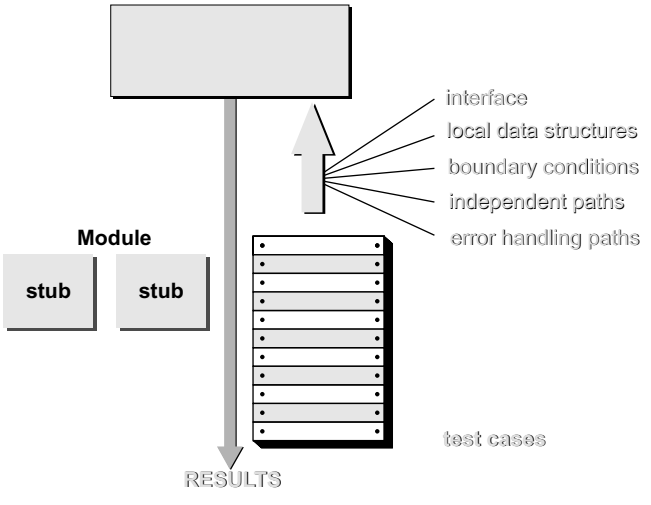


Figure 6.7 Unit Testing

In most applications, a *driver* is nothing more than a "main program" that accepts test case data, passes such data to the component, and prints relevant results.

Stubs serve to replace modules that are subordinate to the component to be tested. A stub "dummy program" uses the subordinate module's interface, may do minimal data manipulation, provides verification of entry, and returns control to the module undergoing testing.

6.6.1.4.2 *Integration Testing Strategies*

Integration testing often forms the heart of the test specification document. Do not be dogmatic about a "pure" top down or bottom up strategy. Rather, emphasise the need for an approach that is tied to a series of tests that (hopefully) uncover module interfacing problems.

Options:

- The "big bang" approach: all components are combined in advance; the entire program is tested as a whole.
- An incremental Integration: is the antithesis of the big bang approach. The program is constructed and tested in small increments, where errors are easier to isolate and correct.

Integration Testing is a systematic technique for constructing the software architecture while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design.

Top-down Integration

Top-down Integration testing is an incremental approach to construction of the software arch.

Modules are integrated by moving downward through the control hierarchy, beginning with the main control module (main program).

Modules subordinate to the main control module are incorporated into the structure in either depth-first or breadth-first manner.

Depth-first integration integrates all components on a major control path of the program structure. Selection of a major path is somewhat arbitrary and depends on application-specific characteristics.

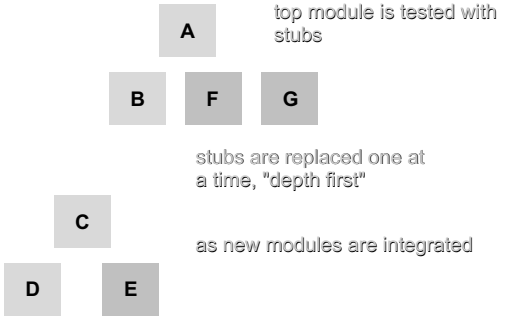
Breadth-first integration incorporates all components directly subordinate at each level, moving across the structure horizontally.

The integration process is performed in a series of 5 steps:

1. The main control module is used as a test driver, and stubs are substituted for all components directly subordinate to the main control module.
2. Depending on the integration approach selected subordinate stubs are replaced one at a time with actual components.

3. Tests are conducted as each component is integrated.
4. On completion of each set of tests, another stub is replaced with the real component.
5. Regression testing may be conducted to ensure that new errors have not been introduced.

The process continues from step 2 until the entire program structure is built.



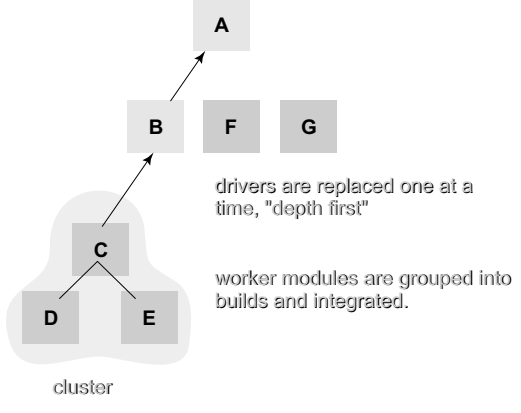
Top-down strategy sounds relatively uncomplicated, but, in practice, logistical problems can arise.

Bottom-down Integration

Bottom-down Integration testing begins construction testing with atomic modules.

The *Bottom-down Integration* strategy may be implemented with the following steps:

1. Low-level components are combined into *clusters* that perform a specific software sub-function.
2. A driver is written to coordinate test case input and output.
3. The cluster is tested.
4. Drivers are removed and clusters are combined moving upward in the program structure.



6.6.1.4.3 *Regression testing*

Regression testing is the re-execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects.

Regression testing is the activity that helps to ensure that changes do not introduce unintended behaviour or additional errors.

The regression test suite contains three different classes of test cases:

1. A representative sample of tests that will exercise all software functions.
2. Additional tests that focus on software functions that are likely to be affected by the change.
3. Test that focus on the software system components that have been changed.

6.6.1.4.4 *Smoke Testing*

Smoke Testing: It is an integration testing approach that is commonly used when software products are being developed.

A common approach for creating "daily builds" for product software smoke testing steps:

1. Software components that have been translated into code are integrated into a "build." A build includes all data files, libraries, reusable modules, and engineered components that are required to implement one or more product functions.
2. A series of tests is designed to expose errors that will keep the build from properly performing its function. The intent should be to uncover "show stopper" errors that have the highest likelihood of throwing the software project behind schedule.
3. The build is integrated with other builds and the entire product (in its current form) is smoke tested daily. The integration approach may be top down or bottom up.

Smoke testing provides a number of benefits when it is applied on complex, time critical software projects.

- *Integration risk is minimised*: because smoke tests are conducted daily, incompatibilities and other errors are uncovered early.
- *The quality of the end*-product *is improved*: because the approach is construction oriented, smoke testing is likely to uncover functional errors and architectural and component-level design errors.
- *Error diagnosis and correction are simplified*: errors uncovered during smoke testing are likely associated with "new software increments".
- *Progress is easier to access:* with each passing day, more of the software has been integrated and more has been demonstrated to work.

6.6.1.4.5 *Test Strategies for Object-Oriented Software*

This section clarifies the differences between OOT and conventional testing with regard to unit testing and integration testing. The key point to unit testing in an OO context is that the lowest testable unit should be the encapsulated class or object (not isolated operations) and all test cases should be written with this goal in mind.

Given the absence of a hierarchical control structure in OO systems integration testing of adding operators to classes is not appropriate.

Unit Testing in the OO Context

An encapsulated class is the focus of unit testing; however, operations within the class and the state behaviour of the class are the smallest testable units.

Class testing for OO software is analogous to module testing for conventional software. It is not advisable to test operations in isolation.

Integration Testing in the OO Context

An important strategy for integration testing of OO software is thread-based testing. Threads are sets of classes that respond to an input or event. Use-based tests focus on classes that do not collaborate heavily with other classes.

Thread-based testing integrates the set of classes required to respond to one input or event for the system. Each thread is integrated and tested individually.

Use-based testing begins the construction of the system by testing those classes (called independent classes) that use very few server (if any) classes.

Next, the dependent classes, which use independent classes, are tested.

This sequence of testing layers of dependent classes continues until the entire system is constructed.

Cluster testing is one-step in the integration testing of OO software. A cluster of collaborating classes is exercised by designing test cases that attempt to uncover errors in the collaborations.

6.6.1.5 Validation Testing

Validation testing is described as the last chance to catch program errors before delivery to the customer. If the users are not happy with what they see, the developers often do not get paid. The key point to emphasise is *traceability* to requirements. In addition, the importance of alpha and beta testing (in product environments) should be stressed.

High Order Testing

Validation Test Criteria:

Focus is on software requirements. A test plan outlines the classes of tests to be conducted, and a test procedure defines specific test cases. Both the plan and procedure are designed to ensure that all functional requirements are satisfied, all behavioral characteristics are achieved, and all performance requirements are attained, documentation is correct, and usability and other requirements are met.

Configuration Review:

It is important to ensure that the elements of the software configuration have been properly developed.

Alpha/Beta testing:

The focus is on customer usage.

The *alpha-test* is conducted at the developer's site by end-users. The software is used in natural setting with the developer "looking over the shoulder" of typical users and recording errors and usage problems. Alpha tests are conducted in a controlled environment.

The *beta-test* is conducted at the end-users sites. The developer is generally not present. Beta test is a live application of the software in an environment that cannot be controlled by the developer. The end-user records errors and all usage problems encountered during the test and the list is reported to the developer. Then software engineers make modifications and then prepare for release of software product to the entire customer base.

6.6.1.6 System Testing

The focus is on system integration. "Like death and taxes, testing is both unpleasant and inevitable."

System Testing is a series of different tests whose primary purpose it to fully exercise the computer-based system. The following are the types of system tests:

Recovery Testing

Forces the software to fail in a variety of ways and verifies that recovery is properly performed. "Data recovery"

Security Testing

It verifies that protection mechanisms built into a system will, in fact, protect it from improper penetration.

Beizer "The system's security must, of course, be tested for invulnerability from frontal attack-but must also be tested for invulnerability from flank or rear attack."

Stress Testing

It executes a system in a manner that demands resources in abnormal quantity, frequency, or volume.

For example:

(1) Special tests may be designed to generate 10 interrupts per second, when one or two is the average rate,
(2) Input data rates may be increased by an order of magnitude to determine how input functions will respond,
(3) Test cases that require maximum memory or other resources are executed,
(4) Test cases that may cause memory management problems are designed,

(5) Test cases that may cause excessive hunting for disk-resident data are created.

A variation of stress testing is a technique called *sensitivity testing*. They attempt to uncover data combinations within valid input classes that may cause instability or improper processing.

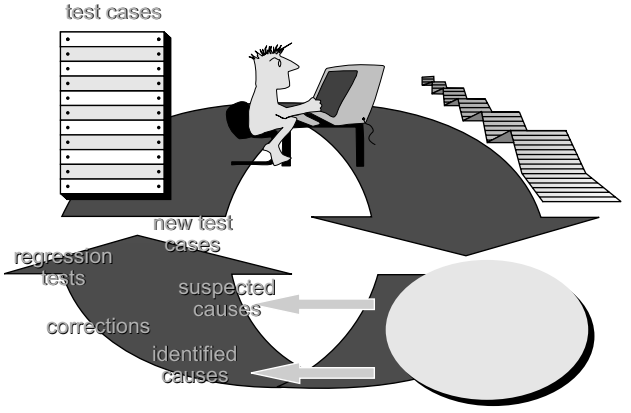
Performance Testing

It tests the run-time performance of software within the context of an integrated system.

Performance tests are coupled with stress testing and usually require both hardware and software instrumentation. "Processing Cycle, log events".

6.6.1.7 The Art of Debugging

The Debugging Process



Debugging occurs as a consequence of successful testing. That is, when a test case uncovers an error, debugging is an action that results in the removal of the error.

Debugging is not testing but occurs as a consequence of testing. Debugging process begins with the execution of a test case.

Results are assessed and a lack of correspondence between expected and actual performance is encountered. In many cases, the non-corresponding data are a symptom of an underlying cause as yet hidden. Debugging attempts to match symptom with cause, thereby leading to error correction.

Debugging will always have one of two outcomes:

(1) The cause will be found and corrected, or

(2) The cause will not be found.

Why is debugging so difficult?

1. The symptom and the cause may be geographically remote ➔ highly coupled components.
2. The symptom may disappear temporarily when another error is corrected.
3. The symptom may actually be caused by non-errors (round-off inaccuracies).
4. The symptom may be caused by human error that is not easily traced.
5. The symptom may be a result of timing problem, rather than processing problems.
6. It may be difficult to accurately reproduce input conditions (a real-time application on which input ordering is indeterminate).
7. The symptom may be intermittent. That is particularly common in embedded systems that couple hardware and software inextricably.
8. The symptom may be due to causes that are distributed across a number of tasks running on different processors.

Psychological Considerations

It appears that debugging process is an innate human trait. Although it may be difficult to learn how to debug, a number of approaches to the problem can be proposed.

Three debugging strategies have been proposed:

1. Brute force
2. Backtracking
3. Cause Elimination

Each of these strategies can be conducted manually, but modern tools can make the process much more effective.

Brute force is probably the most common and least efficient method for isolating the cause of a software error. Using a "let the computer find the error", memory dumps, run-time traces, and loading the program with output statements.

Although the mass of information may ultimately lead to success, it more frequently leads to wasted effort and time.

Backtracking: Beginning at the site where a symptom has been uncovered, the source code is traced backwardly until the site of the cause is found. The larger the program, the harder is to find the problem.

Cause elimination: It is maintained by induction or deduction and introduces the concept of *binary partitioning*. Data related to the error occurrence are organised to isolate potential causes.

A "cause hypothesis" is devised, and the aforementioned data are used to prove or disprove the hypothesis. Alternatively, a list of all possible causes is developed, and tests are conducted to eliminate each.

Its initial tests indicate that a particular cause hypothesis shows promise; data are refined in an attempt to isolate the bug.

Formal reviews by themselves cannot locate all software errors. Testing occurs late in the software development process and is the last chance to catch bugs prior to customer release.

6.6.2 Testing Procedures

Testability: is simply how easily a computer program can be tested.

The following characteristics lead to testable software:

Operability: It operates cleanly if implemented with quality in mind.

Observability: The results of each test case are readily observed. Variables are visible during execution. Source code is available.

Controllability: The degree to which testing can be automated and optimised

Decomposability: Testing can be targeted; independent modules can be tested independently.

Simplicity: Reduce complex architecture and logic to simplify tests; code simplicity ➔ coding standards.

Stability: Few changes are requested during testing, the software recovers well from failures.

Understandability: Changes to the design are communicated to the testers.

6.6.2.1 Test Characteristics

1. A good test has a high probability of finding an error. Tester must understand the system and develop a mental picture of how it might fail.
2. A good test is not redundant; every test should have a different purpose.
3. A good test should be "best of breed"; the test that has the highest likelihood of uncovering a whole class of errors should be used.
4. A good test should be neither too simple nor too complex.

6.6.2.2 Black-Box and White-Box Testing

This section discusses the differences between black-box and white-box testing.

"Bugs lurk in corners and congregate at boundaries ..."

OBJECTIVE	to uncover errors
CRITERIA	in a complete manner
CONSTRAINT	with a minimum of effort and time

Black-Box testing alludes to tests that are conducted at the software interface. It examines some fundamental aspects of a system with little regard for the internal logical structure of the software.

White-Box testing is predicated on close examination of procedural detail. Logic paths through the software and collaborations between components are tested by providing test cases that exercise sets of conditions and/or loops.

6.6.2.3 White-Box Testing

This section makes the case that white-box testing is important, since there are many program defects (e.g. logic errors) that black-box testing can not uncover.

Sometimes called *glass-box testing*, is a test case design philosophy that uses the control structure described as part of component-level design to derive test cases.

The S/W engineering can derive test cases that:

1. Guarantee that all independent paths within a module have been exercised at least once,
2. Exercise all logical decisions on their true and false sides,
3. execute all loops at their boundaries and within their operational bounds,
4. Exercise internal data structures to ensure their validity.

6.6.2.4 Basis Path Testing

This section describes basis path testing as an example of a white-box testing technique.

Basis Path Testing is a white-box testing technique which enables the test case designer to derive logical complexity measure of procedural design and use this measure as a guide for defining a set of execution paths.

Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least once time during testing.

6.6.2.4.1 *Flow Graph Notation*

A flowchart is used to depict program control structure. Each circle, called a *flow graph node*, represents one or more procedural statements. A sequence of boxes and decision diamond can map into a single node. The arrows on the flow graph, called *edges* or *links*, represent flow of control and are analogous to flowchart arrows.

An edge must terminate at a node, even if the node does not represent any procedural statements.

Areas bounded by edges and nodes are called *regions*. When counting regions, we include the area outside the graph as a region.

6.6.2.5 Control Structure Testing

Basis path testing is one form of control structure testing. This section introduces three others (condition testing, data flow testing, loop testing). The argument given for using these techniques is that they broaden the test coverage from that which is possible using basis path testing alone.

6.6.2.6 Black-Box Testing

The purpose of black-box testing is to devise a set of data inputs that fully exercise all functional requirements for a program. It is important to know that in black-box testing the test designer has no knowledge of algorithm implementation. The test cases are designed from the requirement

statements directly, supplemented by the test designer's knowledge of defects that are likely to be present in modules of the type being tested.

It is also called *behavioural testing*, which focuses on the functional requirements of the software.

Black-box testing attempts to find errors in the following categories:

1. incorrect or missing functions
2. interface errors
3. errors in data structures or external database access
4. behaviour or performance errors
5. initialisation and termination errors

Black-box testing tends to be applied during the later stage of testing. Tests are designed to answer the following questions:

- How is functional validity tested?
- How is system behaviour and performance tested?
- What classes of input will make good test cases?
- Is the system particularly sensitive to certain input values?
- How are the boundaries of a data class isolated?
- What data rates and data volume can the system tolerate?
- What effect will specific combinations of data have on system operation?

OOT—Test Case Design

Berard [BER93] proposes the following approach:

1. Each test case should be uniquely identified and should be explicitly associated with the class to be tested
2. The purpose of the test should be stated
3. A list of testing steps should be developed for each test and should contain [BER94]:
   a. a list of specified states for the object that is to be tested.
   b. a list of messages and operations that will be exercised as a consequence of the test.
   c. a list of exceptions that may occur as the object is tested.
   d. a list of external conditions (i.e., changes in the environment external to the software that must exist in order to properly conduct the test).
   e. supplementary information that will aid in understanding or implementing the test.

Testing Methods

Fault-based testing

- The tester looks for plausible faults (i.e., aspects of the implementation of the system that may result in defects). To determine whether these faults exist, test cases are designed to exercise the design or code.

Class Testing and the Class Hierarchy

- Inheritance does not obviate the need for thorough testing of all derived classes. In fact, it can actually complicate the testing process.

Scenario-based Test Design

- Scenario-based testing concentrates on what the user does, not what the product does. This means capturing the tasks (via use-cases) that the user has to perform, then applying them and their variants as tests.

OOT Methods: Random Testing at the Class Level

Random testing

- identify operations applicable to a class
- define constraints on their use
- identify a minimum test sequence
  - — an operation sequence that defines the minimum life history of the class (object)
- generate a variety of random (but valid) test sequences
  - — exercise other (more complex) class instance life histories

OOT Methods: Partition Testing

Partition Testing

- reduces the number of test cases required to test a class in much the same way as equivalence partitioning for conventional software.
- state-based partitioning
  - — categorise and test operations based on their ability to change the state of a class
- attribute-based partitioning
  - — categorise and test operations based on the attributes that they use
- category-based partitioning
  - — categorise and test operations based on the generic function each performs.

OOT Methods: Inter-Class Testing

Inter-class testing

- For each client class, use the list of class operators to generate a series of random test sequences. The operators will send messages to other server classes.
- For each message that is generated, determine the collaborator class and the corresponding operator in the server object.
- For each operator in the server object (that has been invoked by messages sent from the client object), determine the messages that it transmits.
- For each of the messages, determine the next level of operators that are invoked and incorporate these into the test sequence.

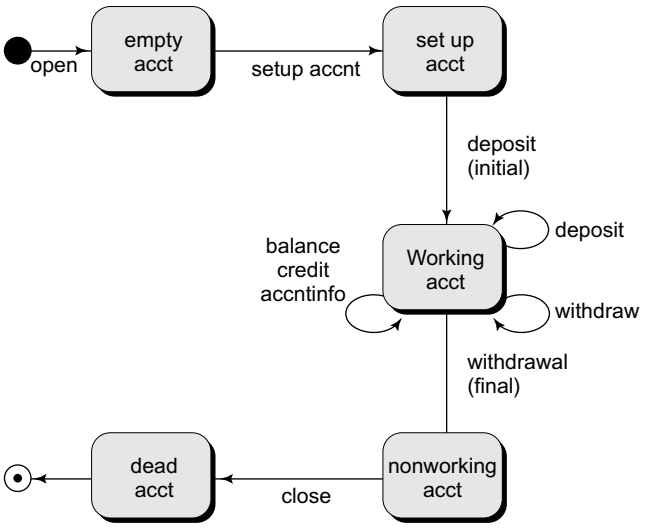


Figure 6.8 State diagram for Account class (adapted from [KIR94])

The tests to be designed should achieve all state coverage [KIR94]. That is, the operation sequences should cause the Account class to make transition through all allowable states.

Testing Patterns

Pattern name: **pair testing**
Abstract: A process-oriented pattern, pair testing describes a technique that is analogous to pair programming in which two testers work together to design and execute a series of tests that can be applied to unit, integration or validation testing activities.

Pattern name: **separate test interface**
Abstract: There is a need to test every class in an object-oriented system, including "internal classes" (i.e., classes that do not expose any interface outside of the component that used them). The separate test interface pattern describes how to create "a test interface that can be used to describe specific tests on classes that are visible only internally to a component."

Pattern name: **scenario testing**
Abstract: Once unit and integration tests have been conducted, there is a need to determine whether the software will perform in a manner that satisfies users. The scenario testing pattern describes a technique for exercising the software from the user's point of view. A failure at this level indicates that the software has failed to meet a user visible requirement. [KAN01]

6.7 Software Quality

Software quality can be defined as:
Conformance to the explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

This implies the existence of a set of standards used by the developer and customer expectations that a product will work well. Conformance to implicit requirements (e.g. ease of use and reliable performance) is what sets software engineering apart from simply writing programs that work most of the time. Several sets of software quality factors are described.

The definition serves to emphasise three important points:

1. Software requirements are the foundation from which quality is measured. Lack of confirmation to requirement is lack of quality.
2. Specified standards define a set of development criteria that guide the manner in which software is engineered. If the criteria are not followed, lack of quality will almost surely result.
3. There is a set of implicit requirements that often goes unmentioned. If software conforms to its explicit requirements but fails to meet implicit requirements, software quality is suspect.

6.7.1 McCall's Quality Factors

McCall's quality factors were proposed in the early 1970s. They are as valid today as they were in that time. It's likely that software built to conform to these factors will exhibit high quality well into the 21st century, even if there are dramatic changes in technology.

The factors that affect software quality can be categorized in two broad groups:

1. factors that can be directly measured (defects uncovered during testing)
2. factors that can be measured only indirectly (usability and maintainability)

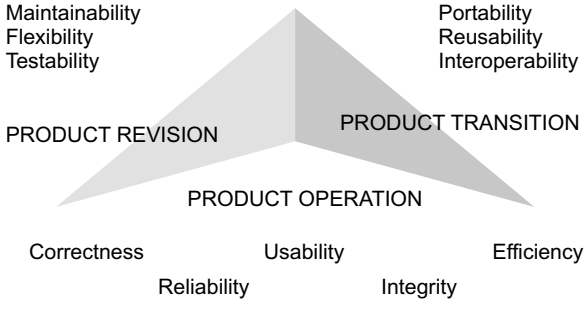


Figure 6.9 McCall's Triangle of Quality

The software quality factors shown above focus on three important aspects of a software product:

- Its operational characteristics
- Its ability to undergo change
- Its adaptability to new environments

Referring to these factors, McCall and his colleagues provide the following descriptions:

Correctness: The extent to which a program satisfies its specs and fulfills the customer's mission objectives.

Reliability: The extent to which a program can be expected to perform its intended function with required precision.

Efficiency: The amount of computing resources and code required to perform is function.

Integrity: The extent to which access to software or data by unauthorised persons can be controlled.

Usability: The effort required to learn, operate, prepare input for, and interpret output of a program.

Maintainability: The effort required to locate and fix errors in a program.

Flexibility: The effort required to modify an operational program.

Testability: The effort required to test a program to ensure that it performs its intended function.

Portability: The effort required to transfer the program from one hardware and/or software system environment to another.

Reusability: The extent to which a program can be reused in other applications related to the packaging and scope of the functions that the program performs.

Interoperability: The effort required to couple one system to another.

6.7.2 9126 Quality Factors

ISO 9126 is an international standard for the evaluation of software quality.

- **Functionality** – A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.
  - Suitability
  - Accuracy
  - Interoperability
  - Compliance
  - Security
- **Reliability** – A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.
  - Maturity
  - Recoverability
- **Usability** – A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.
  - Learnability
  - Understandability
  - Operability
- **Efficiency** – A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.
  - Time Behaviour
  - Resource Behaviour
- **Maintainability** – A set of attributes that bear on the effort needed to make specified modifications.
  - Stability
  - Analysability
  - Changeability
  - Testability
- **Portability** – A set of attributes that bear on the ability of software to be transferred from one environment to another.
  - Installability
  - Replaceability
  - Adaptability

6.7.3 A Framework for Technical Software Metrics

General principles for selecting product measures and metrics are discussed in this section. The generic measurement process activities parallel the scientific method taught in natural science classes (formulation, collection, analysis, interpretation, feedback).

If the measurement process is too time consuming, no data will ever be collected during the development process. Metrics should be easy to compute or developers will not take the time to compute them.

The tricky part is that in addition to being easy compute, the metrics need to be perceived as being important to predicting whether product quality can be improved or not.

6.7.3.1 Measures, Metrics and Indicators

- A ***measure*** provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product or process.
- The IEEE glossary defines a ***metric*** as "a quantitative measure of the degree to which a system, component, or process possesses a given attribute."
- An ***indicator*** is a metric or combination of metrics that provide insight into the software process, a software project, or the product itself.

6.7.3.2 Measurement Principles

- The objectives of measurement should be established before data collection begins.
- Each technical metric should be defined in an unambiguous manner.
- Metrics should be derived based on a theory that is valid for the domain of application (e.g., metrics for design should draw upon basic design concepts and principles and attempt to provide an indication of the presence of an attribute that is deemed desirable).
- Metrics should be tailored to best accommodate specific products and processes.

Measurement Process

- *Formulation.* The derivation of software measures and metrics appropriate for the representation of the software that is being considered.
- *Collection.* The mechanism used to accumulate data required to derive the formulated metrics.
- *Analysis.* The computation of metrics and the application of mathematical tools.
- *Interpretation.* The evaluation of metrics results in an effort to gain insight into the quality of the representation.
- *Feedback.* Recommendations derived from the interpretation of product metrics transmitted to the software team.

Software metrics will be useful only if they are characterised effectively and validated to that their worth is proven.

- A metric should have desirable mathematical properties.
- When a metric represents a software characteristic that increases when positive traits occur or decreases when undesirable traits are encountered, the value of the metric should increase or decrease in the same manner.
- Each metric should be validated empirically in a wide variety of contexts before being published or used to make decisions.

6.7.3.3 The Attributes of Effective Software Metrics

- *Simple and computable.* It should be relatively easy to learn how to derive the metric, and its computation should not demand inordinate effort or time.
- *Empirically and intuitively persuasive.* The metric should satisfy the engineer's intuitive notions about the product attribute under consideration.
- *Consistent and objective.* The metric should always yield results that are unambiguous.
- *Consistent in its use of units and dimensions.* The mathematical computation of the metric should use measures that do not lead to bizarre combinations of unit.
- *Programming language independent.* Metrics should be based on the analysis model, the design model, or the structure of the program itself.
- *An effective mechanism for quality feedback.* That is, the metric should provide a software engineer with information that can lead to a higher quality end product.

6.7.3.4 Metrics for the Analysis Model

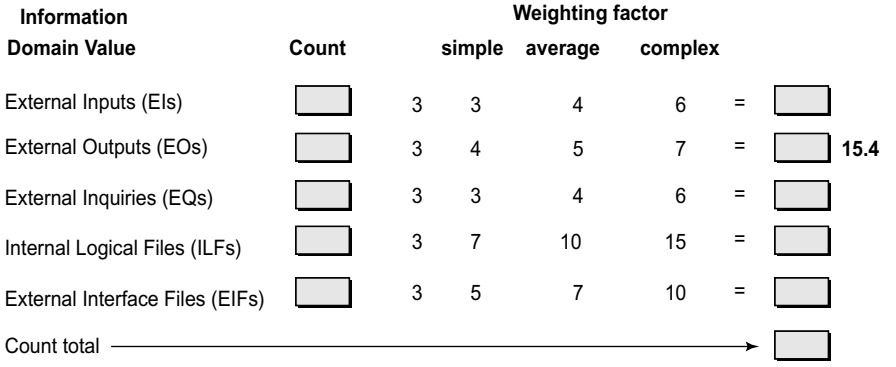
Collection and Analysis Principles

- Whenever possible, data collection and analysis should be automated.
- Valid statistical techniques should be applied to establish relationship between internal product attributes and external quality characteristics.
- Interpretative guidelines and recommendations should be established for each metric.

Analysis Metrics

- Function-based metrics: Use the function point (FP) as a normalising factor or as a measure of the "size" of the specification. FP can be used to:
  1. Estimate the cost required to design, code, and test.
  2. Predict the number of errors that will be encountered during testing.
  3. Forecast the number of components and/or the number of project source lines in the implemented system.
- Specification metrics: Used as an indication of quality by measuring number of requirements by type.
- The *function point metric* (FP), first proposed by Albrecht [ALB79], can be used effectively as a means for measuring the functionality delivered by a system.
- Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity.
- Information domain values are defined in the following manner:
  - number of external inputs (EIs)
  - number of external outputs (EOs)
  - number of external inquiries (EQs)
  - number of internal logical files (ILFs)
  - number of external interface files (EIFs)

Computing Function Points



Metrics for the Design Model

Design metrics for computer software, like all other software metrics, are not perfect. And yet, design without measurement is an unacceptable alternative.

6.7.3.5 Architectural Design Metrics

- Structural complexity = g(fan-out), fan-out is defined as the number of modules immediately subordinate to the module, that is, the number of modules that are directly invoked by module i. Fan-in is defined as the number of modules that directly invoked module i.
- Data complexity = f(input & output variables, fan-out), provides an indication of the complexity in the internal interface for a module i.
- System complexity = h(structural & data complexity), is defined as the sum of structural and data complexity.
  - HK metric: architectural complexity as a function of fan-in and fan-out
  - Morphology metrics: a function of the number of modules and the number of interfaces between modules.

6.7.3.6 Metrics for OO Design

Whitmire [WHI97] describes nine distinct and measurable characteristics of an OO design:

Size: Size is defined in terms of four views: population, volume, length, and functionality.

Complexity: How classes of an OO design are interrelated to one another?

Coupling: The physical connections between elements of the OO design.

Sufficiency: "The degree to which an abstraction possesses the features required of it, or the degree to which a design component possesses features in its abstraction, from the point of view of the current application."

Completeness: An indirect implication about the degree to which the abstraction or design component can be reused.

Metrics for OO Design-II

Cohesion: The degree to which all operations working together to achieve a single, well-defined purpose

Primitiveness: Applied to both operations and classes, the degree to which an operation is atomic

Similarity: The degree to which two or more classes are similar in terms of their structure, function, behaviour, or purpose

Volatility: Measures the likelihood that a change will occur.

6.7.3.6.1 *Class-Oriented Metrics–The CK Metrics Suite*

Weighted methods per class (WMC): The number of methods and their complexity are reasonable indicator of the amount of effort required to implement and test a class.

Depth of the inheritance tree (DIT): The maximum length from the node to root of the tree.

Number of children (NOC): The subclasses that are immediately subordinate to a class in the class hierarchy are termed its children.

Coupling between object classes (CBO): This is the number of collaborations listed for a class on its CRC card. Keep CBO low.

Response for a class (RFC): This is a set of methods that can potentially be executed in response to a message received by an object of that class. RFC is the number of methods in the response set. Keep RFC low.

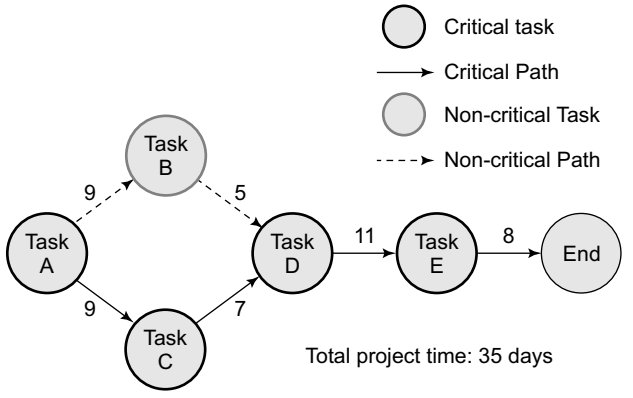
Lack of cohesion in methods (LCOM): This is the number of methods that access one or more of the same attributes. Keep LCOM low.

6.8 Solved Questions

1. Consider the following task along with their efforts. Apply CPM and find out the optimal path.

Task	Immediate prerequisite tasks	Effort (person-days)
A	None	9
B	A	5
C	A	7
D	B, C	11
E	D	8

The following figure illustrates the critical path and non-critical path.



Forward Pass

Task	Tasks precedence	Task length	Earliest possible start time (ES)	Earliest possible finish time (EF)
A	None	9	0	9
B	A	5	9	14
C	A	7	9	16
D	B,C	11	16	27
E	D	8	27	35

Backward pass

Task	Tasks precedence	Task length	Late start time (ES)	Late finish time (EF)
A	None	9	0	9
B	A	5	11	16
C	A	7	9	16
D	B,C	11	16	27
E	D	8	27	36

Total slack time of an activity is the difference in start time between a non-critical task's late start time and its early start time or its late finish time and early finish time.

Total slack time of a task = LS – ES

Or

Total slack time of a task = LF – EF

For example, Activity B: LS – ES (11 – 9)
or LF – EF (16 – 14) => 2

Total slack time is the maximum allowable delay for all non-critical activities.

2. Effort Equation
   a. **PM = C * (KDSI)^{n}** (person-months)
      i. where **PM** = number of person-month (=152 working hours),
      ii. **C** = a constant,
      iii. **KDSI** = thousands of "delivered source instructions" (DSI) and
      iv. **n** = a constant.
3. Productivity equation
   a. **(DSI) / (PM)**
      i. where **PM** = number of person-month (=152 working hours),
      ii. **DSI** = "delivered source instructions"
4. Schedule equation
   a. **TDEV = C * (PM)^{n}** (months)
      i. where TDEV = number of months estimated for software development.
5. Average Staffing Equation
   a. **(PM) / (TDEV)** (FSP)
      i. where FSP means Full-time-equivalent Software Personnel.

OBJECTIVE TYPE QUESTIONS

1. What are the major steps in performing object-oriented design?
   A. Identify the objects and operations
   B. Determine the relationships among objects
   C. Design the driver
   D. A and B above
   E. A, B, and C above

2. Which of the following is not true about structured design (functional decomposition)?
   A. The focus is on actions to be performed.
   B. Control flow plays the primary role; data is secondary.
   C. The top-level algorithm is designed first.
   D. B and C above
   E. None of the above (all of the statements are true)

3. Which of the following is not true about object-oriented design?
- A. The focus is on data and associated operations.
- B. Data plays the primary role; control flow is secondary.
- C. The top-level algorithm is designed first.
- D. A and C above
- E. None of the above (all of the statements are true)

4. In object-oriented design, which of the following is not likely to be a problem-domain object?
- A. A factory assembly line
- B. A robot arm
- C. A sorted list
- D. A and B above
- E. A, B, and C above

5. Which of the following best describes the objects that are listed initially in an object table?
- A. They are problem-domain objects that definitely become solution-domain objects.
- B. They are problem-domain objects that may or may not become solution-domain objects.
- C. They are solution-domain objects that definitely become problem-domain objects.
- D. They are solution-domain objects that may or may not become problem-domain objects.
- E. They are solution-domain objects for which there are no equivalent problem-domain objects.

6. In the object-oriented design of a card-playing program, suppose that "card deck" has been identified as one object and "user command" has been identified as another object. Focusing specifically on the card deck object, how does it relate to the user command object?
- A. A has-a relationship
- B. An is-a relationship
- C. An independent and equal relationship
- D. A and B above
- E. None of the above

7. Consider the following portion of a problem definition:

The program displays a menu to the user. If the user responds with 'M', the program displays emails of the user. If the user responds with 'x', the program exits.

In an object-oriented design, which of the following probably would *not* be identified as an object?
- A. Menu
- B. User
- C. Time of day
- D. Date
- E. Response

8. The nature of software applications can be characterized by their information
- A. Complexity
- B. Content
- C. Determinacy
- D. Both B and C

9. Which of the items listed below is not one of the software engineering layers?
- A. Process
- B. Smelting
- C. Methods
- D. Tools

10. Which of these are the 5 generic software engineering framework activities?
- A. Communication, planning, modeling, construction, deployment
- B. Communication, risk management, measurement, production, reviewing
- C. Analysis, designing, programming, debugging, maintenance
- D. Analysis, planning, designing, programming, testing

11. Process models are described as agile because they
- A. Eliminate the need for cumbersome documentation
- B. Emphasize maneuverability and adaptability
- C. Do not waste development time on planning activities
- D. Make extensive use of prototype creation

12. Which of these terms are level names in the Capability Maturity Model?
- A. Performed
- B. Repeated
- C. Reused
- D. Optimised
- E. Both A and D

13. The incremental model of software development is
- A. A reasonable approach when requirements are well defined.
- B. A good approach when a working core product is required quickly.
- C. The best approach to use for projects with large development teams.
- D. A revolutionary model that is not used for commercial products.

14. The rapid application development model is
- A. Another name for component-based development.
- B. A useful approach when a customer cannot define requirements clearly.
- C. A high speed adaptation of the linear sequential model.
- D. All of the above.

15. Evolutionary software process models
- A. Are iterative in nature
- B. Can easily accommodate product requirements changes

C. Do not generally produce throwaway systems
D. All of the above

16. The prototyping model of software development is
A. A reasonable approach when requirements are well defined.
B. A useful approach when a customer cannot define requirements clearly.
C. The best approach to use for projects with large development teams.
D. A risky model that rarely produces a meaningful product.

17. The spiral model of software development
A. Ends with the delivery of the software product
B. Is more chaotic than the incremental model
C. Includes project risks evaluation during each iteration
D. All of the above

18. The concurrent development model is
A. Another name for the rapid application development model.
B. Often used for the development of client/server applications.
C. Only used for development of parallel or distributed systems.
D. Used whenever there are a large number of changes

19. The component-based development model is
A. Only appropriate for computer hardware design.
B. Not able to support the development of reusable components.
C. Works best when object technologies are available for support.
D. Not cost effective by known quantifiable software metrics.

20. The result of the requirements engineering elaboration task is an analysis model that defines which of the following problem domain(s)?
A. Information B. Functional
C. Behavioral D. All of the above

21. The use of traceability tables helps to
A. Debug programs following the detection of run-time errors
B. Determine the performance of algorithm implementations
C. Identify, control, and track requirements changes
D. None of the above

22. Which of these is not an element of an object-oriented analysis model?
A. Behavioral elements
B. Class-based elements
C. Data elements
D. Scenario-based elements

23. UML activity diagrams are useful in representing which analysis model elements?
A. Behavioral elements
B. Class-based elements
C. Flow-based elements
D. Scenario-based elements

24. The data flow diagram
A. Depicts relationships between data objects
B. Depicts functions that transform the data flow
C. Indicates how data are transformed by the system
D. Indicates system reactions to external events
E. Both B and C

25. Control flow diagrams are
A. Needed to model event driven systems.
B. Required for all systems.
C. Used in place of data flow diagrams.
D. Useful for modeling real-time systems.
E. Both A and D

26. Which of the following are areas of concern in the design model?
A. Architecture B. Data
C. Interfaces D. Project scope
E. A, B and C

27. The importance of software design can be summarised in a single word
A. Accuracy B. Complexity
C. Efficiency D. Quality

28. A useful technique for evaluating the overall complexity of a proposed architecture is to look at the component
A. Cohesion flow
B. Dependencies
C. Sharing dependencies
D. Size
E. Both B and C

29. In component-level design "persistent data sources" refer to
A. Component libraries
B. Databases
C. Files
D. All of the above
E. Both B and C

30. Which of the following need to be assessed during unit testing?
A. Algorithmic performance
B. Code stability
C. Error handling
D. Execution paths
E. Both C and D

31. Regression testing should be a normal part of integration testing because as a new module is added to the system, new
A. Control logic is invoked
B. Data flow paths are established
C. Drivers require testing
D. All of the above
E. Both A and B

32. Smoke testing might best be described as
A. bulletproofing shrink-wrapped software
B. rolling integration testing
C. testing that hides implementation errors
D. unit testing for small programs

33. Which of the following are characteristics of testable software?
A. Observability B. Simplicity
C. Stability D. All of the above

34. The testing technique that requires devising test cases to demonstrate that each program function is operational is called
A. Black-box testing B. Glass-box testing
C. Grey-box testing D. White-box testing

35. The testing technique that requires devising test cases to exercise the internal logic of a software module is called
A. Behavioral testing B. Black-box testing
C. Grey-box testing D. White-box testing

36. What types of errors are missed by black-box testing and can be uncovered by white-box testing?
A. Behavioral errors B. Logic errors
C. Performance errors D. Typographical errors
E. Both B and D

37. The cyclomatic complexity metric provides the designer with information regarding the number of
A. Cycles in the program
B. Errors in the program
C. Independent logic paths in the program
D. Statements in the program

38. Data flow testing is a control structure testing technique where the criteria used to design test cases is that they
A. Rely on basis path testing
B. Exercise the logical conditions in a program module
C. Select test paths based on the locations and uses of variables
D. Focus on testing the validity of loop constructs

39. Loop testing is a control structure testing technique where the criteria used to design test cases is that they
A. Rely basis path testing
B. Exercise the logical conditions in a program module
C. Select test paths based on the locations and uses of variables
D. Focus on testing the validity of loop constructs

40. Black-box testing attempts to find errors in which of the following categories?
A. Incorrect or missing functions
B. Interface errors
C. Performance errors
D. All of the above
E. None of the above

41. Product quality is defined as:
A. Delivering a product using correct development procedures
B. Delivering a product which is developed iteratively
C. Delivering a product with correct requirements
D. Delivering a product using high quality procedures
E. Delivering an initial product and changing its once released to meet customer requirements

ANSWER KEY

1. E	**2.** E	**3.** C	**4.** C
5. B	**6.** C	**7.** B	**8.** D
9. B	**10.** A	**11.** B	**12.** E
13. B	**14.** C	**15.** D	**16.** B
17. C	**18.** B	**19.** C	**20.** D
21. C	**22.** C	**23.** D	**24.** E
25. E	**26.** E	**27.** D	**28.** E
29. E	**30.** E	**31.** E	**32.** B
33. D	**34.** A	**35.** D	**36.** E
37. B	**38.** C	**39.** D	**40.** D
41. C			

Previous Years' GATE Questions

1. A company needs to develop a strategy for software product development for which it has a choice of two programming languages L1 and L2. The number of lines of code (LOC) developed using L2 is estimated to be twice the LOC developed with L1. The product will have to be maintained for five years. Various parameters for the company are given in the table below.

Parameter	Language L1	Language L2
Man years needed for development	LOC / 10000	LOC / 10000
Development Cost per year	Rs. 10,00,000	Rs. 7,50,000
Maintenance Time	5 years	5 years
Cost of maintenance per year	Rs. 1,00,000	Rs. 50,000

Total cost of the project includes cost of development and maintenance. What is the LOC for L1 for which the cost of the project using L1 is equal to the cost of the project using L2? **(GATE 2011)**

A. 4000 B. 5000

C. 4333 D. 4667

Explanation: Let L1, L2 be the LOC's of both the languages. Thus, L2=2L1.

Now, we equate the total cost of both the languages.

x/10000*1000000+5*100000=2x/10000*750000+5*50

x=5000

2. A company needs to develop digital signal processing software for one of its newest inventions. The software is expected to have 40000 lines of code. The company needs to determine the effort in person months needed to develop this software using the basic COCOMO model. The multiplicative factor for this model is given as 2.8 for the software development on embedded systems, while the exponentiation factor is given as 1.20. What is the estimated effort in person months?

(GATE 2011)

A. 234.25 B. 932.50

C. 287.80 D. 122.40

Explanation: Effort in person months = $\alpha(KDSI)^{\beta}$ where KDSI is the code size in kilo lines. By substituting the given values, we get $=2.8*(40)^{1.2}=234.25$

3. What is the appropriate pairing of items in the two columns listing various activities encountered in a software life cycle? **(GATE 2010)**

P. Requirements Capture	1. Module Development and Integration
Q. Design	2. Domain Analysis
R. Implementation	3. Structural and Behavioral Modeling
S. Maintenance	4. Performance Tuning

A. P-3, Q-2,R-4,S-1 B. P-2, Q-3,R-1,S-4

C. P-3, Q-2,R-1,S-4 D. P-2, Q-3,R-4,S-1

4. The following program is to be tested for statement coverage:

```
begin
if (a== b) {S1; exit;}
else if (c== d){ S2;}
else {S3; exit;}
S4;
end
```

The test cases T1, T2, T3 and T4 given below are expressed in terms of the properties satisfied by the values of variables a, b, c and d. The exact values are not given.

T1 : a, b, c and d are all equal

T2 : a, b, c and d are all distinct

T3 : a=b and c !=d

T4 : a !=b and c=d

Which of the test suites given below ensures coverage of statements S1, S2, S3 and S4? **(GATE 2010)**

(A) T1, T2, T3 (B) T2, T4

(C) T3, T4 (D) T1, T2, T4

Explanation: If T1 is given S1 will be covered and exited. If T2 is the given one, then S3 will be executed and exited. If T4 is given S2 will be executed then S4. Thus, all S2, S2, S3 and S4 are covered.

5. The coupling between different modules of a software is categorised as follows: **(GATE 2009)**

I. Content coupling

II. Common coupling

III. Control coupling

IV. Stamp coupling

V. Data coupling

Coupling between modules can be ranked in the order of strongest (least desirable) to weakest(most desirable) as follows:

A. I-II-III-IV-V

B. I-III-V-II-IV

C. V-IV-III-II-I

D. IV-II-V-III-I

6. In a software project, COCOMO (constructive cost model) is used to estimate

A. Effort and duration based on the size of the software

B. Size and duration based on the effort of the software

C. Effort and cost based on the duration of the software

D. Size, effort and duration based on the cost of the software

7. The diagram that helps in understanding and representing user requirements for a software project using UML (unified modeling language)) is

A. Entity relationship diagram

B. Deployment diagram

C. Data flow diagram

D. Use case diagram

8. A software organisation has been assessed at SEI CMM Leve l4. Which of the following does the organisation need to practice beside Process Change Management and Technology Change Management in order to achieve Level 5?

A. Defect Detection. B. Defect Prevention.

C. Defect Isolation D. Defect Propagation.

9. A software configuration management tool helps in

A. Keeping track on the schedule based on the milestones reached

B. Maintaining different versions of the configurable items

C. Managing manpower distribution by changing the project structure

D. All of the above

10. A software project involves execution of 5 tasks T1,T2,T3,T4 and T5 of duration 10,15,18,30 and 40 days respectively. T2 and T4 can start only after T1 completes. T3 can start after T2 completes. T5 can start only after both T3 and T4 complete. What is the task T3 in days?

A. 0 B. 3

C. 18 D. 30

11. Assume that the delivered lines of code L of a software is related to the effort E in person months and duration t in calendar months by the relation $L\ P^* (E/B)^{1/3} * t^{4/3}$, where P and B are two constants for the software process and skills factor. For a software project, the effort was estimated to be 20 person months and the duration was estimated to be 8 months. However, the customer asked the project team to complete the software project in 4 months. What would be the required effort in person months?

A. 10 B. 40

C. 160 D. 320

12. A software was tested using the error seeding strategy in which 20 errors were seeded in the code. When the code was tested using the complete test suite, 16 of the seeded errors were detected. The same test suite also detected 200 non-seeded errors. What is the estimated number of undetected errors in the code after this testing?

A. 4 B. 50

C. 200 D. 250

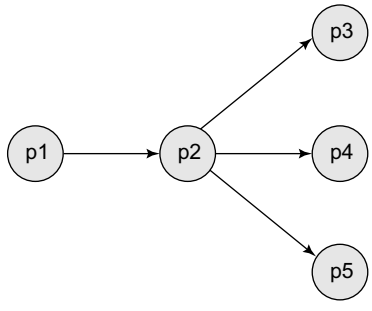
13. The availability of a complex software is 90%. Its Mean Time Between Failure (MTBF) is 200 days. Because of the critical nature of the usage ,the organization deploying the software further enhanced it to obtain the availability of 95%. In the process, the Mean Time To Repair (MTTR) increased by 5 days.

What is the MTBF of the enhanced software?

A. 205 days B. 300 days

C. 500 days D. 700 days

14. In a data flow diagram, the segment shown below is identified as having transaction flow characteristics, with p2 identified as the transaction center.



A first level architectural design of this segment will result in a set of process modules with an associated invocating sequence. The most appropriate architecture is

A. P1 invokes p2, p2 invokes either p3 or p4 or p5

B. P2 invokes p1, and then invokes p3, or p4 or p5

C. A new module Tc is defined to control the transaction flow. This module Tc first invokes P1 and then invokes P2. P2 in turn invokes p3, or p4 or p5

D. A new module Tc is defined to control the transaction flow. This module Tc invokes P2, p2 invokes P1, and then P3 or P4 or p5.

15. A software project has four phases P1, P2, P3 and P4. Of these phases, P1 is the first one and needs to

be completed before any other phase can commence. Phases P2 and P3 can be executed in parallel. Phase P4 cannot commence until both P2 and P3 are completed. The optimistic, most likely and pessimistic estimates of the phase completion times in days, for P1, P2, P3 and P4 are respectively, (11,15,25), (7,8,15), (8,9,22) and (3,8,19).

The critical path for the above project and the slack of P2 are, respectively

A. P1-P2-P4, 1 day B. P1-P3-P4,1 day
C. P1-P3-P4, 2 day D. P1-P2-P4, 2 day

16. The costs (in rupees per day) of crashing the expected phase completion times for the four phases respectively are 100, 2000, 50 and 1000. Assume that the expected phase completion times of the phases cannot be crashed below their respective most likely completion times. The minimum and maximum amounts (in rupees) that can be spent on crashing so that ALL paths are critical are, respectively

A. 100 and 1000 B. 100 and 1200
C. 150 and 1200 D. 200 and 2000

17. In the spiral model of software development, the primary determinant in selecting activities in each iteration is

A. Iteration size
B. Cost
C. Adopted process such as rational unified process or extreme programming
D. risk

18. Find the following statements in the context of software testing are true or false (S1) Statement coverage cannot guarantee execution of loops in a program under test (S2) Use of independent path testing criterion guarantees execution of each loop in a program under test more than once

A. True, True B. True, False
C. False, True D. False, False

19. A software project plan has identified ten tasks with each having dependencies as given in the following table:

Task	Depends on
T1	
T2	T1
T3	T1
T4	T1
T5	T2
T6	T3
T7	T3,T4
T8	T4
T9	T5,T7,T8
T10	T6,T9

On the basis of above table answer the following.

(Q1)What is the maximum number of tasks that can be done concurrently?

(Q2) What is the minimum time required to complete the project, assuming that each task requires one time unit and there is no restriction on the number of tasks that can be done in parallel?

A. 5,6 B. 4,4
C. 4,5 D. 5,4

20. A software engineer is required to implement two sets of algorithms for a single set of matrix operations in an object-oriented programming languages, the two sets of algorithms are to provide precisions of 10-3 and 10-6, respectively. She decides to implement two classes, Low precision matrix and high precision matrix, providing precisions 10-3 and 10-4 respectively. Which of the following is the best alternative for the implementation?

(S1) The two classes should be kept independent

(S2) Low prevision matrix should be derived from high precision matrix

(S3) High precision matrix should be derived from low precision matrix

(S4) One class should be derived from the other; the hierarchy is immaterial.

A. S1 B. S2
C. S3 D. S4

ANSWER KEY

1. B **2.** A **3.** B **4.** D
5. C **6.** C **7.** A **8.** B
9. B **10.** B **11.** D
12. Ambiguous question **13.** D
14. C **15.** Ambiguous question
16. Ambiguous question **17.** D
18. D **19.** D **20.** C

CHAPTER SEVEN

Computer Networks

7.1 Introduction to Networks

Computer network is a system (collection and connection of computers) which allows two or more computers to communicate. Elements of computer network are intermediate message processors such as bridges, routers and gateways in addition to computers.

7.1.1 Classification of Networks

Networks can be classified based on

- Range of communication
- Relationship between components
- Physical design structure (topology)
- Communication techniques (protoco)

7.1.1.1 Networks by Range (LANs, MANs and WANs)

Local Area Network (LAN)
This is a network typically set up in a home, office or small group of buildings. The range of this type of network is limited, confined to an area of 1000 square metres. The effect of this small range is that no computer on this network will be further than 30-50 metres from its nearest neighbour. The number of computers/devices on an individual LAN will normally not exceed 50. Note that multiple LANs can be linked together to provide greater network access/coverage.

Metropolitan Area Network (MAN)
This is a network that spans a city or extended group of buildings as in a college campus. To enhance speed MANs often use fibre optic cables to connect segments of the network. The optical cables allow for extremely high rates of data transfer. A number of LANs may be joined together to form part of a MAN. It is expected that several hundred computers may be present on a MAN.

Wide Area Network (WAN)
This type of network spans a wide geographical area. Due to the dispersed nature of the computers, WANs must handle and support multiple networking technologies. In contrast to a typical LAN, users may not be using the same types of hardware/software. The need for the constant availability of the WAN means that dedicated personnel, equipment and lines must be used. The internet is an extreme example of a WAN. Large organisations may also maintain their own private WANs with the use of leased lines. The existence of these networks provide global reach. Many private WANs are also connected to the internet but outsiders cannot enter without providing the relevant usernames/passwords or other security information.

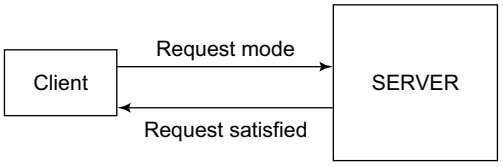
Note

Internet is global and refers to WAN. Intranet refers to LAN.

7.1.1.2 Networks by Relationships between Components

Client/Server
This is a relationship primarily between two computers in which one machine makes requests (called the client) and the other machine (called the server) satisfies the requests. For example, when a user downloads a web page they are actually making a request for information from another machine (called the *web* server).

Typically the server is a powerful machine which uses special hardware to improve performance. The server may be connected to thousands of clients at a time. The client typically will be a PC or machine with reduced computing ability when compared to the server.



Client/Server setup

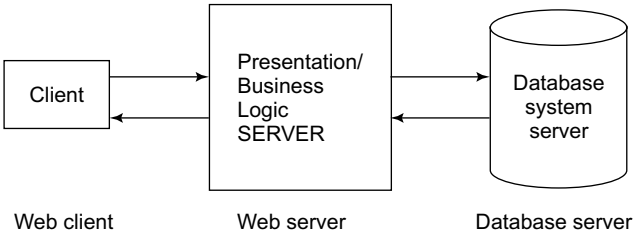
Peer-to-peer

This is a setup where two computers (both clients) communicate directly with each other, acting alternately as client (i.e., making requests) and server (i.e., satisfying requests). The machines typically will be on the same level in terms of hardware/software capabilities unlike the client/server setup where the server is the more powerful machine.

There is no concept of a server in a true peer-to-peer setup, only clients (i.e., each machine is responsible for making and satisfying requests as is required).

Multi-tier (n-tier)

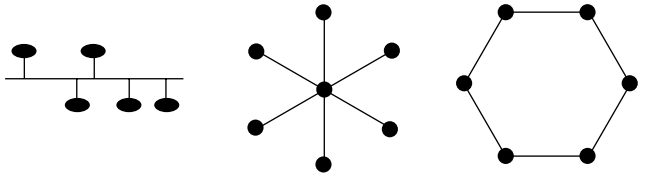
Under this setup the functions of the server are distributed throughout a number of tiers. This model has been adopted for e-commerce websites. The database system may be stationed on an independent machine which allows for greater efficiency, while the server which produces the web pages is in contact with the client. In more advanced setups four or more tiers may be needed.



Internet 3-tier setup

7.1.1.3 Networks by Physical Design Structure (Topologies)

A network can be visualised as a series of nodes connected by links. The links represent communication pathways to other machines. The topology of a network is determined only by the configurations of connections between nodes. Network topology is not concerned with the things that physically affect the network such as signal strength, distances between nodes, tranmission rates, etc. Topology is seen as a subfield in the field of Graph Theory which belongs to Computer Science/Mathematics.



Network Topologies

Bus topology: This is a network in which a single line (called the bus) connects all the nodes in the network. Communication occurs up and down this pathway.

Advantages

1. Easy to implement
2. Cheap
3. Failure of a node does not affect others

Disadvantages

1. Difficult to pinpoint errors (troubleshoot)
2. Performance may decrease as number of nodes is increased
3. Limited cable length and nodes

Star topology: This is a network in which one central node connects all peripheral nodes. Transmissions are sent to the central node and then re-broadcasted to all peripheral nodes.

Advantages

1. Quick setup
2. Easy to pinpoint errors
3. A cable break may not bring down entire network

Disadvantages

1. Maintenance costs may be high
2. Performance may decrease as number of nodes is increased
3. Limited cable length and nodes

Ring topology: This is a network in which each node has two other nodes connected to it. The nodes form a circular arrangement of communication channels. Transmissions from one node may travel through several others before reaching the targeted destination node. Transmission may occur in one direction only.

Advantages

1. Performance maintained as nodes added
2. Equal access granted to all nodes

Disadvantages

1. Expensive
2. One node failing may affect others

7.1.2 Communication Architectures

There are a number of network architectures, and two of the most widely used are :

- ISO Open Systems Interconnection (OSI) 7 layer reference model.
- TCP/IP reference model

Both of them follows layered architecture. Each layer is intended to perform a specific task in the overall problem of communication. Each layer is independent of all the others. Communication with the layers immediately above

and below is via a well defined interface. Layer N is said to request service from layer N-1 (below) and provide a service to the layer N+1 (above). Layer N in one protocol stack communications with the same layer in a remote protocol stack via the layers below. This is known as virtual or peer-to-peer communication.

OSI Model :	**TCP/IP Model :**
Application Layer	Application Layer
Presentation Layer	Transport Layer
Session Layer	Network Layer
Transport Layer	Data Link Layer
Network Layer	Physical Layer
Data Link Layer	
Physical Layer	

7.1.2.1 Comparison of TCP/IP and OSI Reference Models

Both Models have similarities and differences.
Similarities

1. Both use the concept of a protocol architecture, where there are a number of independent layers, each carrying out a specific task.
2. The functionality is very similar for most of the layers in each reference model. For example, both have a transport layer which operates end-to-end.

Differences

1. OSI reference model makes clear the distinction between services, interfaces and protocols. The service defines what services the layer offers, the interface defines how they are accessed and the protocols are the actual implementation of the services. This adheres to standard software engineering practice. In contrast, the TCP/IP Reference Model does not use this approach and hence the protocols (implementations) are not always transparent.
2. TCP/IP has no presentation or session layer.
3. OSI supports connection-oriented and connectionless communication in the network layer, but only connection-oriented communication in the transport layer. In contrast, TCP/IP allows only connectionless communication in the network layer but a choice of connection-oriented and connectionless in the transport layer.
4. The OSI defines very precisely the physical and data link layers. TCP/IP ignores this approach and instead the Host-to-network layer merely defines the *interface* to the underlying network.

Some Important Points

To be critical, **data communication** deals with aspects related to data link layer and physical layer and **computer networks** deals with remaining layers. Moreover, PL and DLL aspects are realised in HW i.e, in formware. For example, network cards or ethernet cards are the ones which carries the activities of DLL and PL. Whereas, NL, TL, and AL are implemented in SW. CISCO people have implemented NL, TL related services in their routers.

Protocols means rules and regulations but they will not tell anything about how they are implemented or realised.

Physical layer protocol specifies physical, electrical, mechanical, procedural, functional interface between a data terminating equipment (computer) and data communicating equipment such as modem.

Null Modem is used to connect two computers via their RS 232 ports either COM1 or COM2 First popular serial communication software was Kermit developed by Cambridge University, UK.

Data units are normally divided into smaller units while communicating

- To utilise the bandwidth in a better manner
- Not to allow one application to monopolise the lines
- Because of the protocol constraints, s/w constraints such as buffer sizes etc
- Because of the error characteristics of the practical channels.

(For example, if a channel is identified as having error rate as 1 in 100000 bits and the size of the communication data unit is more than 100000 bits than every communication data unit is bound to face an error. Even if we transmit, the retransmitted one also going to face error. Thus, there will not be any effective data transfer. Thus, frames data communication units < 100000 bits are used. That to preferable their sizes should be less than 1/2*100000).

Data Link layer main responsibility is successfully delivery of frame from one side of a link to another side of a link in the case of point to point networks. Its responsibility also includes acknowledge management, flow control, slow receiver problem.

Flow control is related to one particular link and is done by data link layer whereas congestion control is a global phenomenon carried out by many m/c's in the subnet and this is the responsibility of the network layer.

Main responsibility of network layer is delivery of packet from source m/c to destination m/c (peer communication). Its responsibility also includes route finding, congestion control, deadlock avoidance, extending connection oriented and connectionless services.

Transport Layer responsibility is connection establishment, management, multiplexing of services, etc.

Session Layer is related to management of connections after failures i.e., after connection failures when the connection is re-established, the data transfer should start from the point where it is broken.

Presentation layer is concerned with data encryption, compression, etc, of communicatable data units. In the case of TCP/IP model this is normally implemented as an application layer service.

Data is usually sent by varying some physical parameters such as volatge or current.

Bandwidth Limited signals are the ones which are made to have some range of frequencies only. The channel also limits the signal bandwidth. Filters are used to limit the amount of bandwidth available to a customer. Limiting the bandwidth limits the data rate.

7.1.3 Switching Methods

There are 2 types of switching methods

- Circuit switching
- Packet switching

7.1.3.1 Circuit Switching

Set up a dedicated end-to-end connection. Switching implies that the connection is switched through a number of intermediate exchanges. For example, present telephone networks, mobile cellular networks

7.1.3.2 Packet Switching

Information is broken into segments. These segments are called *packets* at layer 3 and *frames* at layer 2. More generally they are called *Protocol Data Units* (PDUs). Packets are sent individually through the network. For example, Internet, Superhighway, most data networks

Advantages and disadvantages of switching methods
Circuit switching

- Private, secure, not subject to congestion
- But inefficient use of bandwidth, pay for time call is connected regardless of amount of data

Packet switching

- Shared use of high cost components, efficient use of bandwidth, only pay for data in transit
- But not secure or private, subject to congestion

7.1.4 Delays Associated with Networks

Propagation Delay

- Time taken for a signal to travel from the transmitter to the receiver
- Speed of light is the fastest a signal will propagate

3×10^8 m/sec through space
2×10^8 m/sec through copper

Transmission Delay (Time)

- Time taken to put the bits on the transmission media

Transmission speed of 2Mbps means
2 X 106 bits can be transmitted in 1 second

Processing Delay

- Time taken to execute protocols

check for errors
send Acks etc.

Queuing Delay

- Only in packet switched networks
- Time spent waiting in buffer for transmission
- Increases as load on network increases

Round Trip Delay

Round trip delay is defined as the time between the first bit of the message being put onto the transmission medium, and the last bit the acknowledgement being received back by the transmitter. It is the sum of the all the delays detailed above. The round trip delay is a critical factor in the performance of packet switched protocols and networks. Indeed, it has been stated that a good algorithm for estimating the round trip delay is at the heart of a good packet switch protocol.

7.1.5 Bandwidth

- Bandwidth is a measurement of the width of a range of frequencies and is measured in hertz (Hz).
- In data networks bandwidth is normally specified as bits per second (BPS)
- Shannon-Hartley Theorem states that
  Dmax = Blog2(1 + S/N)
  where Dmax is the maximum bit rate
  B is the bandwidth in Hz
  and S/N is the signal to noise ratio

All transmission mediums are degraded by 'noise'. If the average power of the signal is given by S, and the average power of the noise is given by N, then the signal to noise ratio is given by S/N. The greater the value of S/N then the greater is the theoretical transmission rate of that medium.

7.1.5.1 Bandwidth of a Signal

According to fourier series any arbitrary signal can be decomposed into a set of periodic components which are commonly known as integral component of that signal. Out of these components, the component with lowest frequency value and component with highest frequency value are said to be the band of frequencies available in the signal. The difference in their frequencies is known as bandwidth of that signal. In practice, the smallest component frequency

may be very small and often negligible. Thus bandwidth of the signal can be mentioned as the frequency of the largest frequency component.

Bandwidth of a channel is the largest frequency of the signal which the channel can carry without much distortion

Bandwidth of human voice is 4 KHz whereas and width of video is 5-6 MHz.

Baud rate is the number of signal transitions per unit time or number of signalling elements per unit time.

Data Rate = $\log_2$ V * Baud rate

Where,

V = number of voltage levels

For a binary valued channel, baud rate will be same as the data rate. For multi-valued channels, data rate will be always more than baud rate.

7.1.5.2 Shannon's Sampling Theorem

Maximum number of samples should not be more than 2 times of the bandwidth of the signal. If the sampling rate is more than the Shannon's rate, we may get aliasing effect or ghost signals will be coming into the system.

Explain why for real-time voice communication, 64 Kbps lines are required?

Answer: Voice bandwidth =4KHz

∴ According to Shannon's sampling theory the sampling rate is 8000 samples/sec.

∴ Data generated for one second = 8000x8bits (assuming each sample is represented through 8-bits).

= 64000bps = 64kbps

Note

In data communication, kilo = 10^3, Mega=10^6, Tera=10^9, …

Pulse coded modulation encoded symbols normally takes 8 bits/sample

According to Nyquists theorem, (it talks about channel)

Max data rate of a channel = 2*H*$\log_2$ V

H = Bandwidth of channel

V = Number of voltage levels which channel carried

Note

Unless otherwise told, assume channels are binary valued channels. That is, they carry signals of two levels only.

This theorem is applicable for ideal channels only.

For Practical channels,

Max data rate = H * $\log_2$ (1 + S/N)

S/N is called signal to noise ratio

SNR value = 10 * $\log_{10}$ S/N dB (units for SNR is decibals)

1. A channel bandwidth is 3 KHz and SNR value is marked as 30 decibals. Calculate the max achievable data rate on this channel ?

Answer: SNR = 10 * $\log_{10}$ S/N

30 = 10 * $\log_{10}$ S/N

S/N = 1000

Max data rate =3000 * $\log_2$ (1 + 1000) = 30,000 bps

Note

Two materials (channels) are given with different SNR values in decibals, the one with higher SNR value is preferred as it is less immune to external disturbance or noise.

2. A 4 KHz band width line is proposed to send digital voice. What should be the SNR value required for the same?

Answer: Digital voice date rate = 64000bps

64000 = 4000 * $\log_2$ (1+ S/N)

S/N = 65535

SNR = 10 * $\log_{10}$ 65535 ~ 43 dB

3. A video telephone which sends 200X150 pixels per frame, 15 frames/sec is connected to a line channel whose band width is 8 kHz. It supports 32 shades monochrome pictures only. Calculate what should be the required SNR value for channel to carry this video phone o/p in real time.

Answer: As 32 shades are used, each pixel requires 5 bits.

Video telephone o/p = 64kbps + 200 * 150 * 15 * 5

(implicit audio + video)

= 2314 kbps

2314 = 8000 * $\log_2$ (1 + S/N)

S/N = 1.183E87

then,

SNR = 10 * $\log_{10}$ (S/N) = 871dB

4. A fax m/c supports two scanning resolutions 120 dpi and 300 dpi (The first one is known as standard mode and second mode is fine mode). This is connected to a 4 kHz line It is reported that max achievable data rate on this line is 19,200 bps. Find out how much time it may take to send an A4 size page in std mode and fine mode.

Answer: Standard mode:

A4 size = 10″ X 8″

Data required to send = 10″ X 8″ X 120 X 120 X 1 bits

So,

Time required = (10 X 8 X 120 X 120) / 19200 = 60 secs.

Fine Mode:

Data required to send = 10 X 8 X 300 X 300 X 1 bit

So,

Time required = (10 X 8 X 300 X 300)/19200 = 375 sec

5. Television channels are 6MHz wide. How many bits/sec can be sent if four-level digital signals are used?

Answer: Data rate=2x10x10^6x$\log_2$4=24Mbps

6. If a binary signal is send over a 3KHz channel whose S/N is 20. What is the maximum achievable data rate?.

Answer: MaxachievableDataRate=3x$10^3$$\log_2$(1+20)=15Kbps

7. If our receiving and transmitting devices can distinguish among four different voltage levels in a given signal instead of just two, how many bits can be transmitted in a single signal element? If the baud rate is 1200 what is the data rate we can achieve?

Answer: V= 4

$\therefore$ No of bits a signal element can carry = $\log_2 4 = 2$

$\therefore$ Data Rate = 2*1200=2400bps

8. Given a channel with intended capacity of 20Mbps, the bandwidth of the channel is 3MHz. What SNR is needed in order to achieve this capacity?

Answer: 20Mbps = 3MHz $\log_2(1+S/N)$

$S/N = 2^{5.66}$

SNR = $10 \log_{10} 2^{5.66}$ =17 dB

9. A five-bit start-stop asynchronous transmission with start and data pulses of each 13.5min duration plus a stop pulse of 19min giving a total character duration of 100min is used? What is the signalling speed? What is the data rate?

Answer: Signalling speed = baud rate= 1/13.5ms=74

Data rate = 5bits/100ms= 50 bps

10. It is desired to send a sequence of computer screen images over an optical cable. The screen images are 640x480 pixels each and each pixel requires 24bits. It is required to send 60 frames per second?. What is the minimum bandwidth needed for the channel which is supposed to carry this data?

Answer: Data Required to be sent = 60 X 640 X 480 X 24= 442.368

Bandwidthrequiredforthechannel=442.368/2=221.184MHz

11. A Modem to be used with a PSTN (public switched telephone network) which used amplitude and phase shift keying with eight levels per signalling element. If the bandwidth of the PSTN is 3100Hz, deduce the Nyquist maximum data rate.

Answer: Nyquist data rate = 2 X 3100 X $\log_2 8$ = 18600bps

12. Deduce maximum data rate of a tele networking with a bandwidth of 500 Hz and a SNR of 5dB.

Answer: 5 = $10 \log_{10}$ S/N. Therefore, S/N=3.16

Max data rate = 599 X $\log_2$ (1+3.16) =1000bps

13. Assume that the TV picture is sent over a channel with 4.5 Mhz bandwidth and 35dB SNR value. Find the capacity of the channel.

Answer: L 35=$10\log_{10}$ S/N Therfore, S/N=3162.3

Data Rate = 4.5 X 10^6 x $\log_2$(1+3162.3) =52Mbps

14. What is the channel capacity for a teleprinter channel with 300Hz bandwidth and S/N value of 3?.

Answer: Date Rate = 300 X $\log_2$(1+3) =600bps

15. For a video conferencing system a 10MHz channel is given. Video camera outputs 30 frames for a second with 512x512, 8-bit pixels. At the same rate images are required to be transferred due to some technical reasons (real time video communication). Will the given channel is adequate? If it is not possible, it is proposed to compress and send. If so, how much compression ratio is needed for the compression algorithm selected?

Answer: Data rate = 30 X 512 X 512 X 8= 62.9Mbps

What max data rate we can achieve on the channel = 2X 10 MHz X $\log_2$2 =20Mbps

(As nothing is given about voltage levels we assume binary signal itself)

$\therefore$ Channel is not sufficient.

Compression ratio required for the compression algorithm = 62.9/20=3.195

7.2 Physical Layer

These describe the electrical and mechanical interface necessary to establish a communications path.

Layer 1 protocols are concerned with the physical and electrical interfaces. It defines for example:-

- Connection types and allocation of signals to pins
- Electrical characteristics of signals which includes bit synchronisation and identifying a signal element as a 0 or 1

Hence, layer 1 is responsible for transmitting and receiving the signals.

RS232/V.24

Signal voltage levels

- –3V to –25V binary 1 for data, OFF for a control signal
- +3V to +25V binary 0 for data, On for a control signal

25 Volts is the maximum rating for a line without a load. In practice RS232/V24 signals are set to typically be + –12V

Use of RS232/V.24 as DTE/DCE interface standard

Ground Signals

- Pin 1 (SHG) Protective Ground/Shield Ground to reduce external interference
- Pin 7 (SIG) Signal Ground provides a reference for other signals

Transmit and Receive

- Pin 2 (TxD) Transmit Data
- Pin 3 (RxD) Receive Data

Maintaining a Connection / 'Hardware Handshaking'

- Pin 6 (DSR) Data Set Ready, Modem indicates to DTE that it is ready, i.e., connected to a telephone wire
- Pin 20 (DTR) Data Terminal Ready, DTE uses this to prepare the modem to be connected to the telephone line. If it is placed in an OFF condition it causes the

modem to drop any connection in progress. Thus the DTE ultimately controls the connection.

'Hardware' Flow Control

- Pin 4 (RTS) Request to Send, Sent by DTE to modem to prepare it for transmission.
- Pin 5 (CTS) Clear to Send, Modem indicates to DTE that it is ready to transmit.
- Pin 8 (CD) Carrier Detect, Sent by modem to DTE, to inform it that a signal has been received from the other end of link.

Other

- Pin 22 (RI) Ring Indicator, sent by modem to DTE to inform it that a ringing signal has been received from the other end of the link. Used by auto-answer modems to wake-up the attached terminal.

7.2.1 Modulation Techniques

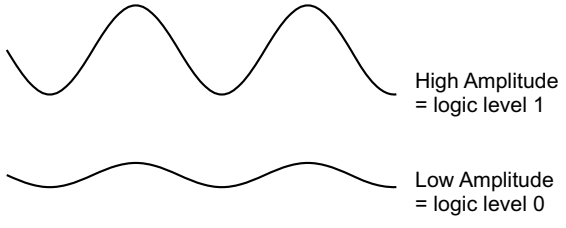
The Public Switch Telephone Network (PSTN) was designed for carrying analogue (i.e. voice) signals, not digital data. How can we transmit digital data over the PSTN? The solution to this problem is to *modulate* the digital information onto an analogue *carrier* signal. This is achieved by one of three main techniques -

(1) Amplitude Shift Keying (ASK)
(2) Frequency Shift Keying (FSK)
(3) Phase Shift Keying (PSK)

In order to connect a digital data source to a telephone line we use a piece of equipment known as a MODULATOR/DEMODULATOR or MODEM for short. The modulator part of a MODEM converts the digital data that is to be transmitted into a modulated analogue signal, the demodulator part accepts a modulated analogue signal off of the line and turns it back into digital data.

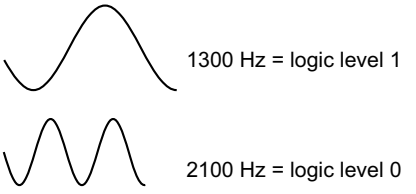
7.2.1.1 ASK

Amplitude shift keying uses a single carrier frequency, that is transmitted at two different amplitude (or volume) levels in order to represent a logic level 0 and a logic level 1.



7.2.1.2 FSK

Frequency Shift Keying uses two different frequencies to represent a logic level 0 and a logic level 1. For example, the V23 MODEM standard uses a signal of 1300 Hz to represent a 1 and a signal of 2100 Hz to represent a 0.



Full duplex operation is achieved by using two other frequencies (390 Hz and 450 Hz) for the other (or *back*) channel.

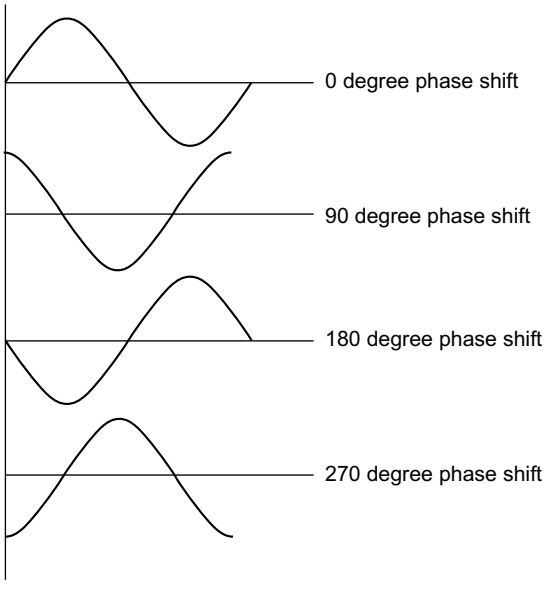
Note

Less susceptible to errors than ASK, used up to 1200 BPS on voice lines. Techniques is also used in high frequency radio transmission and in LANs

7.2.1.3 PSK

Phase shift keying uses a single carrier frequency for each channel (2 are required for full duplex operation). These are 1200 Hz and 2400 Hz. The logic levels are represented by phase changes in the signal.

Consider a system with 4 phases -



If we have four different states we can represent 2 bits with each signalling element. So we can define 0 degrees as 00, 90 degrees as 01, 180 degrees as 11 and 270 degrees as 10.

This now gives you a clue as to how we can obtain very high data rates (currently up 56600 bits per second) down a telephone line that was designed far at best 3 kHz of analogue data. The total number of data bits transmitted is double the number of signalling element changes on the telephone wire.

The total number of signalling element changes is known as the BAUD RATE, and this is bandwidth limited by the transmission medium. However if each signalling element represents N bits then the actual data rate is N * BAUD RATE.

7.2.1.4 Differential Phase Shift Keying (DPSK)

In order to design electronic circuitry that detects phase shifts, it is advantageous to ensure that lots of phase shifts occur even if the line is idle. A PSK system would just transmit a continuos tone in these circumstances, so the receiver clock will tend to drift. One solution is to use DPSK, which defines each pair of data bits (or dibits) as the phase change between two signalling elements.For example V22 defines the following coding system -

DIBIT VALUE	PHASE CHANGE
00	90 Degrees
01	0 Degrees
11	270 Degrees
10	180 Degrees

The carrier frequencies are again 1200 Hz and 2400 Hz, with a baud rate of 600. This means that the data rate is 1200 bite per second (BPS).

7.2.1.5 Quadrature Amplitude Modulation QAM

Higher data rates are achieved by a combination of PSK with ASK. So as well as changing the phase of the transmitted signal we can also alter its amplitude. The V22bis MODEM standard is the simplest example of this technique. V22bis defines 16 different types of signalling element, so each element represents 4 binary bits. The baud rate is still 600 baud, so the data rate achieved by V22bis is 2400 BPS.

Another examples is V32 which defines 16 states, transmitted as 2400 baud = 9600 BPS.

Higher data rates are now achievable (up to 56.6 kBPS) by increasing the number of discrete signalling elements available to the MODEM.

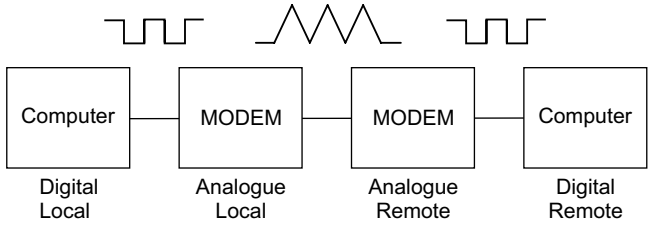
7.2.1.6 Modems

Sending Computer Data over Telephone Channels

- Computers produce digital data (pulses)
- Telephone channels are designed for analogue signals
- So digital data must be converted into a suitable format (analogue signals) if telephone channels are to be used
- Device which does this is called a MODEM

Modulator Demodulator

A MODEM can set up a switched path through the telephone network, or use a leased line.



7.2.2 Transmission Modes

There are two transmission modes to name **Asynchronous and Synchronous transmission.**

Fundamental difference between the two modes is:

Asynchronous Transmission – The receiver clock is not synchronised with respect to the received signal.

Synchronous Transmission – The receiver clock operates in synchronisation with the received signal.

For *both* types of transmission the receiver must be able to achieve *bit synchronisation.*

For Asynch transmission *byte* synchronisation must also be achieved.

For Synch transmission, synchronisation of a *block* of bits (or bytes) must also be achieved.

Asynchronous Transmission

Bit Synchronisation

- Transmitter must operate with the same characteristics as receiver
- Receiver clocks runs asynchronously with respect to the incoming signal
- Problem is to ensure the incoming signal (bit) is sampled as near centre as possible
- Local receiver clock runs at N times transmitted bit rate (typically x16 or x64)
- Each new signal is sampled after N ticks of the clock
- The higher the receiver clock rate, the closer to the centre the signal will be sampled

Character Synchronisation

- Each character is enveloped in start and stop bits.
- Transmitter and receiver must be programmed to operate with the same number of start and stop bits.
- Transmitter and receiver must be programmed to operate with the same number of bits for the transmitted character. This is typically 7 for ASCII, 5 for TELEX, or 8 for CEPT display profiles (e.g. teletext).
- When the line is idle, 1's are normally transmitted and the stop bits are also 1's.
- Start bit is usually a zero, thus there is *always* a 1-0 transition at the start or every character.
- Note the start bit is sample at N/2 clock ticks.

- Receiver can achieve character synchronisation simply by counting the number of bits in the character
- These are then transferred to a buffer.
- Next 1-0 transition indicates the start of the next character on the line.

Other Information

- Oldest, most common technique
- Application areas
  slow speed modems - up to 56.6 Kbps switched
  38.4 Kbps leased (over distance of 50 feet)
  interactive applications running on dumb terminals
- Transmitter and receiver must be configured to have the same characteristics:-
  5,6,7,8 data bits
  0,1 parity bits
  1 start bit
  1, 1.5, 2 stop bits
  These can be set by software, or alternatively can be set using hardware switches.
- Large overhead associated with asynchronous transmission i.e., start stop bits for every character therefore the true information rate is much less that the bit rate.
- Less reliable as bit rate increases.

7.2.2.1 Synchronous Transmission

Two variants of synchronous transmission

- Bit oriented – used by most modern protocols because it is more efficient
- Character oriented – older protocols

Bit and frame (block) synchronisation must be obtained

Frame Synchronisation

This relates to delimiting the frame, i.e. finding the start and end of the frame.

There are a number of ways in which this can be achieved. Three typical methods are :-

- Fixed length frames – used in ATM
- Carry frame length in fixed position in packet – used in Ethernet
- Use of FLAGS and Bit stuffing – as in X.25 and Frame Relay, which marks both the beginning and the end of the frame.

Flags and Bit Stuffing

- Data transmission entity is a frame.
- Frame is view as a string of bits.
- Typically a frame consists of many thousands of bits.
- The frame is encapsulated by two flags. Flags have bit value 01111110.
- Bit Stuffing is used to ensure the flag is not embedded in the frame, thus causing the end of the frame to be assumed – incorrectly. This process ensure that the encapsulated data is TRANSPARENT to the link level protocol.
- Bit stuffing is the process of automatically stuffing (adding) a 0 in the bit stream when 5 consecutive 1's are found.
- Thus, 6 consecutive 1's never appear in the frame contents, thus flag pattern (and end of frame) is never found in the frame contents.
- Normally 1's are transmitted when the line is idle.

Note that a detailed discussion of how frame synchronisation is achieved in character oriented protocols is not covered. However, the techniques are similar in principle to those used in Bit Stuffing.

7.2.2.2 Bit Synchronisation in Synchronous Transmission

There are two ways in which the receiver can obtain bit synchronisation:-

- Encoding the clock in the data
- Use of a digital phase lock loop circuit – with this scheme frequent transitions in the data are needed (i.e. frequent changes of binary zeros and ones)

7.2.2.3 Encoding Schemes

Manchester encoding

Data is encoded using two signal levels

- A binary 1 is encoded as a low-high signal
- A binary 0 is encoded as a high-low signal
- The transition (i.e. from high to low) always occurs at the centre of the bit
- The receiver uses this transition to sample the signal close to the centre of the second half
- So for a binary 1 which is low-high the signal will be high
- For a binary 0 which is high-low the signal will be low
- Bit is then added to the register
- Twice the bandwidth is needed for this scheme, so it is normally only used with LANs

Differential Manchester encoding

- A transition at the start of the bit only occurs if the next bit to be coded is a 0
- There is still a transition in the centre of each bit
- (Similar concept to differential PSK modulation technique)

7.2.2.4 Bit Synchronisation using Digital Phase Locked Loop (DPLL) Circuit

A DPLL is the electronic equivalent of a musical tuning fork. It resonates at a fixed frequency, which is used to sam-

ple the input data stream. The role of the DPLL is to ensure that the 'tuning fork' oscillates in phase with the arriving data. It therefore must see a transition (1 to 0 or 0 to 1) every now again in order to adjust itself back to the correct phase. When the output of the DPLL is in phase with the input data stream it is said to be 'in lock'.

There are two aspects to the synchronisation process

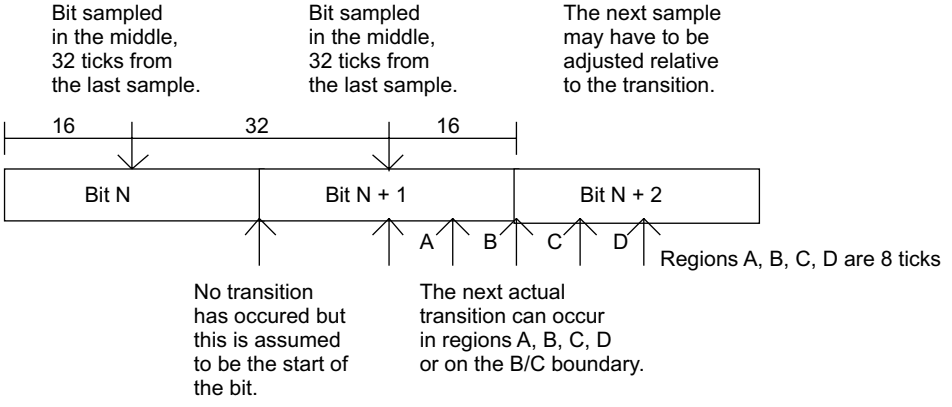
- Must obtain the same frequency as the transmitter
- Must sample in the middle of a bit

Obtaining the frequency of the transmitter

- Receiver can extract this from the signal, but it will drift unless transitions occur. How are transitions guaranteed?

Finding the middle of the bit.

- Receivers clock runs at a multiple of the transmitters clock (32 is a typical number)
- Pulse sample is adjusted to quickly find the middle
- Need occasional transitions to maintain synchronisation



7.2.3 Serial Communication

This is the barest possible means available in every PC to communicate with each other. IBM PC will be having two ports COM1: (address 0x3F8), COM2: (0x2F8). Normally this communication is used between computer and devices outside. It is evident that CPU does not communicate serially with such devices. Rather it communicate through UART interface which takes care of converting serial data to parallel data and vice versa. Prominently asynchronous and synchronous serial communication techniques are in wide use. For example modem is a good example for the first type.

Asynchronous Serial Communication

The connected devices do not have common clock, must synchronise their data transfer. This is carried out by making both sides to agree on some transmission parameters before data transfer. They are: 1. **Speed** (mentioned either in data rate in bits per second or baud rate/signaling rate) 2. No of data bits per transmission, 3. Whether uses parity (odd parity or even parity) or not. 4. No of stop bits and start bits.

Usually when the transmission line is idle its value is logic 1. Usually least significant bit of the data is transmitted first. After sending the last bit, parity bit will be send till that point receiver will be waiting. Usually 1, 1 1/2, 2 stop bits are used. The width of the start bit is same as any data bit.

N81 setting in which one stop/start bits with 8-bit data (including parity) is the one which is commonly used in many modem transmissions. Here percentage of overhead is 20.

Synchronous Serial Communication

Especially, this method is proposed to reduce the transmission overhead. This is achieved by sending a block of data rather then byte by byte. It does appends header and trailer information such as source address, destination address, checksum (for error detection), start/stop bit sequences and then the resulting data unit known frame is sent. Though this additional information also overhead but normally this is at lower percentage than asynchronous communication.

A common synchronous communication transmission standard, the High level Data Link Control (HDLC). Here total overhead bits are 48 bits. If we assume data is 256 bits then the overhead is 15.79%. Even if the data size grows the overhead percentage will be reduce, as overhead bits will not change from 48 bits. This is not true in the case of asynchronous serial communication.

When a byte or series of bytes (frame) is received or transmitted at constant time intervals with uniform phase differences, the communication can be called as synchronous. Bits of a data frame are sent in a fixed maximum time interval. Whereas in Iso-synchronous mode the maximum time interval can be varied. Two salient characteristics of this style of communication are:

1. Frames can not be sent at random intervals. Thus no need of handshaking.
2. A clock ticking at a certain rate has to be always there for transmitting serially the bits of all the frames.

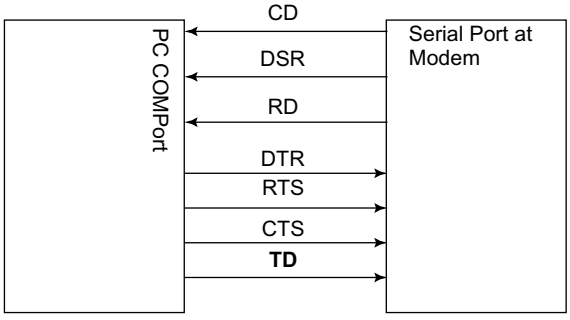
Universal Asynchronous Receiver/Transmitter (UART)

As mentioned above the CPU always communicates in parallel mode. Thus in order to communicate on serial lines UART's are used which takes the responsibility of converting to serial data stream to parallel stream and vice versa. A typical organisation of a UART is shown below.

The UART is the one which is responsible for generating start, stop, and parity bits as well as removing them. The processor can send control signals to UART to indicate speed, word size, parity, and no of parity bits.

UART employs Transmit Holding Register (THR), Transmit Shift Register (TSR). While one byte is transferred from CPU to THR, the one TSR is sent on the line. This is one form of double buffering employed to reduce delays.

Example devices using UART's are: keypad, mouse, modem, character input/output devices (terminals).



Serial Communication Standards (RS232-C)

At most only one device can be connected to this device unlike USB port. Serial ports of most of the PC's can transmit data up to 115,200bps. This standards supports nine signals given as:

RTS (Request to Send)
CTS (Clear to Send)
TD Transmit Data
DTR Data Terminal Ready
DSR Data Set Ready
RD Receive Data
CD Carrier Detect
RI Ring Indicator
G Ground

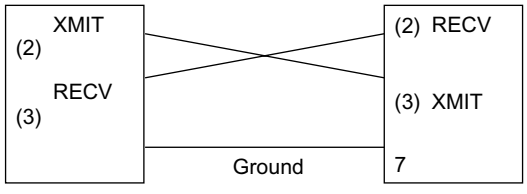
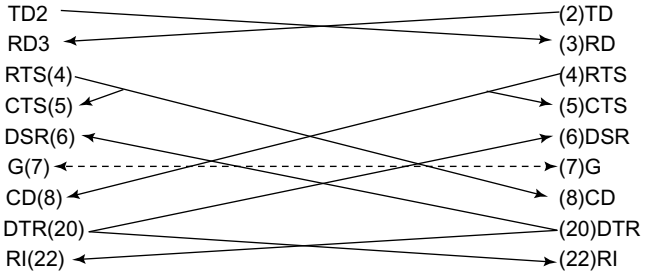


Figure 7.1

Usually RS232-C connector is a 25 pin D-connector (of course 9 pin D-connectors can be also used). At the barest level only 3 circuits are used as shown in Fig. 7.1 and resulting cabling is known as Null Modem.

A complete Null Modem cabling



A maximum separation of 15m at 9600bps is supported in this standard. This uses NRZ-L signalling.

Main drawbacks of RS232C is its limited distance of 15m and if the ground reference pin 7 is different for both sides then undesirable electrical disturbances will be applied to transmitted signal.

RS-449 calls for two sets of connectors: 37-pin for data, control, timing, diagnostics, and a 9-pin for secondary channel. Whereas RS232C has a single 25 pin connector. RS449 supports both balanced and unbalanced while RS232C supports only unbalanced. A balanced one is in which the signals are carried between the DTEs on a pair of wires. They are sent as a current down on one wire and return on the other; the two wires create complete circuit. In unbalanced one the signal is sent over a single wire with DTEs sharing a common ground. A balanced is less effected by noise.

RS422A and RS423A supports balanced and unbalanced, respectively. RS422A supports 100000bps for 1000m and 10000000bps for 10m whereas RS423A support 3000bps for 1000m and 300000bps for 10m.

The Universal Serial Bus Standards

Unlike RS232-C here we can connect at most 127 devices to an USB port. The data is transmitted in packets. The USB port is much faster than RS232-C port. USB version 1.1 supports 1.5Mbps (3m channel), 12Mbps for 25m channel, where as version 2 supports 480Mbps for 25m channel.

USB bus cable has four wires, one for +5V, two for twisted pairs and one for cable. There will be termination impedance at each end.

Serial signals are NRZI (non return to zero) type.

USB supports data transfer of 4 types given as:

1. controlled data transfer support guaranteed bus access
2. bulk data transfer support low priority large data transfer
3. interrupt driven support periodic bandwidth
4. iso-synchronous transfers which support guaranteed bandwidth

USB employs device polling. USB controllers are further classified as UHCI (universal Host controller interface), OHCI (open host controller interface). Communication overhead is more on host CPU in UHCI.

The USB standard specifies four types of packets such as token, data, handshake, and special.

The token packet is used to initiate data transfers. It specifies address (ADDR), direction specifier (packet identifier PID), end of packet (ENDP), and CRC checksum.

Here, data packets contains no addresses. Data field value can be upto 8192 bits.

Handshake packets are to either send ACK or NACK (negative ACK).

■ **Example** A computer sends 0.5KB (6144 bits) of data to one of its USB peripherals.

a. Show the packets sent by the computer to perform this transfer. What is the total no of bits transferred?

b. What percentage of the bits transmitted is overhead?

c. How many bits would be required to send the same data using an RS232C serial ort with no parity, 8 data bits and 1 stop bit is used?

■ **Answer:** According to the above discussion, a data packet in a USB standard can have at most 8192 bits. Our data is only 6144 bits. Thus, no of overhead bits = 8 + 16 = 24.

Percentage of overhead = 24/(6144 + 24) = 0.38%

In the case of RS232C for every 1 byte 1 start bit and 1 stop bit has to be sent. Thus overhead bits = 2*0.5*1024 =1024bits

Percentage of overhead = 1024/(6144 + 1024) = 14.3%

7.2.4 Line Sharing

A communication line can be shared using the multi-point configuration. For instance, terminals that wish to communicate with the master computer. In practice multiplexing is used to achieve this. Various Multiplexing methods are:

- Time division multiplexing – mainly in circuit switching.
- Statistical multiplexing – mainly on packet switching.
- Frequency Division Multiplexing – mainly on unguided transmissions
- Space division

7.2.4.1 Time Division Multiplexing (TDM)

Each TE has a separate connection to the TDM (Fig. 7.2). TDM samples each TE, in turn and puts aggregate onto high speed link. Each cycle (of servicing all TE's) has a fixed time period, and data from cycle is called a frame (time frame). This is similar to time-slicing in operating systems. Thus, the bandwidth allocated to a TE is fixed.

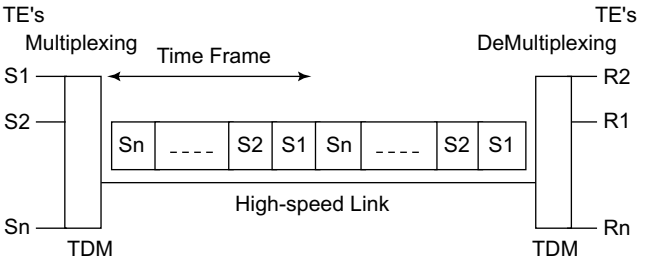


Figure 7.2 Time Division Multiplexer

In USA, TDM carriers are used to support PSTN lines as shown in Table 7.1. For example, T1 carrier supports 24 voice grade channels and a control channel of 8000bits(framing bits). Table 7.1 illustrates the carrier hierarchy. In Europe uses 31B+D is used (2.048Mbps) called as E1 lines there allow for 32 64k channels where 31 are voice and one D channel (64Kbps) is for signalling or framing. Also, ISDN carriers are also made available in the mean time. The 2B+D Basic Rate Interface uses 64k for each B-channel plus 16k for the D-channel, the 23B+D Primary Rate Interface uses 64k for each B-channel plus 64k for the D-channel, which equals 1.544 Mbps, T1 bandwidth.

Table 7.1

Carrier	Signal Level	# of T1 signals	# of Voice Channels	Speed
T1	DS-1	1	24	1544 kbps
T1c	DS-1c	2	48	3152 kbps
T2	DS-2	4	96	6312 kbps
T3	DS-3	28	672	44736 kbps
T4	DS-4	168	4032	274760 kbps

7.2.4.2 Statistical Multiplexing (Stat MUX)

Figure 7.3 shows a number of TEs connected to a Stat MUX. Note that typically there will be another Stat MUX connected to other end of the synchronous link, which also terminates a number of TEs. Before data can be accepted from a TE, the TE must inform the Stat MUX of the destination TE it wishes to communicate with. This information is conveyed to the Stat MUX at the other end of the synchronous link. Thus, at the destination, when a frame arrives from TE S1 (this is stored in the frame header), the Stat MUX routes it to the appropriate destination.

The Stat MUX collects data (characters – asynchronous transmission) from the TEs and builds variable length frames. The end of a frame will be recognised for instance

when carriage return is detected. Frames are moved to the stat mux's output buffer to await transmission. Mux will operate a particular layer 2 protocol for synchronous link (e.g., LAPB). With statistical multiplexing, only *active* TEs are serviced by the multiplexer. During transmission a frame will occupy the entire bandwidth of the high-speed link.

Sum of the average transmissions, for all TEs, must not exceed approximately 0.7 of the capacity of the high-speed link.

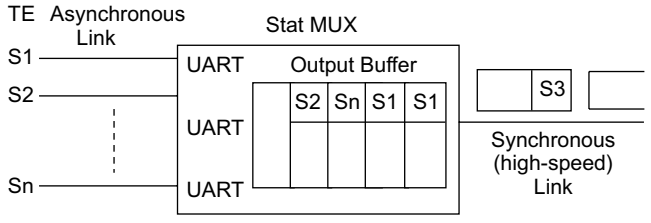


Figure 7.3 Statistical Division Multiplexer

7.2.4.3 Multiplexing – Comparison

Stat MUX (or concentrator) :

Frames are layer 2 frames and can be variable length.

Frames from DTE occupies entire bandwidth of high-speed link while it is being transmitted.

Therefore, each DTE can take a variable amount of bandwidth – bandwidth on demand.

Frames are stored on a buffer while they await transmission.

Sum of ***average*** transmissions for low-speed links must not exceed approximately 0.7 of the capacity of the high-speed link. Note, the sum of the average transmissions is *not* the same as the sum of the capacity.

TDM:-

Time Frames have a fixed length and have the same structure.

Time Frame contains a data sample from all TE's.

Bandwidth allocation is fixed.

Capacity of high-speed link is equal to sum of capacity of low-speed links.

7.2.4.4 Frequency Division Multiplexing

Unlike TDM and STATISTICAL Multiplexing, FDM is a technique that divides the available bandwidth into discrete frequency bands or *channels.* Each DTE is allocated their own channel, all of which can be used simultaneously. Data is put onto the communications media using a *radio frequency modem* which is tuned to that DTE's channel frequency.

7.2.5 Note on Multistage Crossbar Switches Design

Optimal design of a three stage crossbar switch :

Nx = 2N(2n-1) + 1(n-1)(N/n)^2

k = 2n–1

Nx = 4N(sqrt(2N) - 1)

N = no. of i/p lines

Nx = no. of crossing point

k = no. of crossbar switches in middle layer

n = no. of switches in first, last layer

7.3 Data Link Layer

The main function of layer 2 is to provide an error free link to layer 3.

Error Detection

Parity Bit

- Very simple scheme, where transmitter adds an extra parity bit to each character
- If even parity is used then the total number of 1's in the character (including the parity bit) must be even
- Can also have odd parity

Which error patterns can the parity scheme not detect? If two bits gets spoiled, this fails.

Block Check Sum

- This is an extension to the parity bit method
- There is a parity bit associated with each character as before
- String of characters are viewed as a vector 8 bits wide with a 'parity character' added to the end
- So there is a parity bit for each bit position in string of characters
- For a one character overhead, many more errors can be detected

Which error patterns the block check sum can not detect?

Cyclic Redundancy Check (CRC)

- CRC techniques used to produce the Frame Check Sequence (FCS) included in layer 2 frames
- Normally it is placed at the end of the frame
- CRCs can have varying lengths

  16 bits is the normal CRC in WANS

  32 bits is the normal CRC in LANs
- The greater the number of bits in the CRC, the greater the length of the frame that can be covered

7.3.1 Error Recovery (Correction) Classification

Error detection allows the receiver to detect transmission errors, for instance using the CRC. Having detected an error, the receiver must recover from it.

There are two mains types or error correction schemes

- Forward error correction
- Backward error correction

With forward error correction, extra information is generated by the transmitter and sent as part of the frame. The receiver uses the information to *detect* and *correct* the errors. The overhead (extra bits) associated with this form of recovery dictates that it is normally used only on channels with a high Bit Error Rate (BER) e.g., air interface on mobile cellular networks.

With backward error correction, the receiver asks the transmitter to *retransmit* the necessary frames. Currently backward error correction is the most common form of error correction.

7.3.1.1 Forward Error Correction (FEC)

Additional bits are added to the message which enable receiver to *detect* and *correct* errors. Thus there is an overhead associated with FEC.

The Hamming Single Bit Code is used to explain the principles of error correction. However, in practice, much more complicated encoding schemes, based on convolutional codes, are used.

Hamming Single Bit Code: Read First Unit
CRC: Do read example elsewhere in this Unit

7.3.1.2 Backward Error Correction

The principle of BEC is that the transmitter repeats frames which have got 'lost'. There are many different BEC protocols. These can be roughly classified according to the Repeat reQuest (RQ) strategy they operate. The three most widely used RQ strategies are:-

Idle RQ :
With Idle RQ the transmitter sends a frame and waits for an acknowledgement before sending the next frame. The transmitter remains idle until the acknowledgement arrives – hence the name.

Go-back-N Automatic Repeat reQuest (ARQ) :
The receiver identifies an error and requests that all outstanding frames are retransmitted, starting with the frame in error. It therefore, discards all frames that is has already received which were transmitted after the lost or corrupted frame.

Selective Repeat ARQ :
The receiver identifies an error and requests that *only* the frame in error is retransmitted.

Before these protocols can be fully understood the mechanics of sequence numbers and sliding windows must be mastered.

7.3.1.3 Sequence Numbers

- Errors can occur on the link, for example a frame gets corrupted by noise on the link.
- The receiver detects the error and requests another copy of the frame.
- Implication is both receiver and transmitter can *uniquely identify specific frames.*
- Frames are uniquely identified by sequence numbers.
- The sequence number must be carried in the frame - normally in the header.
- The obvious format is an integer.
- The link will carry large numbers of frames (perhaps thousands per minute) but it is not possible to reserve a large number of bits for the frame number. Remember, in packet switching a link is shared by a large number of calls.
- Thus a scheme is needed, whereby sequence numbers are reused after a period of time.
- The receiver controls when the sequence number can be reused by acknowledging correctly received frames.

For example, with a 3 bit sequence number we will have 8 sequence numbers – which is the sequence number space. The stream of sequence numbers allocated would be

0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,0

In summary, sequence numbers uniquely identify frames and allow the receiver to detect missing frames.

7.3.2 Idle RQ Protocols

Idle RQ is probably the simplest BEC protocol. The transmitter sends the first frame, frame 0 and waits for a positive acknowledgement. When the acknowledgement arrives it sends frame 1 and again waits for an acknowledgement. Eventually the sequence number will cycle back to 0 again.

If the receiver detects an error in a frame, then it discards the frame sends a negative acknowledgement. The transmitter responds to the negative acknowledgement by resending the frame. At any time, there is only one frame awaiting acknowledgement, therefore there is never any confusion over which frame must be retransmitted.

A frame may also get 'lost'. Clearly, in this case the receiver will not receive the frame, therefore an acknowledgement cannot be sent. There is a potential deadlock situation here. The transmitter cannot send a new frame until an acknowledgement is received and the receiver cannot send an acknowledgement until the frame is received. This is resolved by introducing a 'time-out'. When a transmitter sends a frame a clock is started. If the clock goes off before the acknowledgement is received, then the frame is automatically retransmitted again.

With Idle-RQ the link utilisation is poor. To increase the efficiency of the protocol Idle-RQ is extended to allow

the transmitter to have many frames outstanding. This is achieved by operating a sliding window (which is also called as pipelining) as follows.

7.3.3 Sliding Window Protocols

Related to sequence numbers is the concept of a sliding window. The following constants/variables are required to implement a sliding window

Constants Sequence number space
Window size
Variables : Upper window edge = V(S) S denotes Send
Lower window edge = V(A) A denotes Acknowledge

V(S) is the value of the next sequence number which will be allocated by the transmitter. It is *incremented* by 1 every time a frame is transmitted.

V(A) is *updated* by the receiver, when an acknowledgement is received from the other end of the link. Note, an acknowledgement may (and typically does) acknowledge multiple frames.

Together, these two variables control the *send* window.

V(R) is present in the receiver and is used to check whether the correct frame sequence has been received. The frame sequence number in the frame is compared with V(R). If they are equal, then this is the frame expected, and V(R) is incremented by one, ready for receipt of the next frame.

These variables always refer to the *next* frame. For instance, V(S) is the sequence number that will be allocated to the next frame that is transmitted. Similarly, V(R) in the receiver, is the sequence number of the frame that the receiver expects to receive next. The contents of the variable V(A) is the number of frame that will next be acknowledged. Also note that these are the variables required for data transmission in one direction and acknowledgements in the opposite direction. For a full duplex link (simultaneous transmissions in both directions) a set of these variables would be required in both transmitter and receiver.

How Sequence Numbers work

Initial state:- Nothing has been transmitted

Assume the sequence number space is 8 and the window size is 3

(Note that with 8 numbers the window could be bigger)

Remember the sequence numbers must wrap round from 7 back to 0

If V(A) and V(S) have the same value, then there are no *outstanding* acknowledgements.

The window size constrains the *relative* values of V(A) and V(S). With a window size of 3, *if* V(A) is 1 then V(S) *cannot* be greater than 4.

The numerical difference between V(S) and V(A) is the number of acknowledgements outstanding, i.e. frames which have been sent but not yet acknowledged.

Note that copies of the transmitted frames are stored in a buffer and only deleted when an acknowledgement is received. Thus, the window size also dictates the buffer size required at the transmitter.

7.3.3.1 Go-Back-N ARQ

With backward error control protocols, transmitters must store transmitted frames in a buffer until they have been acknowledged by the receiver. Acknowledgements may be piggy-backed on information frames going in the reverse direction (piggy-backing will be explained later) or alternatively separate explicit acknowledgements may be generated (Receiver Ready (RR) frames). A timer is used to ensure that acknowledgements are sent on time, otherwise the transmitter will time-out and automatically retransmit all unacknowledged frames. In the event of a transmission error, the receiver requests retransmission, beginning with a specified frame number and the transmitter responds accordingly. Clearly there must be no ambiguity in the frame number requested by the receiver and the frame number used by the transmitter. This has implications for the maximum window size, as demonstrated in the timing diagrams below.

Maximum window size for Go-back-N

In summary, with Go-back-N the *maximum* window size must be less than the sequence number space (N-1). Thus for a 2 bit sequence number, the sequence number space is 4 and the *maximum* window size is 3 (4-1). Note that a window size smaller than the maximum is also acceptable. The retransmission buffer size is dependant upon the maximum window size. The transmitter must be able to retransmit any of the unacknowledged frames therefore, the retransmission buffer size is also N-1.

How Go-back-N recovers from the following situations:

Corrupted Information frame

- Corrupted frame will be discarded by the receiver.
- Receiver will become aware of the *missing* I frame when the next I frame arrives – it will detect an *out of sequence* error (V(R) not = frame sequence number).
- Receiver sends an REJ frame, which requests retransmission.

(Note, if the corrupted I frame is also the last frame to be sent, then the transmitter will time-out because it will be expecting an acknowledgement from the receiver. On time-out the frame will be automatically retransmitted.)

Corrupted Receiver Ready (ACK) frame

- Transmitter *may* time-out and retransmit unacknowledged frames.
- However, if a subsequent RR frame is received in time, then the transmitter and receiver are oblivious to the *missing* acknowledgement.

Corrupted REJ

- On detecting an out of sequence error, the receiver will send an REJ frame to the transmitter.
- The REJ frame informs the transmitter what frame number the receiver is expecting and the transmitter begin retransmission from that point.
- Clearly all previous frames have been received, thus the REJ also acts as an acknowledgement.
- If the REJ is corrupted then the transmitter will eventually time-out and retransmit all unacknowledged frames.

7.3.3.2 Selective Repeat (sometimes called Selective Reject)

As the name implies, with Selective Reject, the destination requests retransmission of individual frames which have been corrupted. Retransmissions are requested using the frame type SREJ. In other respects however the protocol functions in much the same way as Go-back-N, including frame formats, control field formats, etc. The higher layer expects to receive Protocol Data Units (PDUs) in order, thus with Selective Reject, it is necessary to buffer PDUs at the destination while awaiting retransmission of earlier ones.

Using a similar approach as was taken for Go-back-N, it is possible to show that Selective Reject has the following characteristics:-

For a sequence space of 8 (3 bit sequence number)

1. The window size cannot exceed half the sequence space, thus for a 3 bit sequence number the window size must not exceed 4. This can lead to a situation known as sequence number starvation.
2. Clearly the transmit buffer is the same size as the window.
3. A buffer of the same size as the transmit buffer must also be implemented in the destination. This buffer is typically more complicated than the transmit buffer because PDUs may have to be reordered within the buffer.

7.3.3.3 Comparative performance of Go-back-N and Selective Reject

1. Throughput efficiency – Retransmissions scheme for Go-back-N leads to reduced throughput compared to Selective Reject. For Go-back-N, with a high offered load, this can lead to a situation known as congestion collapse.
2. Maximum window size – The maximum window size for Selective Reject is half the sequence number space compared to the sequence number space less one for Go-back-N, i.e. Selective Reject the window size is almost half that for Go-back-N, for the same sequence number space. Note that for both schemes, the window size dictates the transmit buffer size.
3. For Go-back-N the receive buffer in the destination is 1, while for Selective Reject it is the same as transmit buffer.

7.3.3.4 Performance LLC Protocols

Stop and Wait (ARQ)

Source station sends a frame and waits for its acknowledgment. Only after receiving the ACK next frame is sent. If it does not receive any ACK before time elapses, same frame will be retransmitted.

a = Tp / Tf (how many frames we can send with in propagation delay)
u = 1 / (1 + 2a) = Tf / (Tf + 2Tp)
u = 1 / Nr (1 + 2a)
Nr = expected number of retransmissions
u = (1 – p) / (1 + 2a) = utilisation
p = probability that a single frame is in error
Tf = frame time
Tp = propagation delay

General Sliding Window Protocol (pipelining)

Unlike ARQ, here a group of frames will be sent at a time for the duration equal to Tf+2Tp.

u = 1 if N > 2a + 1
u = N / (2a + 1) if N < 2a + 1
N= no of frames sent

Go back N

Receiver, will not send ACK if I'th frame is spoiled and at the same time it will not accept subsequent frames also even if they reach with out any error. At the source side for I'th frame timer gets elapsed and in the next transmission attempt I'th frame and subsequent frames and some more new frames are sent.

u = (1 – p) / (1 + 2*a*p) if N > 2a +1
u = N (1 – p) / (2a + 1) (1 - p + N * p) if N < 2a +1

Selective Repeat:

Here, NAK frames are sent by the destination when it recieves a spoiled or garbled frame. Receiver, during the next transmission those frames for which NAK frames are received are transmitted.

u = 1 – p if N > 2a +1
u = N (1 – p) / (2a + 1) if N < 2a + 1

* If N = 1 it turns to stop and wait protocol

■ **Example** A channel is employing stop and wait protocol with frame size of 10000 bits and at data rate of 100 Kbps. The propagation delay is 250 ms. The probability of a frame facing an error is 0.5. Then calculate the efficiency of the channel under this protocol.

■ **Answer :** Tp = 250 ms
Tf = 10000/100000 = 100 ms
a = Tp/Tf = 250/100 = 2.5
u = (1 – p) / (1 + 2a) = (1 – 0.5) / (1 + 2.5 *2)
= 0.0833

■ **Example** What should be the frame size for the above problem to have atleast 50% efficiency under same error conditions.

■ **Answer :** 0.5 = (1 – 0.5) / (1 + 2a)
a = 0

This situation may occur only if the propagation delay is almost 0 or the frame is of infinite size.

■ **Example** A sliding window protocol is employed on a satellite channel with a propagation delay of 250 ms and frame sizes are 1000 bits long and data rate of the channel is 100 Kbps. Then calculate the efficiency if we send 1 frame at a time, 7 frames at a time, 127 frames at a time.

■ **Answer :** U = N / 2a + 1
a = 250 / 10 = 25
(i) 1 < 25 so, U = 1 / (1 + 50)
(ii) N = 7 so, U = 7 / (1 + 50)
(iii) U = 1 since N > 2a +1

■ **Example** A channel has a bit rate of 4 Kbps and propagation delay of 20 ms. For what range of frame sizes does the stop and wait protocol gives an efficiency of at least 50%?

■ **Answer :** a = ?
0.5 = 1 / (1 + 2a)
a = 0.5
Tf = Tp / a = 40 ms
so, Frame size = 4000 * 40 * 10^{-3} = 160 bits

■ **Example** Frames of 1000 bits are sent over 1 Mbps satellite channel. 3 bit sequence no's are used. What is the channel utilisation for (i) Go back N (ii) Selective repeat

■ **Answer :** Go back N :
No of frames sent at a time = 7 (as 3 bit no's)
Tp = 250 ms
Tf = 1000 / 10^6 = 1ms
a = 250
so, U = N / 2a + 1 =7 / 500 +1
Selective Repeat :
No of frames sent at a time = $2^{(3-1)}$ = 4 (N/2 half of the frames)
so, U = 4 / 500 +1

■ **Example** Compute the fraction of bandwidth i.e, wasted on overhead (headers and retransmissions) for selective repeat on a heavily loaded 50 Kbps satellite channel with data frames consisting of 40 header, 3960 data bits. ACK frames never occur, NAK frames are 40 bits. The error rate for NAK frames are negligible. The sequence no's are 8 bits long

■ **Answer :** Selective Repeat:
No of frames sent at a time N = 2 ^ (8 –1) = 128
Overhead bits = 128 * 40 + 0.01 * 128 (40 + 4000)
so, Fraction of Bandwidth wasted = Overhead bits/50* 10^3

7.3.4 Local Area Networks (LAN's)

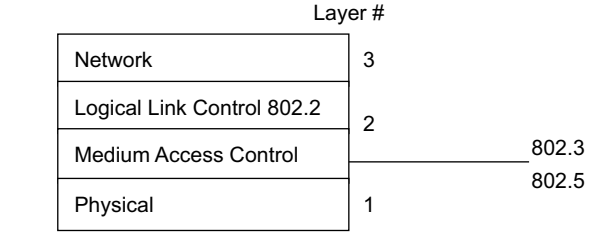
LAN protocols are also called as medium access control protocols. Media is shareable in LAN, everyone is equally sharable but there will be some control in accessing the shared channel.

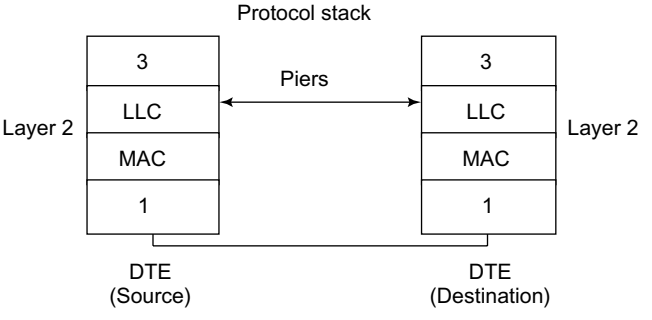
Issues

- MAC (Medium Access Control) layer (how nodes gain access to transmission media).
- Topology
- Transmission media
- Ownership
- Applications
- Speed

Media Access Control

	Deterministic	Probabilistic (Random access)
E.g.	TOKEN RING	ETHERNET
Standards	(IEEE 802.5)	(IEEE 802.3)





General Characteristics:

- End to end communications LLC to LLC.
- With bus and ring topologies, all DTEs share the same physical transmission media, i.e. they are all attached to it. All frames are transmitted on it.
- MAC layer is concerned with how they gain access to, and that they share transmission media in a fair way.

CSMA/CD (Carrier Sense, Multiple Access / Collision Detect)

- Used in technical / office environments.
- 10 Mbps baseband COAXIAL cable.
  10 base 2 - thin wire (0.25 diameter)
  maximum segment length : 200m
  10 base 5 - thick wire (0.5 diameter)
  maximum segment length : 500m
  10 base T - Hub (star) topology, ***but using twisted pair***
- It is possible to connect together with repeaters to extend the length to 2.5 kilometres.

Assumptions about the stations:

(i) carrier sensing
(ii) collision detection
(iii) time is continuous
(iv) employs binary exponential back off algorithm after collision
(v) if no collision is detected before sending last bit of frame, station concludes that frame is sent successfully
(vi) The channel is multiple access type i.e, more than one station can access the channel simultaneously

Frame transmission.

- Sending DTE encapsulates data in an MAC frame, with required source and destination addresses in the frame header.
- The frame is broadcast on media, the bits will propagate in both directions of the bus.

Frame reception.

- All DTEs are linked to the cable, and the DTE for whom the frame is destined, recognises the destination address and continues to read the rest of the frame until completion. Once the DTE realises the destination address is not its own it ignores the remainder of the transmission.
- The data part of the frame is sent up to the LLC (Logical Link Control) layer.

Frame Format

Preamble	7 Octets
Start of frame	1 Octet
Destination address	2 or 6 Octets
Source address	2 or 6 Octets
Length indicator	2 Octets
Data	<= 1500 Octets
Pad	Optional (Minimum field length)
FCS	4 Octets (32 bit CRC)

Preamble – allows MAC unit to achieve bit synchronisation (Manchester encoding used). It consists of 7 octets with the format 10101010 followed by a single 'start of frame' octet of the form 10101011.

Frame transmission in detail.

- DTE listens to transmission media, to decide whether another frame is being transmitted. (Carrier sense)
- If / when media is idle, DTE transmits frame, and simultaneously monitors media to ascertain whether another DTE has also transmitted a frame i.e., to ascertain whether a collision has occurred.

What are the implications of this ?
(If it listens while it transmits it must listen for the round trip delay time plus a small amount for error handling.)

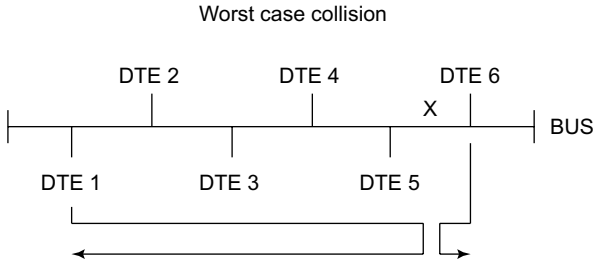


Figure 7.4

If a collision occurs at X, which is close to DTE 6 but some distance from DTE 1, DTE 1 will have to wait much longer than DTE 6 for the occurrence of the collision to reach it, this is shown by the arrows below the diagram in Fig. 7.4, the large arrow going both ways shows DTE 1 round trip delay, DTE 6 is much smaller.

- When a DTE which is transmitting a frame detects a collision, it reinforces it by transmitting (Immediately) a jam sequence, and discontinues transmitting its frame.
- (How many DTEs will detect a collision? As many as the amount that have decided the line is idle.)
- If a DTE detects a collision it attempts a (maximum) number of retransmissions before giving up.
- It retries after an integral number of slot times.
  Slot time = twice propagation delay of LAN + transmission delay of jam sequence.
- Number of slots : $0 \le R \le 2k$ k = number of retries R = number of slots

7.3.4.1 Important Points about Ethernet IEEE 802.3 LAN

In Ethernet Protocol, the min required frame time should be atleast 2Tp (Tp=end to end propagation delay). If the required data to be sent is very small then padding is done to make the frame bigger and satisfies 2Tp condition.

■ **Example** Calculate the min frame size required for a 10 mbps line of max length 2500 ms with 4 repeaters where the propagation delay is 25.8 micro sec. Also calculate the min frame size if the distance is reduced to 250 ms. Also calculate the min frame size if the data rate is 1 Gbps?

■ **Answer:**

2Tp = 2 * 25.8 51.6 micro sec

So, min frame size = $10 * 10^6 * 51.6 * 10^{-6}$ = 516 bits

= 64 bytes

case 1: Distance = 250 m

So, 2Tp = 5.12 micro sec

So, min frame size = $10 * 10^6 * 5.12 * 10^{-6}$ = 51.2 bytes

= 7 bytes

case 2: Distance = 2500 ms

data rate = 1 Gbps

2Tp = 51.2 micro sec

So, min frame size = $51.2 * 10^{-6} * 10^9$ = 51.2 * 1000

= 6400 bytes

Note

If the network spread increases, round trip propagation delay increases. consequently, min frame size also increases. Similarly if the data rate of the LAN increases, min frame size also increases.

IEEE 802.3 Frame structure

preamble : 7 bytes
start : 1 byte
source : 2 or 6
destn : 2 or 6
size : 2
data : 0 – 1500
padding : 0 – 49
checksum : 4 bytes

2 byte addresses are used for device control which are connected to systems such as mainframe computers

For LAN, 6 byte addresses is nothing but network card address

■ **Example** A LAN is having a propagation delay of 5.12 micro sec and is working at 100Mbps then calculate the min frame size and also calculate the worst possible number of padding bytes required?

■ **Answer:** Min frame size = $5.12 * 2 * 100 * 10^6 * 10^{-6}$

= 1024 bits

= 128 bytes

number of padding bytes required = 128 – (8+6+6+2+0+4) = 102 bytes

7.3.4.2 TOKEN Ring LAN

- Topology : Closed loop.
- Each station is attached to two other stations : 1 up stream - 1 down stream.
- It is actually a series of point to point links.
- All DTEs are connected together in a physical ring.
- Just like BUS based LAN's, all the frames share the same transmission path i.e. ring.
- One/A token (small frame, 24 bits long) circulates round the ring.
- The token is either available or in use.
- In order to send a frame a DTE must acquire the token.

A free token consists of 24 bits divided into 3 Octets. The first octet is the Start Delimeter (SD), the middle octet is the Access Control (AC) field, and the final octet is the End Delimeter (ED).

Manchester encoding is used at the physical layer. Violations of Manchester encoding are used to create the SD and ED octets.

Frame Transmission

- A DTE must first wait for the token to be available, to circulate round the ring to the DTE.
- It then changes the token from available to in use. (Set the T bit in Ac field to 1) It then transmits its MAC frame immediately after the AC field of the token, CRC computed, etc.
- The frame is repeated. i.e., each bit is received by all DTEs and retransmitted.
- Until it circulates back to the initiating DTE where it is removed, and the token is made available again, and passed on.

Frame Reception

- All DTEs receive frame, but only DTE whose address matches the destination address in the frame header keeps a copy of the frame.
- Receiving DTE updates Acknowledge (A) and Copy (C) fields in the frame trailer, this informs the originator that another DTE has received and copied the frame.
- The more DTEs on the ring the better the throughput, as the transmission line fairly used and there are no collisions, the only delay is once around the ring.

Other issues.

- Priorities can be implemented by setting the Reservation bits in an in-use tokens AC field..
- 1 of the DTEs must be set up as a monitor to ensure frames do not circulate continuously on the ring.
- Rings are usually wired such that if one DTE fails, then the others are still connected in a ring.

Slotted Rings.

- Fixed length frames circulate round the ring in slot frames. Most common slotted ring LAN is the Cambridge ring.

- Frame comprises : Source address (1 octet)
  Destination address (1 octet)
  Two data octets
  5 control bits
  Total of 32 bits.
- The ring operates in a similar way manner to token ring.
- Frame transmission : Sending station waits for an empty frame.
  The full / empty bit is set to 1.
  The source and destination addresses are copied onto the frame header.
  2 octets of data are copied to the data field.
  The frame is circulated onto the ring.
- Frame reception : At each station the frame is received and if the destination address matches the stations address, a copy of the frame is kept.
  The destination station sets two bits which indicate that :
  - The station is active.
  - The frame was accepted.
- The frame continues round the ring until it is received by the sender, which checks that the frame has been accepted, it then resets the full / empty bit.
- Note, a sending station is not allowed to have more than one frame in transit.
- A monitor station is required to manage the ring.

7.3.4.3 BUS vs Ring LAN

Distance

BUS - maximum bus length, 5 x 5 segments (each 500m) in CSMA/CD, (2.5Km)

Ring - maximum distance between stations 500m, but no overall limit on size.

Performance

BUS - low delay and high throughput when the offered load is low. As offered load is increased, collisions occur and the delay increases and throughput decreases.

Ring - no collisions. When the offered load is low, there is a delay in waiting for the token to arrive at the sending station. As the offered load increases, the delay does not increase substantially. Throughput is equal to offered load.

Robustness

BUS - more robust under failure. One station going down does not bring the network down. (If no ACK is received then it will not resend to that destination).

Ring - If the ring is broken then the network is down, thus more elaborate wiring systems are required to ensure this does not happen.

Acknowledgements

BUS - separate ACKs are required, thus extra traffic on the network.

Ring - ACKs from the destination to the source are piggy backed on the frame, thus no separate ACK's are required.

Simplicity

BUS - MAC protocol complex.

Ring - MAC protocol simple.

7.3.4.4 Some Important Points about Token Ring and Token Bus

Collision free protocols are used in Token ring, Token bus. Token Bus supports priority based service. The station which is holding the token is the one is eligible to use the channel while other stations keep quiet.

Thus while real data communication takes place collisions will not be there. However while ring adjustments takes place there will be collisions among the control frames which are reserved through binary exponential back-off algorithm.

The ring formed here is a logical ring. It is observed to be useful for assembly line operations.

1. Token Ring is made up of point to point links
2. Broadcasting behaviour is made possible in this physical ring with point to point links by allowing a frame sent by a station to rotate around the ring and letting it to reach the source station again where it will be removed

7.3.4.4.1 *Responsibilities of Monitor*

1. To take care of orphan frames or garbled frames. The monitor has to flush out the orphan frames such that the bandwidth is conserved.
2. Monitor station has to take the responsibility of token circulation. If a station holding a token is crashed along with a token then it is the responsibility of the monitor station to identify this and generate the token and enjoy the token and then hand over it to the next station.
3. It has to send special packet called active monitor present at regular interval such that no one will compete to become the monitor.
4. It has to send beacon frame to identify any breaks in the network.

Stations will be having interface units which works in two modes

(i) listening mode

(ii) promiscuous mode

In the listening mode whatever data is coming is sent to outgoing line without forwarding it to the computer connected to interface unit.

In the promiscuous mode, data will be sent to the computer and at the same time it is made available to the ring.

Token ring supports priority based service. Wire center technology is used in practice to take care of breaks in the ring.

Every frame contains special bits at the end, normally 0's when sent by source station, destination stations or its interface units will be setting these bits to indicate either of the following conditions

(i) frame is spoiled

(ii) frame is reached but destination station did not accept as it doesn't have buffer space

(iii) frame is reached but destination station is not working

(iv) frame is reached and successfully accepted

■ **Example** A 1 Km long, 10 Mbps CSMA/CD LAN has a propagation speed of 200 m per micro sec. Data frames are 256 bits including 32 bits of header, checksum, other overhead. The first bit slot after a successful transmission is reserved for the receiver to capture the channel to send a 32 bit ACK frame. What is the effective data rate, exclude the overhead? (Assume that there are no collisions)

■ **Answer:**

2Tp = 10 micro sec

Frame time = $256 / 10 * 10^6$ = 25.6 micro sec

ACK frame time = $32 / 10 * 10^6$ = 3.2 micro sec

In the worst case, efficiency = 25.6 / 38.8

■ **Example** Consider building a CSMA/CD network running at 1 Gbps over 1 Km cable with no repeaters. The signal speed in the cable is 2 lakh Km per sec. What is the min frame time?

■ **Answer:** 2Tp = 2 Km / 2 * 10 ^ 5 km /sec = 10 micro sec

Min data frame time = 10 micro sec

So, min frame size = $10 * 10^{-6} * 10^9$ = 10000 bits

■ **Example** A token bus system works like this, when a token arrives at a station, a timer is reset to 0. The station then begins sending priority 6 frames until the timer reaches a value T6 then it switches over to priority 4 frames and continues transmitting them until timer reaches a value T4. This is repeated for priority 2 and 0 also. If all the stations are having T6 to T0 values as 40,80,90 and 100 ms. Then what fraction of total bandwidth is reserved for each priority class?

■ **Answer:**

T6	40	40%
T4	80	40%
T2	90	10%
T0	100	10%

Token ring uses 3 byte tokens. A token contains special bit pattern and it is going on circulating in the ring if the stations are idle. Whenever a station wants to send the data, it is going to take over the the token. Really the token is designed such a way that it differs by 1 bit value of any frames first 3 bytes. Whenever, a station takes takes the token that bit will be reversed and then data bits will be appended to it.

In the design of the token ring, physical length of the bit is very crucial. For example, if the length of the ring is very less than monitor station has to set or insert extra delays such that token will be circulated in the ring.

Token Ring variants : IEEE 802.5, IBM Token ring, Cambridge ring

Most of the token rings work at data rates 4 Mbps and 16 Mbps

We can connect 250 m/c 's at most

4 Mbps one uses twisted pair and differential manchester encoding

7.3.4.4.2 *Token Insertion Policies*

Practically 2.5 bits delay will be there at i/f units

No of stations = M

Delay at each station = b bits long

Data rate = R bps

V = signal propagation speed

d = ring circumference

so, propagation delay (T) = d/v + Mb/R

Ring latency = T * R bits

Ist token reinsertion strategy is to reinsert the token after the frame transmission is completed but not until after the last bit of the frame returns to the sending station.

IEEE 802.5 uses this policy

For Example : M = 20

b = 2.5 bits

V = 2 * 10 ^ 8 m /sec

R = 4 Mbps

d = 20 * 100

T = ring latency = 90 bits

Let data frame = 400 bits (including overhead)

so, next token reinsertion is done at 490 bits

so, efficiency = 400/490 = 82 %

with the increase in Ring latency, efficiency falls down drastically.

2nd token reinsertion strategy is to reinsert the token after the frame transmission is completed but not until after the header of the frame returns to sending station.

This is used by IBM token ring.

■ **Example**
Ring latency = 90 bits
Data frame = 400 bits (including overhead)
Header bits = 15 bytes = 120 bits
Last bit of the header arrives at source station after 120 + 90 = 210 bits
As this 210 bits delay is less than 400 bits, soon after sending the last bit of the data frame token is reinserted.
So, efficiency is 100%
but, practically not 100% because of token walk time.

■ **Example**
Ring latency = 840 bits
Data frame = 400 bits (including overhead)
header bits = 120 bits
Last bit of the header arrives at source station after 120 + 840 = 960 bits
So, efficiency = 400/960 = 42%

The 3rd token reinsertion is strategy is to reinsert the token immediately after the frame transmission. Both IEEE 802.5 and IBM token ring with 16 Mbps uses this policy.

■ **Example** Ring latency = 840 bits
Data frame = 400 bits (including overhead)
So efficiency = 100% (theoretically)

Here, the efficiency can be said as highest or 100%. It will be little less than 100% because of the token walk time. However, this necessitates buffers at the stations.

3rd category of token reinsertion is called multi token operation whereas 2nd one is called single token operation and first one is called single packet operation.

The max throughput occurs if always every station has frames to transmit at any time (otherwise delays will occur)

7.3.5 Bridges

Bridge responsibilities are:

(i) Frame formatting
(ii) Priority (Adding ficticious priority and removing if necessary)
(iii) sending ACK's if necessary
(iv) Taking token and serving, it has to drain the frame as well

Bridge will be having only physical layer and data link layer
Bridge has to take care of responsibilities of differences between two LAN's
Hub's available currently works simultaneously as a repeater and also as a bridge
Bridges are the places where you can have filters and fire walls etc.
By combining LAN's you can have a MAN spread of few 10's of Km.
Distributed Queue Dual Bus (DQDB) protocol is used for MAN in many practical buses.

7.3.5.1 Types of Bridges

7.3.5.1.1 *Source Routing Bridges*

The source stations themselves writes via which route (bridge, LAN, bridge,....) the frame has to traverse. Simply the bridges has to transmit the incoming frame from one LAN to other LAN using this routing info. Here the source stations learns about the other stations of other LAN's and uses that info while routing the frames. Thus, the load on the stations are more whereas, the load on bridges is less. They have to do only packet reformatting etc.

7.3.5.1.2 *Transparent Bridges*

These are the one's which can be purchased off the shelf and simply used in our network. Here the source station simply sends the frame to the bridge for which its LAN is connected. It is the responsibility of the bridge to decide about the route inorder to acquire the knowledge about the stations, Bridges uses a special learning Alg. With the knowledge acquired through this alg about the routes, the frame is sent to the destn m/c. Here the load is more on the bridge rather than the m/c.

Spanning tree bridge is a variant of transparent bridge. If the computer network has cycles, then spanning tree is calculated and packets are sent so that the wastage of bandwidth is less.

Flooding : In flooding when a packet arrives at a bridge point or router point, the same will be transmitted on all lines except the line on which it arrived.

Guarantee of service is most important than the efficiency or robustness. It is achieved through sacrificing efficiency by having redundancy.

Every packet will be having record route and every LAN and bridge write ID into that frame while traversing. This info is used by source. If circular then it will be rotating.so,it will keep info of no of hops = the no of bridges and when the count becomes 0 then that will be killed.

Multiport bridges are used to connect more than two LAN's.

7.4 Network Layer

There are two variations of packet switching

- Virtual Circuit
- Datagram

Virtual circuit (Connection Oriented Network Service, CONS)

Usual 3 phases :- Call set up
Data transfer
Disconnect

At call set-up, a Call Request packet is built by the source TE and transmitted into the network. A logical channel number is allocated at each node along the route and this information is stored in the switch table (refer to diagram overleaf). The logical channel number uniquely identifies the call, and has local significance only. The routing information, i.e. the output link which the packets should be switched to, is supplied by the routing algorithm. When the call has been established, data transfer can take place. The logical channel number is carried in the header of every packet. The node accesses the switch table and translates the input logical channel number and link number to the output logical channel number and link number. Thus the packet is switched to the appropriate output link and clearly, *all data packets follow the same route*.

When all the data has been transferred the call can be cleared.

Datagram (Connectionless Network Service, CLNS)

With datagram networks, there is no call set-up, therefore there is no call disconnect. Packets are routed independently, thus they must carry the full destination address in the packet header. Packets can therefore arrive out of order, so it usual for the destination to operate a Selective Reject error control scheme. No flow control can be applied in the network therefore the nodes are more vulnerable to congestion. However, depending on the routing algorithm in operation, it is possible to route around congested areas of the network and network failures. Datagram networks are particularly well suited to calls which transmit only a very small amount of data. Because packets are routed independently they can potentially follow a different route in the network.

IP Addressses

Class	Address Range	Supports
Class A	1.0.0.1 to 126.255.255.254	Supports 16 million hosts on each of 127 networks.
Class B	128.1.0.1 to 191.255.255.254	Supports 65,000 hosts on each of 16,000 networks.
Class C	192.0.1.1 to 223.255.254.254	Supports 254 hosts on each of 2 million networks.
Class D	224.0.0.0 to 239.255.255.255	Reserved for multicast groups.
Class E	240.0.0.0 to 254.255.255.254	Reserved for future use, or Research and Development Purposes.

Ranges 127.x.x.x are reserved for the loopback or localhost, for example,**127.0.0.1** is the common loopback address. Range **255.255.255.255 broadcast** to all hosts on the local network.

■ **Example:** A router has the following (CIDR) entries in its routing table:

Address/Mask	Next Hop
135.46.56.0/22	Interface 0
135.46.60.0/22	Interface 1
192.53.40.0/23	Router 1
Default	Router 2

For each of the following IP addresses, what does the router do if a packet with that address arrives?

■ **Answer:**

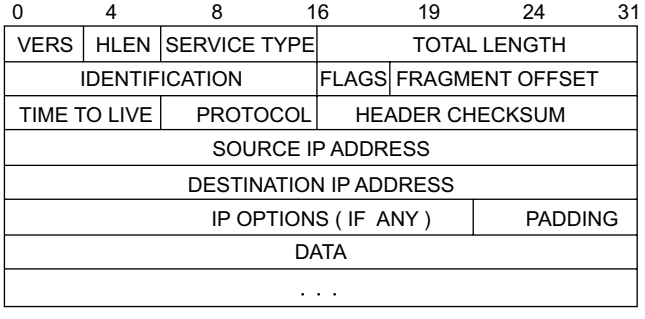
A. 135.46.63.10 – Interface 1
B. 135.46.57.14 – Interface 0
C. 135.46.52.2 – Router 2
D. 192.53.40.7 – Router 1
E. 192.53.56.7 – Router 2

■ **Example** IPv6 uses 16-byte addresses. If a block of 1 million addresses is allocated every picosecond, how long will the addresses last?

■ **Answer:** It would take approximately 10,790,283,070,806 years to allocate all of the available addresses.

7.4.1 Internet Protocol (offers a Connectionless Service)

The Internet Protocol (IP) is the protocol used in the Internet. The Internet comprises a large number of interconnected networks. The computers attached to the networks are called *hosts* (hosts are the equivalent of DTEs in X.25) and the devices used to interconnect the networks are called *gateways* (gateways are the equivalent of X.25 switches). Routing is the main function carried out by a gateway. The basic unit of transfer is a datagram (PDU). Similar to other protocols, an IP datagram (PDU) comprises a header followed by data. The IP datagram format is shown below. The role of the various fields which make up the header are as follows:



VERS (4 bits) – This contains the version of IP that was used to create the datagram. It essentially defines the format of the datagram and is required to ensure all network

components which process the datagram apply the same format. The latest version is 4, but version 6 is currently being defined.

HLEN (4 bits) – Defines the length of the header in 32 bit words.

SERVICE TYPE – Defines how the datagram should be processed. This comprises a number of sub fields which are shown below.

PRECEDENCE (or priority) (3 bits) – Defines the priority of the datagram, from 0 (normal) through to 7 (highest). Generally, this is ignored, but it will become more important in the provision of QoS.

D – Is a one bit flag which when set requests low delay.

T – Is a one bit flag which when set specifies high throughput.

R – Is a one bit flag which when set specifies high reliability.

Note that it is not possible for the network to guarantee the service requests that have been made, however it is clearly important for the network to at least know the user requirements.

The last two bits of the service type field are unused.

TOTAL LENGTH (16 bits) – Defines the total length of the datagram measured in octets.

IDENTIFICATION, FLAGS and FRAGMENT fields (total 32 bits) control the fragmentation and reassembly of datagrams. This is discussed later in this section.

TIME TO LIVE (8 bits) – This is measured in seconds, and defines how long the datagram is allowed to remain in the Internet. This field is decremented as the datagram is moves through the network. When the field reaches zero the datagram is discarded and a message sent back to the source.

PROTOCOL (8 bits) – Defines the high-level protocol that was used to create the message being carried in the data. This essentially defines the format of the data portion of the datagram.

HEADER CHECKSUM (16 bits) – This is a checking field used to check the integrity of the header.

SOURCE IP ADDRESS and DESTINATION IP ADDRESS (32 bits each) – Define the source and destination IP address.

IP OPTIONS – The length of this field is variable, depending on which options are chosen. Options are stored contiguously in the OPTIONS field, and each option comprises an OPTION CODE field followed by an optional LENGTH (8 bits) and DATA field (variable integral number of 8 bits). Options relate to network management and control. For instance, if the record route option is specified then each network element which processes the datagram must add their IP address to the record route option field.

A detailed description of all options is beyond the scope of these notes.

The major functions carried out by IP are Fragmentation/Reassembly, Routing and Error Reporting.

Fragmentation/Reassembly

Datagrams may be transported across many physical networks. There may be various maximum physical frame sizes associated with these physical networks. Consequently if a datagram is longer that the maximum physical frame size defined by the network it must be fragmented into a number of smaller fragments accordingly. Each of the fragments becomes a new datagram and most (not all) of the header fields will be copied from the original datagram to each new fragment.

Of great importance in the fragmentation process is the IDENTIFICATION field. This must be copied into each fragment because this identifies the original datagram to which the fragment belongs. The FRAGMENT OFFSET field specifies where (the offset) this fragment was positioned in the original datagram. The last fragment resets the MORE FRAGMENTS bits. From the FRAGMENT OFFSET and TOTAL LENGTH fields in the last fragment the destination can calculate the length of the original datagram. It is a simple task to reassemble the datagram.

Routing

Before routing can be discussed, addressing must be understood. An IP address is 32 bits long, and comprises a netid (network identifier) field and a hostid (host identifier) field. Each network must be allocated a unique netid within the domain of the Internet, while each host must be allocated a unique hostid within the domain of the network to which it is attached. In this way a (netid, hostid) pair can uniquely identify a host, and thus it is possible to route a datagram from a source host to a destination host without any ambiguity. Note that the hostid is typically divided into subnetid and hostid. This subdivision allows the network identified by the netid to be considered as a 'local internet' comprising networks identified by the subnetid and hosts attached to the subnetwork identified by hostid.

There are three different address classes, Class A, B and C. These differ only in the number of bits allocated to the netid and the hostid. For instance, Class A addresses allocate 7 bits for the netid and 24 bits for the hostid, while Class C allocates 21 bits for the netid and 8 bits for the hostid. Clearly Class A addresses allow fewer networks but a greater number of hosts on each of the networks compared to Class C. The format of these address classes are shown in Fig. 7.6. To make it easier to read the address, the 32 bits are broken down into 4 bytes, the byte values converted to decimal and separated by dots (periods). Also, in quoting network addresses the hostid field is set to zero. For example, the netid for DMU is 146.227.0.0

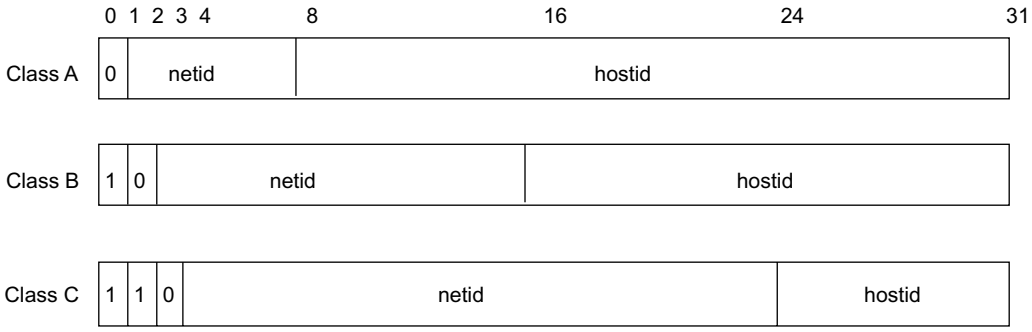


Figure 7.6 Address Classes

To explain the process of routing, consider the example of internet as shown in Fig. 7.7. The internet comprises 5 networks connected together by 3 gateways to form an internet. Each of the networks have a unique netid which is shown in the diagram and each gateway has a (netid,hostid) address for every network to which it is attached. For instance gateway G2 is connected to three networks. Every gateway must maintain a routing table. Typically on a local internet the routing tables are maintained manually by a network administrator who must update them as the network topology changes. In contrast, in the Internet an adaptive distributed routing algorithm maintains the routing tables automatically. Each entry in the routing table gives a netid and the address of the next gateway in the route to that netid. Because the routing process is carried out one hop at a time, this is commonly called hop by hop routing. The routing process carried out by each gateway can be summarised as follows:-

A datagram arrives at a gateway

The destination address is extracted

The routing table is searched using the netid part of the address

If the gateway is attached directly to the network netid

the *destination* address is converted to a physical address

the datagram is encapsulated in a physical frame

the *destination* physical address is included in the physical frame

the frame is transmitted on the network

else

the next gateway address is extracted from the routing table

the *gateway* address is converted to a physical address

the datagram is encapsulated in a physical frame

the *gateway* physical address is included in the physical frame

the frame is transmitted on the appropriate network

Given is the example of network:-

Compile a routing table for G2 and explain how G2 will route datagrams arriving for hosts. Assign network identifiers to the networks.

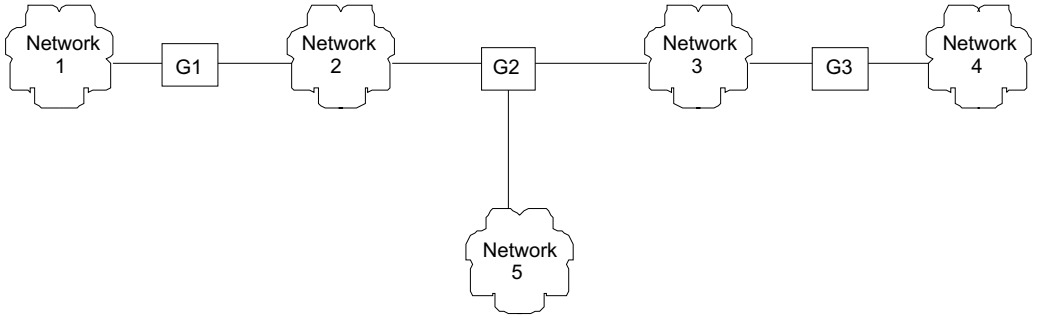


Figure 7.7 Process of Routing

Error Reporting

A datagram service is a best effort service. If a host or gateway discards a datagram for reasons other than transmission errors then an ICMP (Internet Control Message Protocol) is sent back to the source host. This is termed error reporting. The ICMP gives the reason for discard, for instance the destination host is unreachable.

The ICMP is also used to manage congestion. If a datagram is discarded because the buffers are full then a 'source quench' ICMP message is return to the host. The

host is expected to respond to the message by reducing the traffic rate.

There are other functions carried out by the ICMP which are outside the scope of these notes.

7.4.1.1 Some Points about TCP / IP and UDP

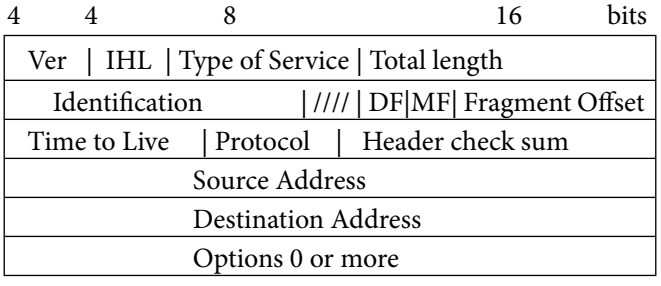
- IP packet is normally called as Datagram
- For connection oriented service – TCP is used
- For connection less service – UDP is used
- Network address length depends on the class type whereas network card address is independent of the class of the network and is always 48-bit long
- Port number, IP number can be combinedly called as socket. But however it is not too valid.
- It's more appropriate to call socket as a file rather than an address. In UNIX O.S it is visualized as a file. Thus, some of the Unix system calls such as read ( ). Write ( ) can be used on the sockets.

* Proxy ARP : An m/c will be responding on behalf of a group of IP addresses i.e, this m/c will respond to ARP message by sending its N/W card address for a group of m/c 's. Thus, the router will be delivering the IP packets to this ARP which may take further action to deliver the packets to the real host.

- DHCP (Dynamic Host Configuration Protocol) :

  DHCP protocol is a variant of bootp protocol, with the help of which A host which is physically moved from one LAN in a cluster is automatically Identified.

IP Packet Format:



In the packet structure shown, we have the MF bit. This is usually to indicate whether this packet is the last fragment or not. All fragments except the last one will have this bit set.

DF bit is used to indicate whether this IP packet can be fragmented or not all m/c's are required to accept this packet as a whole if this bit is set. However, if it is not possible then an ICMP error message is sent to the host.

IHL denote the Internet header length in multiples of 32-bit words.

The version field indicates the the IP protocol version is it 4 or not.

The Header total size thus should be in between 20 and 60 bytes. The type of service specifies, what type of service we wanted, whether we want service with lowest error rate or with min delay or max packet size.

The total length includes the total length of the packet including the header size and data size. As of today, the max allowable packet size is 65535 bytes.

The identification field contains the details about packet seq. no., fragment no. etc.

Fragment offset indicates the offset of the fragment w.r.t original packet in multiples of 8 bytes.

Time to Live field is used to indicate how long the packet can live in the internet. Typically its value is 255 sec. Whenever it reaches router then it gets reduced and when this becomes 0, this packet is recognized as Stalled Packet.

CheckSum is 16 bit and is calculated only for the header of the IP packet. Entire header value is taken as 16 bit and by doing 1's complement of each 16-bit word first each half word of the header are added together using 1's complement arithmetic and then 1's complement is taken for the result. Initially the header checksum is assumed to be all 0's while calculating.

The Options field is used whenever we wanted either of the following services:

(i) Strict source routing
(ii) Loose source routing
(iii) Record route routing
(iv) Time stamp routing

7.4.1.2 Distance Vector Routing Protocols

Most routing protocols fall into one of two classes: distance vector or link state. The name distance vector is derived from the fact that routes are advertised as vectors of (distance, direction), where distance is defined in terms of a metric and direction is defined in terms of the next-hop router. For example, "Destination A is a distance of five hops away, in the direction of next-hop Router X." As that statement implies, each router learns routes from its neighboring routers' perspectives and then advertises the routes from its own perspective. Because each router depends on its neighbors for information, which the neighbors in turn might have learned from their neighbors, and so on, distance vector routing is sometimes facetiously referred to as "routing by rumor."

A typical distance vector routing protocol uses a routing algorithm in which routers periodically send routing updates to all neighbors by broadcasting their entire route tables.

Periodic updates means that at the end of a certain time period, updates will be transmitted. This period typically ranges from 10 seconds for AppleTalk's RTMP to 90 seconds for the Cisco IGRP.

When a router first becomes active on a network, how does it find other routers and how does it announce its own presence? Several methods are available. The simplest is to send the updates to the broadcast address (in the case of IP, 255.255.255.255). Neighbouring routers speaking the same routing protocol will hear the broadcasts and take appropriate action. Hosts and other devices uninterested in the routing updates will simply drop the packets.

Most distance vector routing protocols take the very simple approach of telling their neighbours everything they know by broadcasting their entire route table, with some exceptions that are covered in following sections. Neighbours receiving these updates glean the information they need and discard everything else.

Figure 7.8 shows a distance vector algorithm in action. In this example, the metric is hop count. At time t_0, Routers A through D have just become active. Looking at the route tables across the top row, at t_0 the only information any of the four routers has is its own directly connected networks. The tables identify these networks and indicate that they are directly connected by having no next-hop router and by having a hop count of 0. Each of the four routers will broadcast this information on all links.

At time t_1, the first updates have been received and processed by the routers. Look at Router A's table at t_1. Router B's update to Router A said that Router B can reach networks 10.1.2.0 and 10.1.3.0, both zero hops away. If the networks are zero hops from B, they must be one hop from A. Router A incremented the hop count by one and then examined its route table. It already recognised 10.1.2.0, and the hop count (zero) was less than the hop count B advertised, (one), so A disregarded that information.

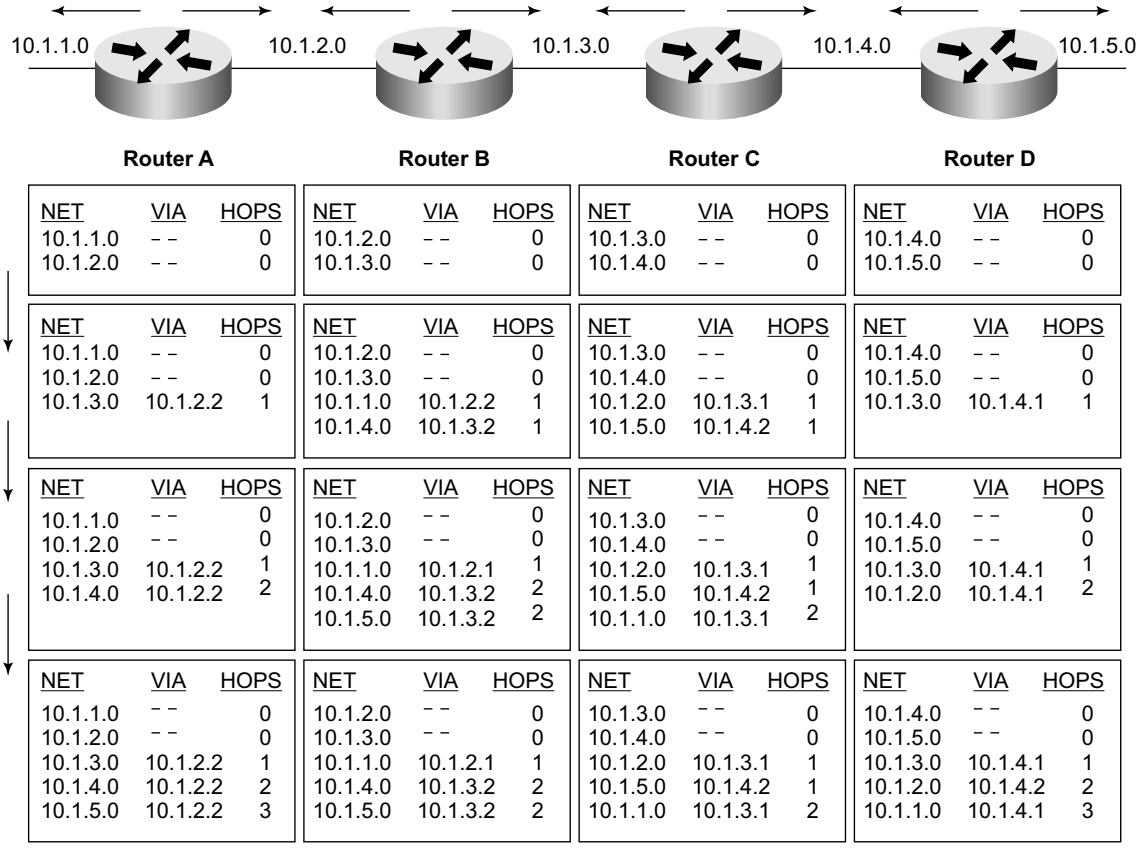


Figure 7.8 Distance vector algorithm in action

Network 10.1.3.0 was new information, however, so A entered this in the route table. The source address of the update packet was Router B's interface (10.1.2.2) so that information is entered along with the calculated hop count.

Notice that the other routers performed similar operations at the same time t_1. Router C, for instance, disregarded the information about 10.1.3.0 from B and 10.1.4.0 from C but entered information about 10.1.2.0, reachable via B's interface address 10.1.3.1, and 10.1.5.0, reachable via C's interface 10.1.4.2. Both networks were calculated as one hop away.

At time t_2, the update period has again expired and another set of updates has been broadcast. Router B sent its latest table; Router A again incremented B's advertised hop counts by one and compared. The information about 10.1.2.0 is again discarded for the same reason as before.

10.1.3.0 is already known, and the hop count hasn't changed, so that information is also discarded. 10.1.4.0 is new information and is entered into the route table.

The network is converged at time t_3. Every router recognizes every network, the address of the next-hop router for every network, and the distance in hops to every network.

Route Invalidation Timers

How will it handle re-convergence when some part of the topology changes? If a network goes down, the answer is simple enough other neighbouring router, in its next scheduled update, flags the network as unreachable and passes the information along.

Split Horizon

A route pointing back to the router from which packets were received is called a reverse route. Split horizon is a technique for preventing reverse routes between two routers.

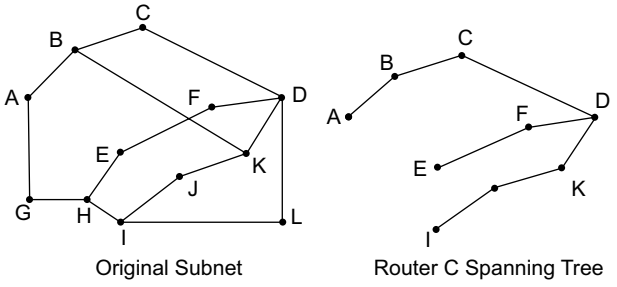
■ **Example**

Consider a subnet with routers A,B,C,D,E. Distance vector routing is used, and the following vectors have just come in to router C: from B: (5,0,8,12,6,2); from D: (16,12,6,0,9,10); and from E: (7,6,3,9,0,4). The measured delays from router C to B, D, and E, are 6, 3, and 5, respectively. What is C's new routing table? Give both the outgoing line to use and the expected delay.

The new routing table would be listed as follows:

To	Outgoing Line	Expected Delay
A	B	11
B	B	6
C	C	0
D	D	3
E	E	5
F	B	8

■ **Example**



Original Subnet Router C Spanning Tree

Suppose that node B in above has just rebooted and has no routing information in its tables. It suddenly needs a route to H. It sends out broadcasts with TTL set to 1, 2, 3, and so on. How many rounds does it take to find a route?

■ **Answer:** It will take 3 rounds of broadcasts to locate a route to H.

7.5 Transport Layer

Main responsibilities of the transport layer is to extend services to the Application layer from which Byte stream is available to the Transport layer.

Socket can be considered as Transport Layer Service Access Point.

(i) Connection establishment, management, closing etc is also the responsibility of the transport layer

(ii) Extending the Quality of service

(iii) Multiplexing of data coming from different applications

(iv) Error control also

Transport Layer extends connection oriented, reliable full duplex service to the application layer.

Binder is a kernel component which registers Process id no's and the Port which it is using. When a packet arrives with a given port no, based upon this port no Binder decides to which process this packet has to be handovered.

Daemon Programs when they are initiated, they will be first creating a socket with a port (well known port) and then they will be waiting for the connection request to this port address. Thus, this port is called as listening port. Every well known service such as Telnet, ftp etc will be extending their services at a fixed port only.

When a Client request arrives to the port at which the server is listening then the server creates a new process. In that process, a new socket is created with different Port no. whose value is more than 1023 and that socket and the client side socket are binded together.

In TCP connection establishment, Client can send connection request and server replies with a special packet called SYN packet and then the Client replies again with an acknowledgment for the same. Since then, both sides knows connection is established.

Sliding Window Protocol is employed while data transfer takes place between two TCP entities. The sender side will be sending a packet with its sequence number. The receiver side TCP entity replies along with acknowledgement, amount of free buffer space available with it.

Both Nagal's algorithm and Silly window syndrome are complementary to each other.

Connection can be closed by anyone, however otherside connection may be still active thus, the one which did not close can continue to use its socket to send the data.

TCP Segment Structure :

<table>
<tr><td colspan="7">Source Port</td><td>Destination Port</td></tr>
<tr><td colspan="8">Sequence no</td></tr>
<tr><td colspan="8">Acknowledge no</td></tr>
<tr><td>HL</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Window size</td></tr>
<tr><td colspan="7">Check sum</td><td>Urgent pointer</td></tr>
<tr><td colspan="8">Options (0 or more 32 – bit words)</td></tr>
<tr><td colspan="8">Data (Optional)</td></tr>
</table>

Source Port and destn port are 16 bit long which conveys from which port this segment is arrived from the source m/c and destn port indicates to which port of the destn m/c this segment has to be delivered.

The sequence numbers and acknowledgements means they are in normal sense only.

As, the TCP extends the byte oriented service, usually the sequence number indicates the starting byte number w.r.t the buffer or message as a whole. Whereas the Ack number indicates the next byte expected.

The Header length is 4 bits long, it conveys how many 32-bit words, the header is having including the options field.

URG bit indicates whether the urgent pointer is in use or not.

If the URG bit is set then the urgent pointer is meaningful. This urgent pointer indicates byte offset from the current sequence number at which urgent data has to be found.

Ack bit is set then the acknowlwdgement field is valid.

If PSH bit (Pushed data) is set then the receiver is requested to deliver the data to the application prog without buffering. However, most of the today's TCP's implementations will be ignoring this bit.

RST bit is used to reset a connection because of post-crash or some other problems.

SYN bit is used to establish the connection. However, the same is used in connection reply also.

FIN (Finish bit) is used to release a connection. It specifies that the sender has no more data to transmit. However, after closing a connection a process may continue to receive data indefinitely.

Window size: This field indicates how many bytes may be sent, starting at the byte acknowledged.

Checksum here is calculated for header and data and calculation is as usual taking 1's complement addition and complementing the final.

Options are available such as to specify the max TCP load.

7.5.1 Gateways

The networks which differ at transport layer level or application layer level are connected through special junction boxes called as gateways.

For example, application gateways are very much employed to country policies such as Max allowed message size, security related key sizes etc.

While internets (WAN) of two countries are supposed to be connected then very commonly application gateways are employed. However, if we employ conventional gateways then common management is required on it, which is not recommended always.

Thus half gateways are employed both sides in which both sides will agree a common protocol and the data packets are converted into this common standards and thus the problems are less.

UDP (User Datagram Protocol):

User's datagram protocol is a connectionless service. Here the Client m/c's can send the IP datagrams without having connection established. Those applications which doesn't require persistent service for a client can employ UDP based design such that delays for connection establishment can be reduced.

Services such as time of the day service etc can be extended through a UDP service.

Whenever a Client m/c wants a centralised timer value, it can simply send a request without establishing the connection.

UDP segment contains 8-byte header followed by the data, the header field contains Source port, destn port, UDP length, UDP check sum is calculated for the data and the header.

7.5.2 Domain Name System (DNS)

1. Domain name of a server is easy to remember by human, but computers still need IP address for communication. Therefore, it is necessary to translate domain name into IP address. It is done by the Domain Name System (DNS). *Example:* ***www.ust.hk → 143.89.14.34***
2. DNS Server, or name server, stores a *database of DNS records*. It also caches DNS responses of recent lookups for a time specified by the time-to-live (TTL) value of the DNS record). DNS servers communicate with each other from time to time to keep the records up-to-date.

3. Example of how DNS works:
   - User wants to open a web site at server **www.ust.hk** (HKUST's web server)
   - The PC makes a request to the DNS server for the required IP address.
   - If DNS record is found, the DNS server will give a response of IP address to the PC.
   - At last, the PC can use the IP address to contact the remote host.

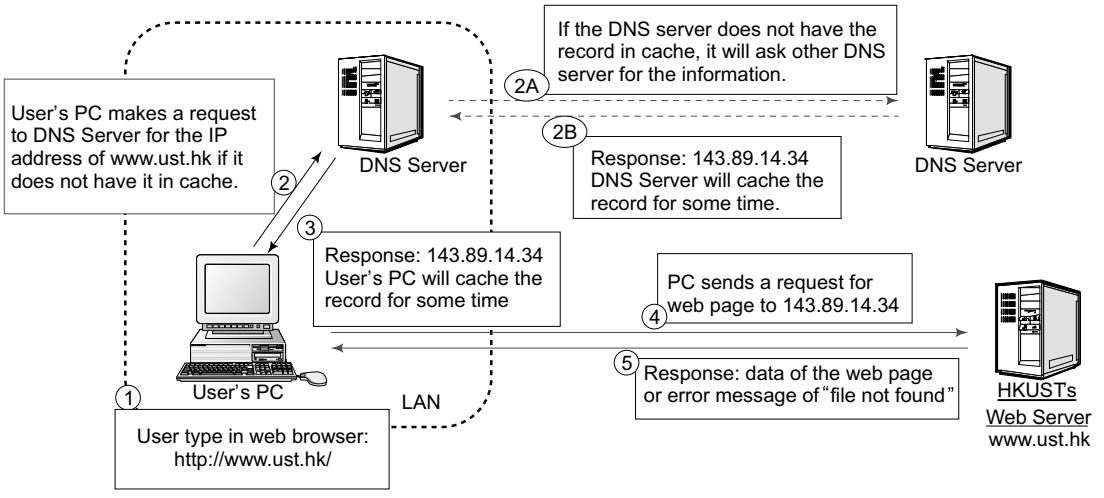


Figure 7.9 Working of TNS

4. Types of DNS records:
   - "A" (*Address)* record is used to translate a domain name to an IPv4 address
   - "MX" (*Mail eXchange*) record is used to translate from a name in the right-hand side of an e-mail address to the name/address of a machine able to handle mail for that domain.
   - "CNAME" (*Canonical Name*) record specifies a *domain name alias*.
   - "PTR" (*Pointer*) record is commonly used for *reverse DNS lookup* (IP-to-name lookup).
5. Note that it is possible to map multiple names to one IP address. It is also possible to map a name to multiple IP addresses for load balancing (Round robin DNS).
6. Dynamic DNS is a service for any host with dynamic IP address. Once a host has its IP address changes (as assigned by the ISP), a program on the host will update the DNS record at the Dynamic DNS server. The TTL (time-to-live) of the record is set to a very small number to facilitate frequent updates.

7.6 Solved Examples

■ **Example** Calculate the minimum bandwidth for a 5000bits/sec signal transmitted over a communication channel. The minimum bandwidth should be calculated for the worst-case sequence 101010. State any assumptions you make when performing the calculation.

■ **Answer:**
Given signal minimum bandwidth=5000bits/sec:
This is 1 bit every 1/5000 Sec.
or 0.2x10-3 sec or 0.2 milli-Sec (0.2ms)
Fundamental frequency only:
worst case 101010:
This means period (T) of "10" is 0.4ms
Giving a fundamental frequency of
f=1/T which is f_o = 1/0.4x10–3 = 2500Hz
has fundamental frequency f_o = 2500Hz,
Minimum bandwidth 0-2500Hz (considering only the fundamental)

■ **Example** A computer on a 6-Mbps network is regulated by a token bucket. The token bucket is filled at a rate of 1 Mbps. It is initially filled to capacity with 8 megabits. How long can the computer transmit at the full 6 Mbps?

Answer: Using the formula S=C/(M-ρ) where S is the burst length, C is the bucket capacity (8 Mb), M is the output rate (6 Mbps), and ρ is the bucket fill rate(1 Mbps) we can determine that the computer could transmit for 1.6 seconds.

■ **Example** Imagine a flow specification that has a maximum packet size of 1000 bytes, a token bucket rate of 10 million bytes/sec, a token bucket size of 1 million bytes, and a maximum transmission rate of 50 million bytes/sec. How long can a burst at maximum speed last?

■ **Answer:** Using the formula S = C/(M– ρ) we can calculate a burst at maximum speed to last for 204 milliseconds.

■ **Example** An audio streaming server has a one-way distance of 50 msec with a media player. It outputs at 1 Mbps. If the media player has a 1-MB buffer, what can you say about the position of the low-water mark and the high-water mark?

■ **Answer:** The low water mark must be higher than the amount of data that can be played in 50 msec so that buffer underrun does not occur and the high water mark must be at a point that is more than the amount of data that can be played in 50 msec below the 1 MB mark so that buffer overrun does not occur.

■ **Example** What is the bit rate for transmitting uncompressed 800x600 pixel color frames with 8 bits/pixel at 40 frames/sec?

■ **Answer:** The bit rate for this transmission would be 154 Mbps.

■ **Example** Consider a 100,000-customer video server, where each customer watches two movies per month. Half the movies are served at 8 P.M. How many movies does the server have to transmit at once during this time period? If each movie requires 4 Mbps, how many OC-12 connections does the server need to the network.

■ **Answer:** This server would need 336,456 OC-12 connections to supply the needed bandwidth.

■ **Example** A client sends a 128-byte request to a server located 100 km away over a 1-gigabit optical fiber. What is the efficiency of the line during the remote procedure call?

■ **Answer:** The line will be in use.02% of the time.

■ **Example** RTP is used to transmit CD-quality audio, which makes a pair of 16-bit samples 44,100 times/sec, one sample for each of the stereo channels. How many packets per second must RTP transmit?

Answer: 176,400 packets must be sent every second.

Consider the effect of using slow start on a line with a 10-msec round-trip time and no congestion. The receive window is 24 KB and the maximum segment size is 2 KB. How long does it take before the first full window can be sent?

■ **Answer:** It will take 120-msec before the first full window will be send.

Example To get around the problem of sequence numbers wrapping around while old packets still exist, one could use 64-bit sequence numbers. However, theoretically, an optical fiber can run at 75 Tbps. What maximum packet lifetime is required to make sure that future 75 Tbps networks do not have wraparound problems even with 64-bit sequence numbers? Assume that each byte has its own sequence number, as TCP does.

■ **Answer:** The maximum packet lifetime should be just shy of the total time it takes to issue all of the available 64-bit sequence numbers.

■ **Example** What is the bandwidth-delay product for a 50-Mbps channel on a geostationary satellite? If the packets are all 1500 bytes (including overhead), how big should the window be in packets?

■ **Answer:** The bandwidth-delay product for this would be 50-Mbps * the delay. Not knowing the exact location of the geostationary satellite makes the delay difficult to determine. Assuming the packets are all 1500 bytes the window size should be equal to the bandwidth-delay product.

■ **Example** Give three examples of protocol parameters that might be negotiated when a connection is set up.

■ **Answer:** The protocol parameters that may be negotiated during the setup of a connection could be maximum packet size, maximum transmission/reception speed, and quality of service standards.

■ **Example** Assuming that all routers and hosts are working properly and that all software in both is free of all errors, is there any chance, however small, that a packet will be delivered to the wrong destination?

■ **Answer:** There is always a chance that a packet will be delivered incorrectly or even totally lost. Interference and noise on the transmission lines can interfere with and create corruption that interrupts the delivery of packets on a network.

■ **Example** To what does the word node (host) refer?

■ **Answer:** A node or host is any addressable device attached to a network.

Name and describe two key issues related to computer networks.

Data transfer rate: The speed with which data is moved across the network

Protocol: The set of rules that define how data is formatted and processed across a network

■ **Example** Describe the client/server model and discuss how has it has changed how we think about computing.

■ **Answer:** The client/server is a model in which resources are spread across the web. The client makes a request for information or an action from a server and the server responds. For example, a file server, a computer dedicated to storing and managing files for network users, responds to requests for files. A web server, a computer dedicated to responding to requests for web pages, produces the requested page. Before the client/server model was developed, a user thought of computing within the boundaries of the computer in front of him or her. Now the functions that were provided within one computer are distributed across a network, with separate computers in charge of different functions.

■ **Example** Just how local is a local-area network?

■ **Answer:** A local-area network connects a relatively small number of machines in a relatively close geographical area,

usually within the same room or building, but occasionally a LAN spans a few close buildings.

■ **Example** Distinguish between the following LAN topologies: ring, star, and bus.

■ **Answer:** A ring topology is one in which the nodes are connected in a closed loop. A star topology is one in which the nodes are all connected to a central node. A bus topology is one in which the nodes share a common line.

■ **Example** How does the shape of the topology influence message flow through a LAN?

■ **Answer:** In a ring topology, messages flow in only one direction around the LAN. In a star topology, messages flow through the central node. In a bus topology, messages flow in both directions along the bus.

■ **Example** What is a MAN and what makes it different from a LAN and a WAN?

■ **Answer:** A MAN is a metropolitan-are network. It is a network with some of the features of both a LAN and a WAN. Large metropolitan areas have special needs because of the volume of traffic. MANs are collections of smaller networks but are implemented using such techniques as running optical fiber cable through subway tunnels.

■ **Example** Distinguish between the Internet backbone and an Internet service provider (ISP).

■ **Answer:** The Internet backbone is a set of high-speed networks that carry Internet traffic. An ISP is a company that provides access to the Internet, usually for a fee. An ISP connects directly to the Internet backbone or to a larger ISP with a connection to the backbone.

■ **Example** Name and describe three technologies for connecting a home computer to the Internet.

■ **Answer:** Phone modem: A modem is a device that converts computer data into an analog audio signal and back again, thus allowing you to transfer data to and from a computer using your telephone line.

DSL line: A DSL (digital subscriber line) is an Internet connection made using digital signals on regular phone lines.

Cable Modem: A cable modem is a device that allows computer network communication using the cable TV connection.

■ **Example** Phone modems and digital subscriber lines (DSL) use the same kind of phone line to transfer data. Why is DSL so much faster than phone modems?

■ **Answer:** Phone modems translate digital signals to analog in order to send them over voice frequencies.

■ **Example** DSL sends the digital signals over the same phone line but at a different frequency.

■ **Answer:** Because DSL and voice are at different frequencies, they can share the same phone line.

■ **Example** Why do DSL and cable modem suppliers use technology that devotes more speed to down loads than to uploads?

■ **Answer:** Users spend more time asking for data to be sent to their machines (downloads) than they do sending data to other machines (uploads). Therefore, DSL and cable modem suppliers maximize the speed on the most common task.

■ **Example** Messages sent across the Internet are divided into packets. What is a packet and why are messages divided into them?

■ **Answer:** A packet is a unit of data sent across a network. It is more efficient to send uniform sized messages across the Internet.

■ **Example** Explain the term packet switching.

■ **Answer:** Packets that make up a message are sent individually over the Internet and may take different routes to their destination. When all the packets arrive at the destination they are reassembled into the original message.

■ **Example** What is a router?

■ **Answer:** A router is a network device that directs packets between networks towards their final destinations.

■ **Example** What is a repeater?

■ **Answer:** A repeater is a network device that strengthens and propagates a signal along a lone communication line.

■ **Example** What problems arise due to packet switching?

■ **Answer:** Because packets may take different routes, they may not arrive in order. Thus, they must be reassembled into the right order at the receiving end.

■ **Example** What are proprietary systems and why do they cause a problem?

■ **Answer:** A proprietary system is one designed and built by a commercial vendor that keeps the technologies used private. If a network's software is a proprietary system, then it can only communicate with other networks that use the same software.

■ **Example** What is an open system and how does it foster interoperability?

■ **Answer:** An open system is a system based on a common model of network architecture adhering to an accompanying suite of protocols. If all commercial vendors adhere to a common logical architecture and protocols, then networks on multiple platforms from multiple vendors can communicate.

■ **Example** Compare and contrast proprietary and open systems.

■ **Answer:** Both proprietary and open systems can be used to create networks. Networks using the same proprietary systems can communicate with each other, but not with

networks that do not use the same system. Networks using open systems can all communicate.

■ **Example** What is the seven-layer logical breakdown of network interaction called?

■ **Answer:** Open Systems Interconnection (OSI) Reference Model

■ **Example** What is a protocol stack and why is it layered?

■ **Answer:** A protocol stack is layers of protocols that build and rely on each other. Protocols are layered so that new protocols can be developed without abandoning fundamental aspects of lower levels.

■ **Example** What is a firewall, what does it accomplish, and how does it accomplish it?

■ **Answer:** A firewall is a computer system that protects a network from inappropriate access. A firewall filters incoming traffic, checking the validity of incoming messages, and perhaps denying access to messages. For example a LAN might deny any remote access by refusing all traffic that comes in on port 23 (the port for telnet).

■ **Example** What is a hostname and how is it composed?

■ **Answer:** A hostname is a unique identification for a specific computer on the Internet made up of words separated by dots.

■ **Example** What is an IP address, and how is it composed?

■ **Answer:** An IP address is made up of four numeric values separated by dots that uniquely identifies a computer on the Internet.

■ **Example** What is the relationship between a hostname and an IP address?

■ **Answer:** Hostnames are for people and IP addresses are for computers. Each hostname is translated into a unique IP address. People refer to the machine by its hostname; computers refer to the machine by its IP address.

■ **Example** Into what parts can an IP address be split?

■ **Answer:** An IP address can be split into a network address, which specifies the network, and a host number, which specifies a particular machine on the network.

■ **Example** How many hosts are possible in Class C networks, in Class B networks, and in Class A networks?

■ **Answer:** Class C networks use three bytes for the network number and only one byte for the host number, so they can identify 256 hosts. Class B networks use two bytes for the network number and two bytes for the host number, so they can identify 32768 hosts. Class A networks use one byte for the network number and three bytes for the host number, so they can identify 224 hosts.

■ **Example** What is a domain name?

■ **Answer:** A domain name is that part of the hostname that specifies the organisation or group to which the host belongs.

■ **Example** What is a top-level domain name?

■ **Answer:** The last part of a domain name that specifies the type of organisation or its country of origin.

■ **Example** How does the current domain name system try to resolve a hostname?

■ **Answer:** First a request is sent to a nearby domain name server (a computer that attempts to translate a hostname into an IP address). If that server cannot resolve the hostname, it sends a request to another domain name server. If the second server can't resolve the hostname, the request continues to propagate until the hostname is resolved or the request expires because it took too much time.

■ **Example** Frames are generated at node A and sent to node C through an intermediate node B. Distance between A and B is 4000Km while B and C is 1000Km. Determine the minimum transmission rate required between nodes B and C so that the buffers at node B are not flooded, based on the following:

- The data rate between A and B is 100 kbps.
- The propagation delay is 5 msec/km for both the lines
- There are full duplex, error free lines between the nodes.
- All data frames are 1000 bits long; ACK frames are separate frames of negligible length.
- Between A and B, a sliding window protocol is used, with a window size of 3 (three).
- Between B and C, stop and wait is used.

■ **Answer:**

In order not to flood the buffers of B, the average number of frames entering and leaving B must be the same over a long interval.

A to B: Propagation time = 4000* 5 msec = 20 msec

Transmission time per frame = 1000/(100*103) = 10 msec.

B to C: Propagation time = 1000* 5 msec = 5 msec

Transmission time per frame = x = 1000/R

Frame time in between A to B=1000bits/100Kbps=10msec

R = data rate between B and C (unknown)

A can transmit three frames to B and then must wait for the acknowledgement of the first frame before transmitting additional frames. The first frame takes 10 msec to transmit between A to B (frame time); the last bit of the first frame arrives at B 20 msec after it was transmitted and therefore 30 msec after the frame transmission began. It will take an additional 20 msec for B's ack to return to A. Thus A can transmit three frames in 50 msec.

B can transmit one frame to C at a time. It takes 5 + x msec for the frame to be received at C and an additional 5 msec for C's acknowledgement to return to A. Thus, B can transmit one frame every 10 + x msec, or three frames every 30 + 3x msec. Thus:
30+3x=50
Therefore, x= 6.66 msec
Now, date between B and C, R = 1000/x = 150 kbps.

■ **Example** A 4Mbps token ring has a token holding time value of 10msec. What is the longest frame that can be sent on this ring?

■ **Answer:** Data rate = 4Mbps
Token holding time = 10msec
Therefore, Frame length = 4*106*10*10-3= 40000 bits
Thus, the longest frame that can be send on this ring is 40000 bits or 5000 bytes.

■ **Example** Find the minimum frame length for a 1Mbps bit rate CSMA/CD LAN that is having a maximum network span of 10 kilometers with no repeaters. Assume a medium propagation delay of 4.5 nanoseconds per meter. Is CSMA/CD a reasonable protocol for a network of this span and bit rate?

■ **Answer:** Minimum frame size for CSMA/CD LAN is 2 times of propagation delay.
Propogation Delay, Tpr = $(4.5 * 10^{-9})*(10 * 10^{3})$
$= 4.5 * 10^{-5}$ sec.
Thus, frame size = $(1.0 * 10^{6}) * (9.0 * 10^{-5}) = 11.25$ bytes.
CSMA/CD would be a very reasonable protocol for a network of this span and speed since the minimum frame size is not " excessive" (e.g., larger than 64 bytes)

■ **Example** Assume a 100Mbps link of 10,000meters in length with 5 nanoseconds per meter propagation delay. Assume constant length 400 byte data frames, 64 byte ACK frames, 10 microsecond of processing delay for each data frame, and 5 microseconds of processing time for each ACK. The sender always has data to send. Solve for link utilisation (U) between a sender and a receiver assuming a stop and wait protocol.

■ **Answer:**
t_pr = 10,000 * $5e^{9}$ = 50 s
t_fr = 8 * 400 / $100e^{6}$ = 32 s
Ack Delay : t_ack = 8 * 64 / $100e^{6}$ = 5.12 s
We note that the processing delays are not negligible compared to these values so we must include them in our calculation of U...
U = t_fr / (t_pr + t_fr + t_proc + t_ack + t_proc) = 21%
If we neglect t_ack and t_proc our result would have been U = 24% which is more than 10% "off" from the real result confirming that to ignore these values would not have been correct to do.

■ **Example** For a Gigabit Ethernet the minimum packet length is 512 bytes, for 100 Mbps the minimum packet length is 64 bytes. Compute media (or wire) speed for Gigabit and 100Mbps Ethernet for minimum length packets. What conclusion can you draw?

■ **Answer**
$1.0e^{9}$ / (8 * (512 + 12 + 8)) = 234,962 pkts/sec for Gigabit
$1.0e^{8}$ / (8 * (64 + 12 + 8)) = 148,819 pkts/sec for 100Mbps
The "8" is for preamble and the "12" is for inter-frame gap. The media speed for Gigabit is less than double that of 100Mbps for minimum size packets. To get the full gain of Gigabit, you must use large payloads

■ **Example** Consider building an IEEE 802.3 network at 1Gbps over a 1km cable with no repeaters. What is the minimum frame size?. (Assume the signal speed in the cable is 200,000Km/sec).

■ **Answer:** In order to detect collisions, the station must be still transmitting when the first bit reaches the far end of the cable.
For a 1Km cable, the one waypropagation time is t**=5msec.**
So **2t=10msec (RTT, round trip transmission).**
At 1 Gbps all frames shorter than 10,000 bits can be completely transmitted in under 10microsec. Thus, the minimum frame size is 10,000 bits or 1250 bytes.

■ **Example** How many bps can the modem achieve at 1200 baud?

■ **Answer:** There are 4 legal values per baud, or, in other words, each signal change represents 2 bits. Thus the bit rate is twice the baud rate. At 1200 baud, the bit rate is 2400 bps.

■ **Example** Imagine 2 LAN bridges, both connecting a pair of 802.4 networks. The first bridge is faced with 100 512 byte frames per seconds that must be forwarded. The second is faced with 200 4096 byte frame per second. Which bridge do you think will need the faster CPU? Discuss.

■ **Answer:** The 100 frames/sec bridge would need a faster CPU. Although the other one has a higher throughput, the 100 frames/sec bridge has more interrupts, more process switches, more frames passed and more of everything that needs the CPU.

■ **Example** Consider a 200-meter 4Mbps token ring containing 20 stations, each transmitting with equal priority. Suppose no station is allowed to transmit more than 5000 data octets before giving up the token. Once a station gives up a token how long will it take (in the worst case) for that station to get the token again?

■ **Answer:** Designate the station in question by A. The worst case scenario occurs if each of the other 19 stations also has 5000bytes to send.

- Assume speed of signal: 200meters/msec. Token is 24bytes/msec.

- Let t_{5000} is the time to transmit 5000bytes and t_{token} the time to transmit the token(for the nineteen stations plus A).

With data rate 4 Mbps:

- T_{5000} = 5000*8*250nsec = 10000msec.
- t_{token} = 24*250nsec = 6msec.
- Once the last bit is transmitted, it requires 20microsec (micr sec to cover 200m+19microsec internal delay).Once the last frame bit has circulated the ring, station can send the token.

Thus the time between sending the first bit and first token bit is 10020 micro-sec (since stations are 10 m apart, the propagation time of the token between adjacent stations is 0.05 micro second(negligible). Time required to send one frame followed by one token is 10020+6 micr sec

Time A has to wait:19*10026micro-sec+6(this is 1st token)=190500 micro-sec

■ **Example** Ethernet frames must be at least 64 bytes long to ensure that the transmitter can detect collisions. A faster Ethernet has the same minimum frame size but can transmit 10 times faster. How is it possible to still detect collisions?

■ **Answer:** In order to detect collisions, the station must be still transmitting when the first bit reaches the far end of the cable. As the network speed goes up, the minimum frame length must go up or the maximum cable length must come down proportionally.

Indeed, let: u=speed of signal, l = length of the cable, and s=data rate (bps)

Then the minimum frame size is: x=s*l/u.

Thus, since x is the same for both Ethernet and Fast Ethernet, and s in Fast Ethernet is 10 times as much as in Ethernet, l, the wire length, must be 1/10 as long as in Ethernet.

■ **Example** A group of N stations share a 56kbps pure ALOHA channel. Each station outputs a 1000bit frame on an average of once every 100 sec, even if the previous one has not yet been send (e.g. the stations are buffered). What is the maximum value of N?

■ **Answer:** Recall that ALOHA achieves an average throughput of appr 18%, when operating at reasonable load. In an ALOHA network with channel capacity 56 kbps, only 18% of this capacity will be used to deliver meaningful data.

With pure ALOHA the usable bandwidth is 0.184 * 56kbps= 10.3kbps.

This 10 kbps must be divided among N hosts, each of which is transmitting an average of 1000 bits every 100 seconds. This corresponds to a transmission rate of 10 bits per second per host. If the channel can support 10 kbps of data, then it can support up to N users, each transmitting at 10 Bps. Each station requires 10 bps (1000bit/ 100sec), so N = 10300 /10 = 1030 stations.

■ **Example** Ten thousand airline reservation stations are coming for the use of a single slotted ALOHA channel. The average station makes 18 request/hour. A slot is 125 msec. What is the approximate total channel load.

■ **Answer:**

Average requests for 10000 stations = 10^4 x 18 / (60 x 60) =50 requests/sec

Average slots number = 1 / (125 x10^–6) = 8000 slots/sec.

Total channel load = average requests / average slots number

= 50 / 8000 = 0.0625

Hence, the total channel load is 0.0625 request/slot.

■ **Example:** Consider an application which transmits data at a steady rate (e.g., the sender generates an N bit unit of data every k time units, where k is small and fixed). Also, when such an application starts, it will stay on for relatively long period of time. Answer the following questions, briefly justifying your answer:

(a) Would a packet-switched network or a circuit-switched network be more appropriate for this application? Why?

(b) Suppose that a packet-switching network is used and the only traffic in this network comes from such applications as described above. Furthermore, assume that the sum of the application data rates is less that the capacities of each and every link. Is some form of congestion control needed? Why?

■ **Answer:** (a) A circuit-switched network would be well suited to the application described, because the application involves long sessions with predictable smooth bandwidth requirements. Since the transmission rate is known and not bursty, bandwidth can be reserved for each application session circuit with no significant waste. In addition, we need not worry greatly about the overhead costs of setting up and tearing down a circuit connection, which are amortised over the lengthy duration of a typical application session. (b) Given such generous link capacities, the network needs no congestion control mechanism. In the worst (most potentially congested) case, all the applications simultaneously transmit over one or more particular network links. However, since each link offers sufficient bandwidth to handle the sum of all of the applications' data rates, no congestion (very little queuing) will occur.

■ **Example**

$F = M * L$ bits over a path of Q links. Each link transmits at R bps. The network is lightly loaded so that there are no queueing delays. When a form of packet switching is used, the $M * L$ bits are broken up into M packets, each packet with L bits. Propagation delay is negligible.

(a) Suppose the network is a packet-switched virtual-circuit network. Denote the VC set-up time by t_s seconds. Suppose to each packet the sending layers add a total of *h* bits of header. How long does it take to send the file from source to destination?

(b) Suppose the network is a packet-switched datagram network, and a connectionless service is used. Now suppose each packet has *2h* bits of header. How long does it take to send the file?

(c) Repeat (b), but assume message switching is used (i.e., *2h* bits are added to the message, and the message is not segmented).

(d) Finally, suppose that the network is a circuit switched network. Further suppose that the transmission rate of the circuit between source and destination is *R* bps. Assuming t_s set-up time and h bits of header appended to the entire file, how long does it take to send the file?

■ **Answer:**

(a) The time to transmit one packet onto a link is *(L + h) / R*. The time to deliver the first of the *M* packets to the destination is *Q * (L + h) / R* . Every *(L + h) / R* seconds a new packet from the *M* - 1 remaining packets arrives at the destination. Thus the total latency is *ts + (Q + M - 1) * (L + h) / R*

(b) *(Q + M - 1) * (L + 2h) / R*

(c) The time required to transmit the message over one link is *(LM + 2h) / R*. The time required to transmit the message over *Q*links is *Q * (LM + 2h) / R*

(d) Because there is no store-and-forward delays at the links, the total delay is *ts + (ML + h) / R*

■ **Example** This elementary problem explores propagation delay and transmission delay, two central concepts in data networking. Consider two hosts, Hosts A and B, connected by a single link of rate *R* bps. Suppose that the two hosts are separated by *m* meters, and suppose the propagation speed along the link is *s* meters/sec. Host A is to send a packet of size *L* bits to Host B.

(a) Express the propagation delay, d_{prop} in terms of *m* and *s*.

(b) Determine the transmission time of the packet, d_{trans} in terms of *L* and *R*.

(c) Ignoring processing and queing delays, obtain an expression for the end-to-end delay.

(d) Suppose Host A begins to transmit the packet at time t=0. At time t = d_{trans}, where is the last bit of the packet?

(e) Suppose dprop is greater than d_{trans}. At time t=d_{trans}, where is the first bit of the packet?

(f) Suppose d_{prop} is less than d_{trans}. At time t=d_{trans}, where is the first bit of the packet?

(g) Suppose *s*=2.5*108, *L*=100 bits and *R*=28 kbps. Find the distance *m* so that d_{prop} equals d_{trans}.

■ **Answer:**

(a) $d_{prop} = m/s$ seconds.

(b) $d_{trans} = L/R$ seconds.

(c) $d_{end\text{-}to\text{-}end} = (m / s + L / R)$ seconds.

(d) The bit is just leaving Host A.

(e) The first bit is in the link and has not reached Host B.

(f) The first bit has reached Host B.

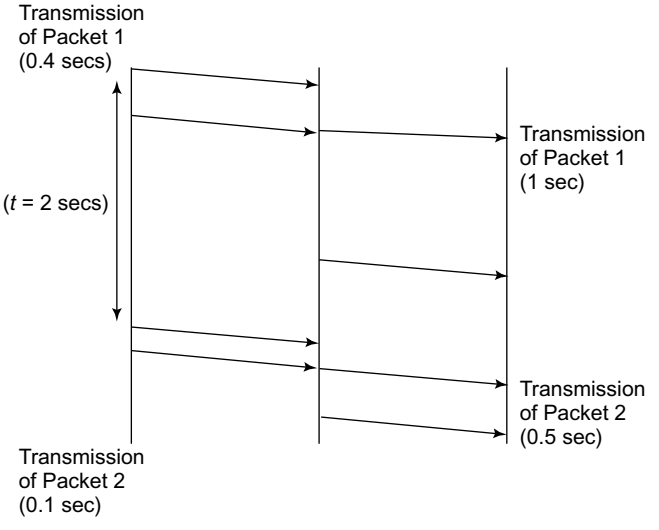
(g) Want m = (L / R) * S = (100 / 28 * 10^3) * (2.5 * 10^8) = 893 km

■ **Example** One host (named X) sends two packets to another host (named Y) through router S1. Assume there is no queuing delay and processing delay. The bandwidth and propagation delay of link 1 are 25 Kbps and 0.1 ms respectively. The corresponding values for link 2 are 10 Kbps and 0.1 ms. The first packet is of size 10 Kb, and the second one of size 5 Kb. Draw the time-line diagrams illustrating these transmissions in the following 2 cases:

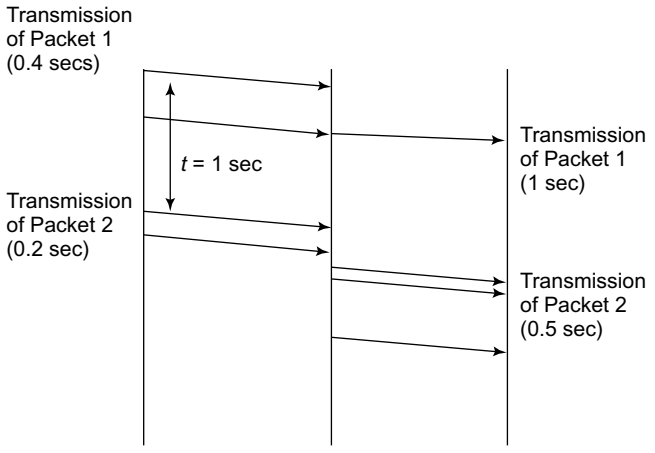
(a) the second packet is sent 2 secs after the first one is sent (to be precise, the transmission of the second packet begins 2 secs after the transmission of the first one)

(b) the second packet is sent 1 sec after the first one is sent (the transmission of the second packet begins 1 sec after the transmission of the first one)

(a)



(b)



■ **Example** When using virtual-circuit transport, the virtual-circuit setup time is 400ms. Packets travel over a path that goes through 10 links and each link is a 56 kbps line. Each packet contains a 7-byte header and 400 bits of data. When using a datagram transport, each packet contains a 12-byte header and 400-bit data. However, there is no circuit setup delay. In the following parts, ignore the processing delay, propagation delay and queuing delay.

(a) How long does it take to transmit N packets using virtual-circuit transport?

(b) How long does it take to transmit N packets using datagram transport?

(c) For what values of N is the transfer by virtual-circuit transport faster? For what value of N is datagram transport faster?

■ **Answer:**

(a) The packet size is the size of data + the size of header = (400 + 56) bits = 456 bits.

Time to transmit 456 bits = 456/56k = 8.14ms.

The transmission of N packets over 10 links takes, N*8.14ms + 9*8.14ms.

Adding the setup-up time, the total delay is 400ms + N*8.14ms + 9*8.14ms = 473ms + N*8.14ms.

(b) The packet size is the size of data + the size of header = (400 + 96) bits = 496 bits.

Time to transmit 496 bits = 496/56k = 8.86ms.

The transmission of N packets over 10 links takes, N*8.86ms + 9*8.86ms.

There is no setup time. As a result, the total delay is N*8.86ms + 79.74ms.

(c) The two are almost equal when N = 548.

Therefore, for messages that are shorter than 548 packets, use datagram transport. Otherwise, use virtual-circuit service.

■ **Example** Suppose a 100-Mbps point-to-point link is being set up between the earth and a new lunar station which is approximately 385000km from earth, and data travels over the link at the speed of light (3×10^8 m/s). The transmission is reliable and sliding window-based.

(a) Calculate the minimum Round Trip Time (RTT) for the link.

(b) Using the RTT as the delay, calculate the delay x bandwidth product for this link.

(c) What is the maximum number of bits that can be in transit at any one point?

(d) A camera on the lunar base takes pictures of the earth and saves them in digital format to disk. Suppose Mission Control on earth wishes to download the most current image, which is 25MB. What is the minimum amount of time that will elapse between when the request for the data goes out and the transfer is finished?

■ **Answer:**

(a) The minimum RTT is two times of the propagation delay on the link = $(2*385000000)/3*10^8$ = 2.57s.

(b) The delay-bandwidth product = (2.57 s)*(100 Mbits/s) = 257Mbits = 32M bytes.

(c) Same as the delay bandwidth product.

(d) It would then take (1/2) RTT for the earth to make the download request and another (1/2) RTT for the propagation delay for sending the data from the moon to the earth. The total time is the sum of the transmission time and the two (1/2) RTT.

Minimum amount of time = 25MB/100Mbps + (1/2) RTT + (1/2) RTT

= 200Mb/100Mbps + 2.57 s

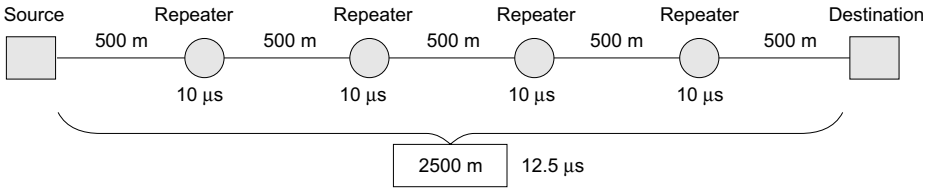
= 2 s + 2.57s

= 4.57s

■ **Example** Suppose the length of a 10Base5 cable is 2500 metres. If the speed of propagation in a thick coaxial cable is 200,000,000 m/s how long does it take for a bit to travel from the beginning to the end of the network? Assume that there is a 10μ s delay in the equipment.

■ **Answer:** A typical 10Base5 network will be having a maximum length of 500 metres, so repeaters should be inserted into the cable in order to ensure transmission is possible over the full length of the 2500m cable. Four repeaters are required as shown below:

Thus, time needed for 1 bit to transfer from one end to the other is =4 x 10μs + 2500/200,000,000 = 52.5μs.



■ **Example** The data rate of 10Base5 is 10Mbps. How long does it take to create the smallest frame?

■ **Answer:** The smallest frame is 64 bytes or 512 bits. With a data rate of 10 Mbps, we have

$$T_{fr} = (512 \text{ bits}) / (10 \text{ Mbps}) = \mathbf{51.2\ \mu s}$$

This means that the time required to send the smallest frame is the same at the maximum time required to detect the collision.

■ **Example** Given the dataword 1010011110 and the divisor 10111

(a) Show the generation of the codeword at the sender site using binary division

(b) Show the checking of the codeword at the receiver site assuming no error has occurred.

(c) What is the syndrome at the receiver end if the dataword has an error in the 5th bit position counting from the right? Namely: dataword 1010001110 is received.

■ **Answer:** (a) Binary division case

CRC Checksum is 1010

Codeword was 10100111101010

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
M=	1	0	1	0	0	1	1	1	1	0				
G=														
10111	1	0	1	0	0	1	1	1	1	0	0	0	0	0
	1	0	1	1	1									
		0	0	1	1	1								
		0	0	0	0	0								
			0	1	1	1	1							
			0	0	0	0	0							
				1	1	1	1	1						
				1	0	1	1	1						
					1	0	0	0	1					
					1	0	1	1	1					
						0	1	1	0	0				
						0	0	0	0	0				
							1	1	0	0	0			
							1	0	1	1	1			
								1	1	0	1	0		
								1	0	1	1	1		
									1	0	0	1	0	
									1	0	1	1	1	
										0	1	0	1	0
											0	0	0	0
											1	0	1	0
T=	1	0	1	0	0	1	1	1	1	0	1	0	1	0
											CRC Checksum			
			Original Message											

(b) At the receiver, result of division operation is given below. We find checksum as 0000.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
M=	1	0	1	0	0	1	1	1	1	0				
G=														
10111	1	0	1	0	0	1	1	1	1	0	1	0	1	0
	1	0	1	1	1									
		0	0	1	1	1								
		0	0	0	0	0								
			0	1	1	1	1							
			0	0	0	0	0							
				1	1	1	1	1						
				1	0	1	1	1						
					1	0	0	0	1					
					1	0	1	1	1					
						0	1	1	0	0				
						0	0	0	0	0				
							1	1	0	0	1			
							1	0	1	1	1			
								1	0	1	1	1		
								1	0	1	1	1		
									0	0	0	0	0	
										0	0	0	0	0
											0	0	0	0

(c) Suppose that we received the corrupted dataword with the old CRC value as follows: **10100011101010.** We can determine the syndrome in this case as:0010

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
M=	1	0	1	0	0	1	1	1	1	0				
G=														
10111	1	0	1	0	0	0	1	1	1	0	1	0	1	0
	1	0	1	1	1									
		0	0	1	1	0								
		0	0	0	0	0								
			0	1	1	0	1							
			0	0	0	0	0							
				1	1	0	1	1						
				1	0	1	1	1						
					1	1	0	0	1					
					1	0	1	1	1					
						1	1	1	0	0				
						1	0	1	1	1				
							1	0	1	1	1			
							1	0	1	1	1			
								0	0	0	0	0		
								0	0	0	0	0		
									0	0	0	0	1	
									0	0	0	0	0	
										0	0	0	1	0
											0	0	0	0
											0	0	1	0

■ **Example** What is the use of subnet mask?

■ **Answer:** The subnet mask enables us to subdivide addresses to achieve more useful mixes of host and subnets for a given range of addresses.

■ **Example** Given IP address 136.27.32.104 and subnet mask of FFFFFE00, which class network is it, how many subnets are there, and what is the host address.

■ **Answer:** Binary equivalent of given IP address: 136.27.32.104 = **10001000 00011011 00100000 01101000**

First two bits are 10, thus it is class B type of network. Thus, most significant 16 bits indicates the network address. Now consider the net mask's binary equivalent. 11111111111111111111000000000. As we have seven 1s after first sixteen 1's, we will be having 2^7 subnets. Also, least 9 bits are for host(from last nine zeros in net mask). Host number is 001101000=104.

■ **Example** Consider the queuing delay in a router buffer with infinite size. Assume that each packet consists of L bits. Let *R* denote the rate at which packets are pushed out of the queue (bits/sec).

(a) Suppose that the packets arrive periodically every *L=R* seconds. What is the average router queuing delay?

■ **Answer:** Zero.

(b) (0.5) Suppose that N packets arrive simultaneously every *(L/R)N* seconds. What is the average router queuing delay?

Answer *L/R*(N-1)/2 second*

■ **Example** Differentiate Non-persistent and persistent HTTP connections.

■ **Answer:**

Non-persistent

- HTTP/1.0
- server parses request, responds, and closes TCP connection
- 2 RTTs to fetch each object
- Each object transfer suffers from slow start

Persistent

- default for HTTP/1.1
- on same TCP connection: server, parses request, responds, parses new request,..
- Client sends requests for all referenced objects as soon as it receives base HTML.
- Fewer RTTs and less slow start.

■ **Example** A user in Delhi, connected to the internet via a 2 Mb/s connection retrieves a 25 KB (B=bytes) web page from a web server in Mumbai, where the page references 3 images of 200 KB each. Assume that the one way propagation delay is 20 ms. Approximately how long does it take for the page (including images) to appear on the user's screen, assuming non-persistent HTTP using a single connection at a time?

■ **Answer:** In non-persistent HTTP connection, every object request is dealt separately by establishing separate connection for each. For every request, we need 4 RTT which includes two RTTs for fetching object in addition to requests and responses. Thus, time needed to load the page = *4*(80 ms) + (8*(25+3*200) Kbits)/(2 Mb/s) = 320 ms + 2.5 sec = 2.82 sec*

■ **Example**

Suppose within your web browser you click on a link to obtain a web page. Suppose that the IP address for the associated URL is cached in your local host, so that a DNS look up is not necessary. Further suppose that web page associated with the link contains a small amount of HTML text as well as N very small objects. Let RTT denote the round trip delay between the local host and the server containing the objects. Assuming zero transmission time of the objects,

(a) With non-persistent HTTP with no parallel TCP connections, how much time elapses from when the client clicks on the link until the client receives the objects?

■ **Answer:**

T = RTT (TCP establishment) + RTT (for HTML base file)

Total Time elapsed is (N+1)*T

(b) Repeat (a) for non-persistent HTTP with parallel connections.

■ **Answer:**

T = RTT (TCP establishment) + RTT (for HTML base file)

Total Time elapsed is 2*T. This is because the client will parse the HTML base file and he should find the referenced objects. Then it established a separate TCP connection in parallel for each object

(c) Repeat (b) for persistent HTTP with pipelining.

■ **Answer:**

T=RTT (TCP establishment) + RTT (for HTML base file)

Total Time elapsed is T+ RTT. This is because the client will parse the HTML base file and he should find the referenced objects. Then it sends requests to all referenced objects using pipelining method. The assumption here is that all these N objects are on the same server.

■ **Example** A user in Lucknow, connected to the internet via a 20 Mb/s (b=bits) connection retrieves a 250 KB (B=bytes) web page from a server in Chennai, where the page references 4 images of 1 MB each. Assume that the one way propagation delay is 25 ms. Approximately, how long does it take for the page (including images) to appear on the user's screen, assuming non-persistent HTTP using a single connection at a time (for this part, you should ignore queueing delay and transmission delays at other links in the network)?

■ **Answer:**
5(100) ms + (2 + 4*8 Mb/(20 Mb/s) = 500 ms + 1.7 sec = 2.2 seconds*
How long does it take if the connection uses persistent HTML (single connection)?
100 ms + 50 ms + 1.7 sec = 1.85 seconds
Suppose that the path from the server to the user passes through a 1 Gb/s link at a router *R*, and that the rate at which packets arrive at router *R* that must be sent on this link is 450,000 packets per second. If the average packet length is 2,000 bits, what is the average queuing delay at this link?
I = (900 Mb/s)/(1 Gb/s) = 0.9, so average queue length = I/(1-I) =.9/.1 = 9 packets
time to send a packet = (2000 bits)/(1000 bits per μs)
average delay = 9*2 μs = 18 μs

■ **Example** Consider a 10 Mb/s link that is 400 km long, with a queue large enough to hold 2,000 packets. Assume that packets arrive at the queue with an average rate of 4,000 packets per second and that the average packet length is 2,000 bits. Approximately, what is the propagation delay for the link (be sure to include the units in your answer)?

■ **Answer:** *(400 km)/(210,000 km/s) is approximately 2 ms*
The transmission time for an average length packet – *(2000 bits)/(10 bits/**μ**s) = 200 **μ**s*
The traffic intensity – *(4000 packets/sec)*(2000 bits/packet)/(10 Mb/s) = 0.8*
The average number of packets in the queue – *0.8/(1- 0.8) = 4*
The average queuing delay – *4*200 = 800 **μ**s*

■ **Example** The diagram below shows three routers and four layer 2 networks. The table is the routing table for the left-hand router. In three of the entries in the routing table, the output has been left blank. Fill in the missing values. Based on the information in the table, what is the IP address of the interface that connects the top router to the central subnet?

■ **Answer:** *3.2.1.3.* If we observe the table ip address that corresponds to top router 4.5.6.0/24 is 3.2.1.3. That means, all the packets that are destined to 4.5.6.0/24 will be sent to 3.2.1.3. This indicates that 3.2.1.3 is the router interface address of top network.

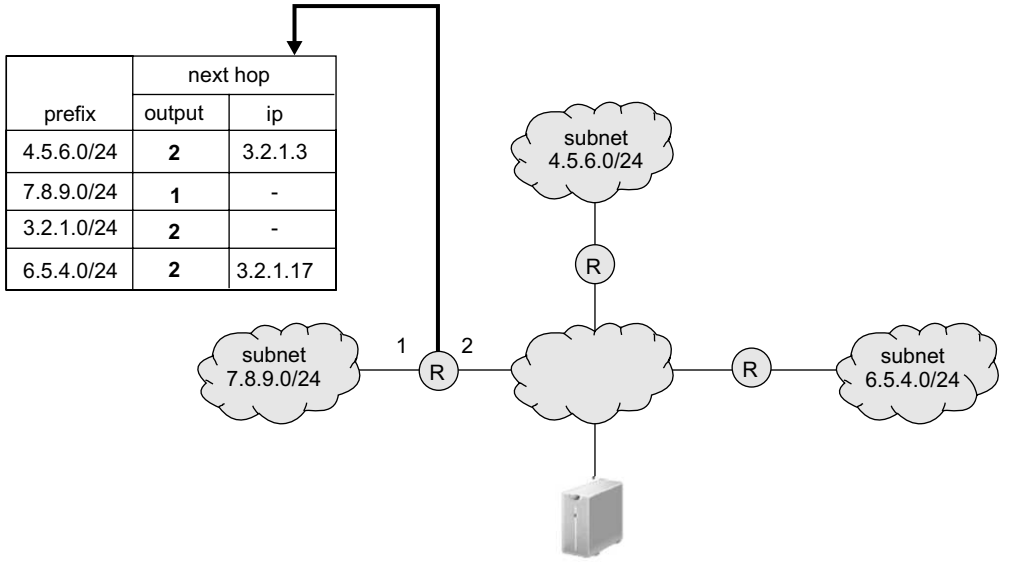
■ **Example** What is the IP address of the interface that connects the right-hand router to the central subnet?

■ **Answer:** *3.2.1.1.* Explanation is same as above.

■ **Example** What is the subnet prefix for the central subnet?

■ **Answer:** *3.2.1.0/24*

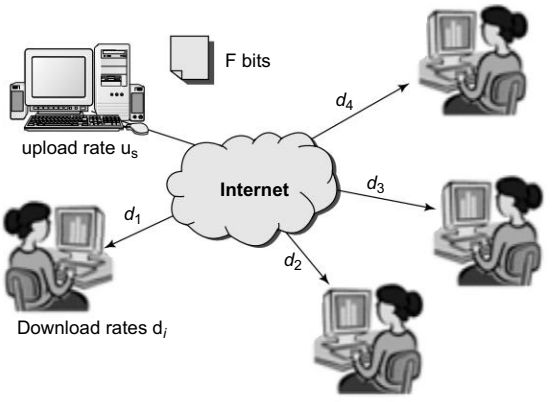
prefix	next hop	
	output	ip
4.5.6.0/24	**2**	3.2.1.3
7.8.9.0/24	**1**	-
3.2.1.0/24	**2**	-
6.5.4.0/24	**2**	3.2.1.17



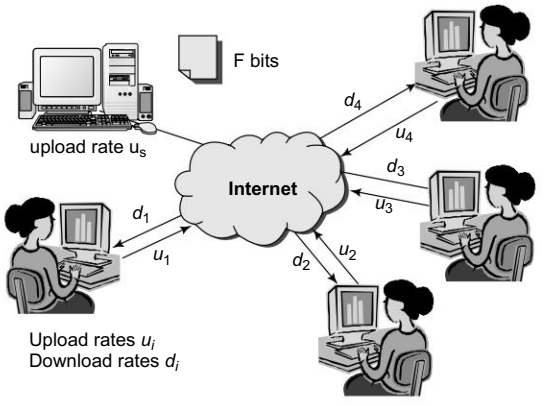
■ **Example** Differentiate server distributed large file and peer to peer based distribution of file among multiple clients.

■ **Answer:** See the following figure in which a server will be distributing a large file of size F bits.
Important points in Server sending a large file to *N* receivers
– Large file with *F* bits
– Single server with upload rate u_s
– Download rate d_i for receiver *i*

- Server transmission to *N* receivers
  - Server needs to transmit *NF* bits
  - Takes at least NF/u_s time
- Receiving the data
  - Slowest receiver receives at rate $d_{min} = min_i\{d_i\}$
  - Takes at least F/d_{min} time
- Download time: $max\{NF/u_s, F/d_{min}\}$



The following figure illustrates peers helping in distributing the file among the multiple clients.



Important points in this approach of distributing a file.

- Start with a single copy of a large file
  - Large file with F bits and server upload rate u_s
  - Peer i with download rate d_i and upload rate u_i
- Two components of distribution latency
  - Server must send each bit: min time F/u_s
  - Slowest peer receives each bit: min time F/d_{min}
- Total upload time using all upload resources
  - Total number of bits: NF
  - Total upload bandwidth $u_s + sum_i(u_i)$

Total: $max\{F/u_s, F/d_{min}, NF/(u_s+sum_i(u_i))\}$

■ **Example** (a) Consider a situation in which 1000 clients are trying to download a 10 MB file from a server. If the server has a 100 Mb/s access link and the clients have access links with a downstream rate of 2 Mb/s, how long does it take to download the file to all clients, under ideal conditions (you may ignore the time to establish a TCP connection to the server).

■ **Answer:** The total download bandwidth is 2 Gb/s(1000x2Mb/s), so the server's access bandwidth is the limiting factor. The number of bits that the server must send is 80 gigabits (1000x10MB=1000x10x10^6x8bits), so under ideal conditions, it would take about 800 seconds(80gigabits/100Mb/s=80x10^9/100x10^6) to deliver the file to all clients.

(b) Now, consider the peer-to-peer situation, in which there is no server and one peer holds the file to be distributed. Assuming that the upstream rate from each peer is 1 Mb/s and the downstream rate is 2 Mb/s, how long does it take to distribute the file to all peers?

■ **Answer:** In this case, the limiting factor is the upstream rate from the peers, so the time is 80 Mb/(1 Mb/s), so 80 seconds. That is, file size divided by upload speed of peer as explained in the previous question.

■ **Example** The diagram at below shows the state of the sending side of a sliding window protocol with a window size of 4 and the selective repeat feature. The array represents the send buffer and each pair in the buffer represents a packet and its sequence number, together with the time at which it is scheduled to be retransmitted. (so, for example, the pair (p3,14) denotes a packet with sequence number 3, which is to be retransmitted at time 14). Assume that the timeout used for retransmitting packets is 20 time units.

	0	
	1	
	2	(p2,10)
sendBase=2	3	(p3,14)
	4	(p4,22)
nexSeqNum=5	5	
	6	
time=6	7	

a. Suppose that at time 7, the application passes us a new payload to be sent, at time 8, we receive an ack with sequence number 3, and that at time 9, we receive an ack with sequence number 2. Show the state of the sender at time 10, in a diagram.
b. If no additional payload or ack is received before time 25, what is the next thing that should happen and how does it affect the sender's state?

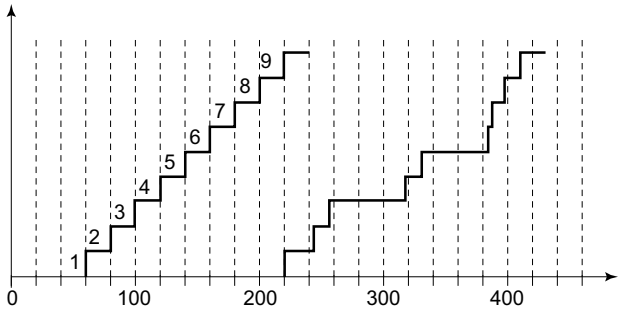
■ **Answer:** (a) If we observe the given figure, we may find that there are 3 packets under transmission and window size is 4. Thus, when new payload arrives at 7, it will be sent by making it retransmission time as 27 (7+timeout for retransmission=7+20). Do remember that its sequence number is taken as 5 and nextSeqNum is made as 6. The following figure illustrates the state of the sender at time 10.

	0	
	1	
	2	
sendBase=4	3	
	4	(p4,22)
nexSeqNum=6	5	(p5,27)
	6	
time=10	7	

(b) At time 22, packet p4 should be retransmitted and its retransmit time should be increased to 42.

■ **Example** The chart below shows packets sent by an audio source and received at a sink. Assume that the time between packet transmissions is 20 ms and that packets are time stamped by the sender and that the receiver playout buffer uses a fixed playout delay of 200 ms. Also, assume that the sender's and receiver's clocks are exactly synchronized.

a. List the packet numbers for those packets that arrive too late to be played out.
b. What is the smallest playout delay required to ensure that packets never arrive too late for playout (for the example)? You may round up to a multiple of 20 ms.



■ **Answer:**

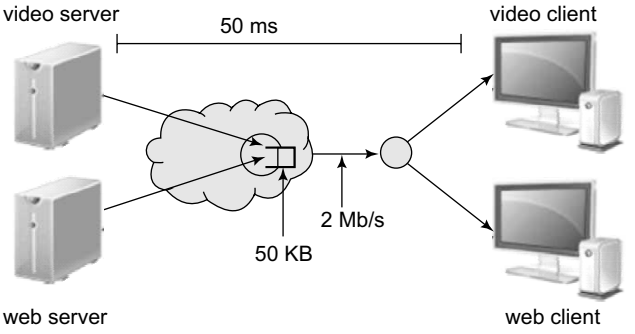
a. Packets 6 and 7 are late. We can verify this from figure.
b. The minimum playout delay is 240 ms.

■ **Example** The diagram below shows a video server and a web site that are simultaneously sending to a residential network with a 2 Mb/s download bandwidth. The one way propagation delay for both data streams is 50 ms. Assume that the video stream is 1 Mb/s and that web server is downloading a large file to the client as fast as it can using TCP.

a. If the output queue at the access router is 50 KB long and the video is sent using UDP, what is the minimum size for the playout buffer to ensure that packets never reach the video application too late to be played out?
b. How big should the playout buffer be if the video is sent using TCP, assuming no packet ever has to be sent more than three times (that is, a packet might get lost twice, but never three times). Assume the worst-case possible delay for a late packet.

■ **Answer:**

a. 2 Mb/s is 250 KB/s, so a 50 KB buffer has a max delay of 200 ms.
   Since the video rate is 125 KB/s, we need 25 KB of delay in the playout buffer.
b. The minimum delay in this case is 50 ms.
   The maximum delay is 2.5×RTT + 3×(max queuing delay)=850 ms. That gives a delay variation of 800 ms, so the playout buffer must be large enough to handle that much delay. At a video rate of 1 Mb/s, the required buffer size is.8 Mb or 100 KB.



■ **Example**

(a) Consider sending large file over TCP with no loss. Suppose TCP uses AIMD(additive increase/multiplicative decrease) for congestion control but without slow start. Assume CongWin increases by 1MSS(maximum segment size) every time an ACK is received. Assume the RTT=3s and is constant. How long does it take CongWin (congestion window) to increase from 1 MSS to 5 MSS?
(b) What is the average throughput for this connection through 12s for an MSS=2400 bytes?

■ **Answer**

(a) It takes 1 RTT to increase CongWin to 2 MSS, it takes 2 RTT to increase CongWin to 3 MSS, it takes 3 RTT to increase CongWin to 4, it takes 4 RTT to increase CongWin to 5 MSS. Therefore it takes 12 s total to increase to 5MSS.
   However, as in many of the Q's posed by the class text ambiguity arises. Assuming there is one packet per MSS one and every packet is acked leading to CongWin increasing for every Ack one could take the following answer. It takes 1 RTT to increase CongWin to 2 MSS, it takes 2 RTT to increase CongWin to 4 MSS, and it takes 3 RTT to increase CongWin to 8 (this is not how AIMD would actually work).
   Answer in this case would be 3x3RTT=9s.
   Either of above answers will be accepted.
(b) In first RTT 1 MSS was sent, in second RTT 2 MSS were sent, in second RTT 2 MSS were sent, in third RTT 3 MSS were sent, in fourth RTT 4 MSS were sent, we have a total of 1+2+3+4=10 MSS.
   Therefore average throughput was 10 MSS/ 4 RTT = 10x2400/12 =2 Kbytes/s
   However, (again using alternative view as use Ack for every MSS as increase signal) another solution is:
   In first RTT 1 MSS was sent, in second RTT 2 MSS in third RTT 4 MSS sent, in fourth RTT 8 MSS were sent, so through 4 RTT (ie 12 s) we have a total of 1+2+4+8=15 MSS.

Therefore, average throughput was 15 MSS/ 4 RTT = 15x2400/12 =3 Kbytes/s

Either of above answers will be accepted.

■ **Example** Consider a simple block cipher that uses 4 bit blocks. Use cipher block chaining with the initial vector is 1011. We xor with the data block and add 3. Retain 4 bits after discarding the overflow bits. We take this as the cipher block for next block. What is the cipher text corresponding to the clear text 0101 1011 0011.

■ **Answer:**

First, we xor the first block with the initial vector getting 1110, then add 3 giving us 0001.

Next, we xor the second block with the previous cipher text getting 1010, then add 3 giving us 1101.

Finally, we xor the third block with the previous cipher text getting 1110, then add 3 giving us 0001.

So, the complete cipher text is 0001 1101 0001.

■ **Example** Using *e = 13, d = 37,* and *n=77* in the RSA algorithm, encrypt the message "FINE" using the values of 00 to 25 for the letters A - Z. For simplicity, do the encryption and decryption letter by letter.

■ **Answer:**

a. Encryption:

$P1 = \text{"F"} = 05 \rightarrow C1 = 05^{13} \bmod 77 = 26$

$P2 = \text{"I"} = 08 \rightarrow C2 = 08^{13} \bmod 77 = 50$

$P3 = \text{"N"} = 13 \rightarrow C3 = 13^{13} \bmod 77 = 41$

$P4 = \text{"E"} = 04 \rightarrow C4 = 04^{13} \bmod 77 = 53$

b. Decryption:

$C1 = 26 \rightarrow P1 = 26^{37} \bmod 77 = 05 = \text{"F"}$

$C2 = 50 \rightarrow P2 = 50^{37} \bmod 77 = 08 = \text{"I"}$

$C3 = 41 \rightarrow P3 = 41^{37} \bmod 77 = 13 = \text{"N"}$

$C4 = 53 \rightarrow P4 = 53^{37} \bmod 77 = 04 = \text{"E"}$

■ **Example** One adaptive rate control method for broadband network management operates by requesting a source to reduce its output rate by a factor *r* every time that a congestion signal is received, and it increases its output rate by an additive amount *b* otherwise. Assuming that a source outputs at a rate of "100%" at time *t = 0.*

(a) Suppose that the source receives four congestion signals and we observe that the output rate has fallen to *60%*. Compute the reduction factor *r* in this case.

(b) Based on the result of a) above, if we receive a further two congestion signals and then no further congestion signals are sent from that point onwards. Compute the time required for the system to recover to its full *100%* output rate if *b = 5%* per *10 msec.*

■ **Answer:**

Rate is 100% at time t = 0.

(a) Rate after 4 signals would be $100r^4 = 60$. Solving for r gives, r as 0.880112

(b) After two more signals we have it as 0.4647588 = 46.5% if it now rises at the rate of 5% per 10 msec there are approximately 11 steps required to restore it to about 100% or 11 x 10 msec = 110msec to reach 100%

■ **Example** Consider an Internet router whose routing table details are given below. For the IP addresses given in parts *a* through *e*, ¯calculate the corresponding output link from the given routing table.

Address/mask	Link Interface
135.46.56.0/22	0
135.46.60.0/22	1
192.53.40.0/23	2
default	3

(a) 135.46.63.10 (b) 135.46.57.14

(c) 135.46.52.2 (d) 192.53.40.7

(e) 192.53.56.7

■ **Answer:**

a. Interface 1 as subnet maks are same.

b. Interface 0 c. Interface 3

d. Interface 2 e. Interface 3

■ **Example**

What are the differences between CSMA/CD of 802.3 Ethernet protocol and CSMA/CA of 802.11 MAC protocol in how they deal with or avoid collisions? What are the actions that they take in case the channel is sensed idle?

■ **Answer:** In CSMA/CD, transmission is aborted when collisions are detected and the sender needs to wait a random amount of time before retrying. However, in CSMA/CA, once the sender starts transmission of a frame, it transmits that frame entirely.

■ **Example** Are frame retransmissions used by wired Ethernet and wireless 802.11 standards? Why or why not?

■ **Answer:** Frame retransmissions for reliable data transfer are employed by 802.11 due to increased error and loss rate in wireless environment. Ethernet does not use frame retransmissions.

■ **Example** Allah and Bobby would like to communicate securely over a network using certificates. Assume that an intruder (Trimurthy) somehow obtained the private key of a Certificate Authority (CA). Describe in detail (including messages exchanged by Trimurthy) what kind of an attack can take place in this situation. How can the intruder intercept messages exchanged between Allah and Bobby? Assume that Allah is the one who tries to initiate communication with Bobby.

■ **Answer:** If Trimurthy obtains the private key of a CA, she can perform the man-in-the-middle attack. Suppose Allah wants to talk to Bobby: - Allah sends a message to the CA asking Bobby's public key.

- Trimurthy intercepts and since she has the CA's private key, decrypts and reads the message.
- Trimurthy generates a pair of fake public key and private key (K^+_{TB}, K^-_{TB})to be used as if they were Bobby's real keys and sends Allah back a message pretending to be from the CA.
- Allah receives the Bobby's fake public key, signs and encrypts a message using her own private key and Bobby's public key and sends it to Bobby.
- Trimurthy intercepts, decrypts the message (using Allah's public key K^+_A which can be obtained from CAs and Bobby's fake private key K^-_{TB}), reads it and modifies it if he wants to.
- Trimurthy generates another pair of fake private key and public key (K^+_{TA}, K^-_{TA}) to be used as Allah's keys, encrypts the message using this fake private key and Bobby's real public key (which can be obtained from CAs).
- Bobby receives the message, he communicates with the CA to obtain Allah's public key (to decrypt the message and possibly reply back)
- Trimurthy intercepts one more time, decrypts the message and sends back Allah's fake public key K^+_{TA}
- The man-in-the-middle attack is accomplished; from this time on, Allah and Bobby believe they communicate with each other in a secure environment, however Trimurthy can read their messages, modify them, discard (not forward) them or send fake messages.

■ **Example** Consider a server sending a 64 MB audio file to a receiver over a 1Mbps connection using packets of size 1 MB. After a packet is sent, the sender waits until an ACK packet of size 8 bytes is received before a new packet can be sent (no pipelining). Find the latency of the connection if the data transfer lasts 10 minutes in total. Assume that the packet processing delays (t_{proc}) at the sender and the receiver are negligible.

■ **Answer:** Given

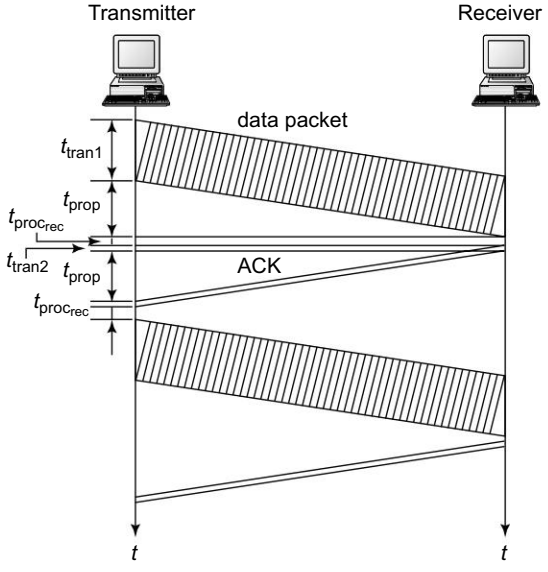
Data required to be sent=64MB audio= M = 512 Mbits

Connection speed= R = 1 Mbps;

Packet size=P=1MB= = 8 Mbits;

Time spent=T=10 minutes = 600 s;

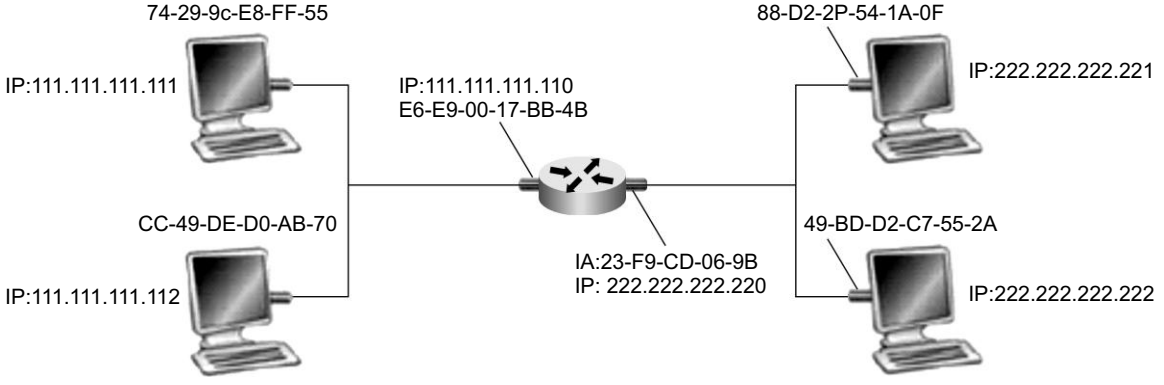
Acknowledgement packet size=A=8bytes = 64bits



Neglecting both processing delays, latency =

$$\frac{(p-A)M + PTR}{2RM} = 8.6874\text{s}$$

■ **Example** Consider the network depicted in the figure below. The IP addresses and MAC addresses of individual interfaces are as denoted in the figure.



Suppose that the sender host with the IP address 111.111.111.111 wants to send an IP datagram to the receiver host with IP address 222.222.222.222. Answer the following questions:

(a) How many subnets are there in this network? Which IP addresses belong to which subnet?

■ **Answer:**

Subnet1: 111.111.111.110, 111.111.111.111, 111.111.111.112
Subnet2: 222.222.222.220, 222.222.222.221, 222.222.222.222

(b) What is the destination IP address of the datagram when it leaves the sender host? What is the destination IP address of the datagram when it leaves the router?

■ **Answer:** In both cases 222.222.222.222

(c) What is the destination MAC address of the frame when it leaves the sender host? What is the destination MAC address of the frame when it leaves the router? Which protocol is used to determine the destination MAC address?

■ **Answer:** ARP protocol is used.
From sender: E6-E9-00-17-BB-4B, From router: 49-BD-D2-C7-56-2A

■ **Example** List 3 network performance characteristics that have a big impact on the end-to-end network QoS, in the perception of the end user.

■ **Answer:** Throughput, delay, jitter, loss, availability.

■ **Example** List 3 mechanisms that can be used in order to achieve better network QoS.

■ **Answer:** Overprovision of capacity, reservation of resources, admission control, prioritization of services.

■ **Example** Why is it said that packet switching employs statistical multiplexing? Contrast statistical multiplexing with the multiplexing that takes place in TDM.

■ **Answer:** In a packet switched network, the packets from different sources flowing on a link do not follow any fixed, pre-defined pattern. In TDM circuit switching, each host gets the same slot in a revolving TDM frame.

■ **Example** In the case of client-server applications over TCP, why the server program be executed before the client program? For the client-server applications over UDP, why the client program be executed before the server program?

■ **Answer:** With the UDP server, there is no welcoming socket, and all data from different clients enters the server through the only one available socket related to that service. With the TCP server, there is a welcoming socket, and each time a client initiates a connection to the server, a new socket is created. To support n simultaneous connections, the server would need n+1 sockets. Also, to responds to clients requests server has to start waiting for connection requests at its welcome port.

■ **Example** In BitTorrent(P2P network), suppose Ali provides chunks to Bobby throughout a 30-second interval. Will Bobby necessarily return the favor and provide chunks to Ali in the same interval? Why or why not?

■ **Answer:** It is not necessary that Bobby will also provide chunks to Ali. Ali has to be in the top 4 neighbors of Bobby for Bobby to send out chunks to her; this might not occur even if Ali provides chunks to Bobby throughout a 30 second interval.

■ **Example** What is a way using Last-Modified: header line in HTTP?

■ **Answer** The header line is used by Conditional GET to check whether the object asked has been modified.

■ **Example** Compare CRC and checksums.
Checksums have a greater probability of undetected errors than do CRCs. That is, CRCs are better at detecting errors and will result in less undetected errors than checksums. CRCs can easily be computed in hardware, but not very easily in software. Checksums can be computed in software much faster than CRCs.

■ **Example** Suppose that 1Mbps connection is shared by 5 *well behaving* TCP and 5 UDP sources. The maximum window size of the TCP sources is 50 KB. UDP sources send at a speed of 100 Mbps. What is the maximum transmission speed of UDP and TCP sources? Justify your answer?

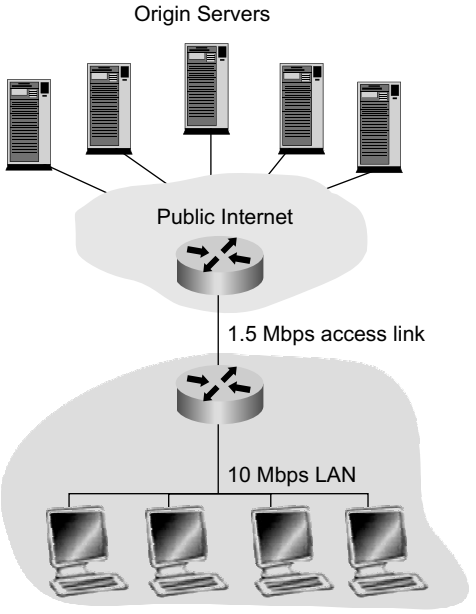
■ **Answer:** UDP sources do not have congestion control mechanism and they will send with their full speed. The TCP sources will decrease their window size, due to congestion in the link, to the minimum size (maximum segment size). Therefore, in average, 1 Mbps will be equally shared by UDP sources only. The TCP sources get almost nothing.

■ **Example** Consider the given figure, showing an institutional network connected to the Internet. Suppose that the average object size is 900,000 bits and that the average request rate from the institution's browsers to the origin servers in 1.5 requests per second. Also suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is two seconds on average.

Model the total average response time as the sum of the average access delay (that is, the delay from Internet router to institution router) and the average Internet delay. For the average access delay, use$\Delta/(1- \Delta\beta)$, where Δ is the average time required to send an object over the access link and β is the arrival rate of objects to the access link.

(a) Find the total average response time.

(b) Now suppose a cache is installed in the institutional LAN. Suppose the hit rate is 0.4. Find the total average response time.



■ **Answer:**

a. Response time T can be calculated as follows:

$$\Delta = \frac{\text{object size}}{\text{access line speed}} = \frac{0.9 \times 10^6 \text{ b}}{1.5 \times 10^6 \text{ b/s}} = 0.6$$

$$\beta = 1.5 \text{ requests/s}$$

$$T = \frac{\Delta}{1 - \beta\Delta} + 2 = 8 \text{ s}$$

b.

$$\beta_1 = 0.6 \,.\, 1.5 = 0.9 \text{ requests/s}$$

$$\beta_2 = 0.4 \,.\, 1.5 = 0.6 \text{ requests/s}$$

$$\Delta_1 = 0.6 \text{ s}$$

$$\Delta_2 = 0.09 \text{ s}$$

$$T_1 = \frac{\Delta_1}{1 - \beta_1\Delta_1} + 2 = 3.304 \text{ s}$$

$$T_2 \quad \frac{\Delta_2}{1 - \beta_2\Delta_2} = 0.0951 \text{ s}$$

Finally we can calculate response time as:

$T = 0{:}6 \ * T_1 + 0{:}4 \ * \ T_2 = 2\text{sec(App)}$

Response time for HTTP requests in this institutional network can be reduced by around 75% if a proxy is introduced, provided that the hit-rate equals to 0.4.

■ **Example** Consider an overlay network with N active peers, with each pair of peers having an active TCP connection. Additionally, suppose that the TCP connections pass through a total of K routers. How many nodes and edges are there in the corresponding overlay network? Why?

■ **Answer:** N nodes and $N(N\text{-}1)/2$ edges. The edges of the overlay network are formed by the individual TCP connections. Routers are not part of the overlay network since they operate at the lower network layer.

■ **Example** Consider a reliable data transfer protocol that uses only negative acknowledgments (NAKs). Suppose the sender sends data only infrequently. Would an NAK only protocol be preferable to a protocol that uses acknowledgments (ACKs) only? Why? Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

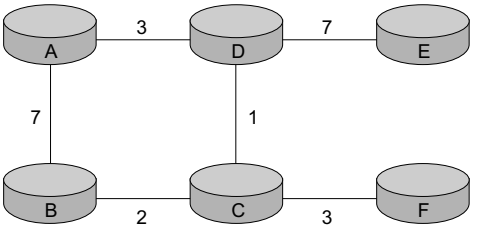
■ **Answer:** In an NAK only protocol, the loss of packet x is only detected by the receiver when packet x+1 is received. That is, the receivers receives x-1 and then x+1, only when x+1 is received does the receiver realise that x was missed. If there is a long delay between the transmission of x and the transmission of x+1, then it will be a long time until x can be recovered, under an NAK only protocol. On the other hand, if data is being sent often, then recovery under an NAK-only scheme could happen quickly. Moreover, if errors are infrequent, then NAKs are only occasionally sent (when needed), and ACK are never sent – a significant reduction in feedback in the NAK-only case over the ACK-only case.

■ **Example** What is the destination MAC address of the frame when it leaves the sender host? What is the destination MAC address of the frame when it leaves the router?Which protocol is used to determine the destination MAC address?

■ **Answer:** ARP protocol is used.

A router implements both link and network layers. A datagram (a network layer packet) keeps its destination address as it travels through routers. If a router were to change the destination IP address (e.g. to the IP address of the next hop router link interface), it would be impossible for the next router to forward the datagram to its original destination. On the other hand, the link layer destination address can be (and is) changed across different subnets. Nodes in different subnets need to make an ARP lookup to find the destination MAC address.

■ **Example** Consider the given network. The nodes in this network run the distance-vector algorithm synchronously using time slots. In a given time slot, all nodes receive distance vectors of their neighbours, update their own distance vectors, and signal the changes in their distance vectors to their neighbours. Using the distance-vector algorithm, calculate the distance vectors at node D at each time slot until there are no more distance-vector updates exchanged among routers. Assume that the nodes only know their distances to their direct neighbours initially.



■ **Answer:**

Step 1:

from \ cost to	A	B	C	D	E	F
A	∞	∞	∞	∞	∞	∞
C	∞	∞	∞	∞	∞	∞
D	3	∞	1	0	7	∞
E	∞	∞	∞	∞	∞	∞

Step 2:

from \ cost to	A	B	C	D	E	F
A	0	7	∞	3	∞	∞
C	∞	2	0	1	∞	3
D	3	3	1	0	7	4
E	∞	∞	∞	7	0	∞

And the final table of distance vectors at node D is

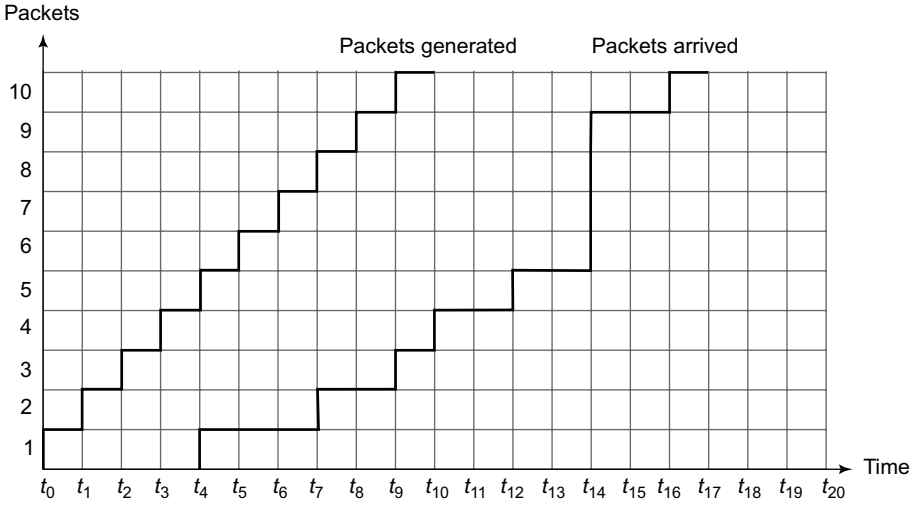
Step 3:

from \ cost to	A	B	C	D	E	F
A	0	6	4	3	10	7
C	4	2	0	1	8	3
D	3	3	1	0	7	4
E	10	10	8	7	0	11

■ **Example** The following figure shows the generation (at sender) and the arrival (at receiver) times for ten audio packets.

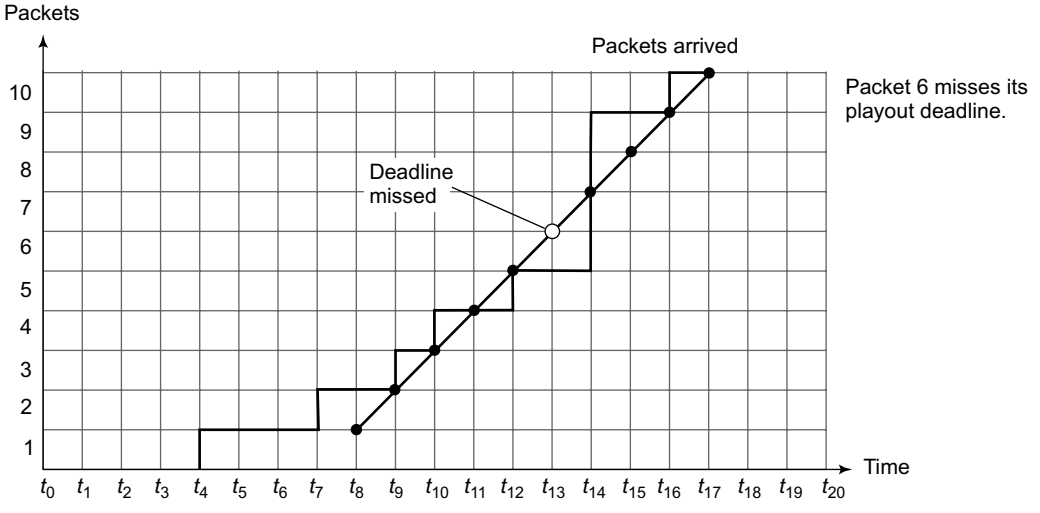
a. If we start our player at $t8$, which packets cannot be played? Show your work on the figure.

b. If we start our player at $t9$, which packets cannot be played?

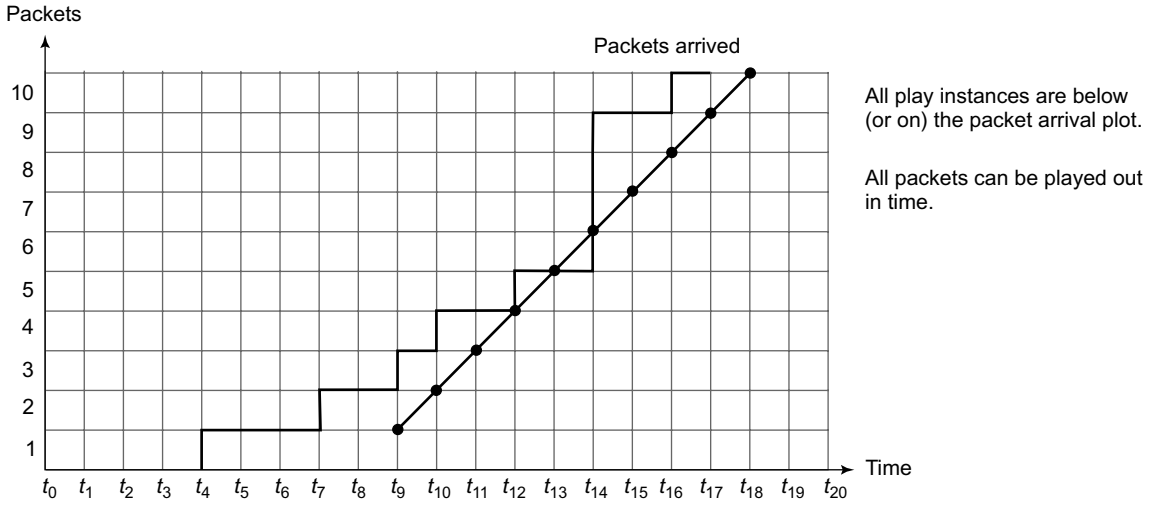


■ **Answer:**

(a) Packet 6 will be missing. See from the following figure



(b) No packet will be missing. All will be available for playing.



■ **Example** In modern packet-switched networks, the source host segments long, application-layer messages (for example, an image or a music file) into smaller packets and sends the packets into the network. The receiver then reassembles the packets back into the original message. We refer to this process as *message segmentation*. Figure illustrates the end-to-end transport of a message with and without message segmentation. Consider a message that is $8 \cdot 10^6$ bits long that is to be sent from source to destination in the figure. Suppose each link in the figure is 2 Mbps. Ignore propagation, queuing, and processing delays.

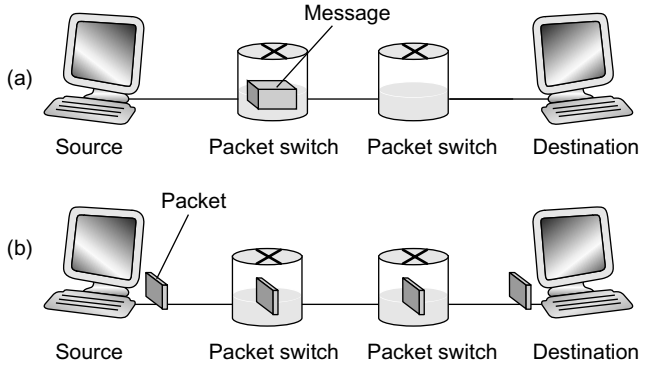


Figure End-to-End message transport: (a) without message segmentation; (b) with message segmentation.

a. Consider sending the message from source to destination *without* message segmentation. How long does it take to move the message from the source host to the first packet switch? Keeping in mind that each switch uses store-and-forward packet switching, what is the total time to move the message from source host to destination host?

Answer: Time from source host to first packet switch = $8 * 10^6 / 2 * 10^6 = 4$ sec, Total time = 4 * 3 hops = 12 secs

b. Now suppose that the message is segmented into 4,000 packets, with each packet being 2,000 bits long. How long does it take to move the first packet from source host to the first switch? When the first packet is being sent from the first switch to the second switch, the second packet is being sent from the source host to the first switch. At what time will the second packet be fully received at the first switch?

Answer: Time to send first packet from source host to first packet switch = $2 * 10^3 / 2 * 10^6 = 1$ msec. Time at which second packet is received at the first switch = time at which first packet is received at the second switch = 2 * 1 msec = 2msec

c. How long does it take to move the file from source host to destination host when message segmentation is used? Compare this result with your answer in part (a) and comment.

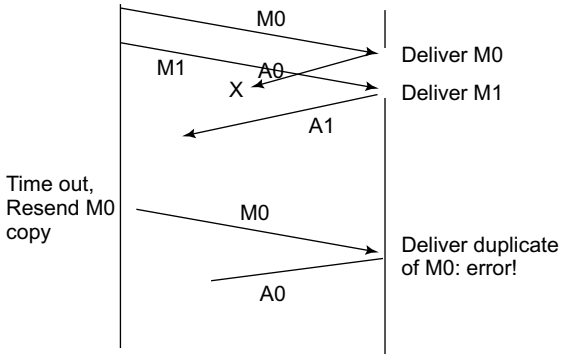
Answer: Time at which first packet is received at the destination host = 1 msec * 3 hops = 3msec. After this, every 1 msec one packet will be received; thus time at which last (4000^{th}) packet is received = 3 msec + 3999*1msec = 4.002sec. It can be seen that delay in using message segmentation is significantly less (almost 1/3).

d. Discuss the drawbacks of message segmentation.

Answer: 1. Packets have to be put in sequence at the destination, 2. the total amount of header bytes is more.

■ **Example** Consider the Go-Back-N protocol. Suppose that the size of the sequence number space (number of unique sequence numbers) is N, and the window size is N. Show (give a timeline trace showing the sender, receiver and the messages they exchange over time) that the Go-Back-N protocol will not work correctly in this case.

■ **Answer:** Suppose that the sequence number space is 0,1 and N=2, i.e., that two messages can be transmitted but not-yet-acknowledged. The timeline shows an error that can occur:



■ **Example** Explain how an FTP protocol works.

■ **Answer:** The file transfer protocol is an application-layer protocol that use TCP as its underlying transport proctor. FTP opens two TCP channels. One for control and the other is for data transfer. It works as follows. First the client sends a request to the FTP server to establish a TCP connection over port 21. This channel is used for exchanging control messages such as error messages, get/put messages and so on. When the client to get a file, for example from the server he sends his request though the control TCP channel. However, the requested file will be sent over the TCP connection established on port 20.

■ **Example** What is the purpose/use of the UDP checksum?

■ **Answer:** To detect bit error, i.e., flipped bits, in the UDP segment.

■ **Example** Assume your SW company "myNetwork" following servers:

1. DNS server: "dns.myNetwork.com" with IP as "128.119.12.39"
2. Web server: "myNetwork.com" with two IP as "128.119.12.54" and "128.119.12.52". Internet users can also access the web server by "www.myNetwork.com".
3. Email server: "mail.myNetwork.com" with IP as "128.119.12.53"

Typical email address of a user of your company looks like: "username@myNetwork.com".

(a) What resource records (RRs) do you need to provide to the upper-level ".com" Registrar?

(b) What RRs do you need to put in your company's DNS server?

■ **Answer:**

(a) Need to provide registrar with names and IP addresses of your authoritative name server. So the company needs to provide two RR records:

(myNetwork.com, dns.myNetwork.com, NS)

(dns.myNetwork.com, 128.119.12.39, A)

(b) (myNetwork.com, 128.119.12.54, A)

(myNetwork.com, 128.119.12.52, A)

(www.myNetwork.com, myNetwork.com, CNAME)

(myNetwork.com, mail.myNetwork.com, MX)

(mail.myNetwork.com, 128.119.12.53, A)

■ **Example** Assume a campus has a single access link to Internet. Assume computers in the institution send out 13 requests per second. Each object average size is 100,000 bits. Also assume the internet side delay of a request is 1 seconds. Using M/M/1 queue to model the access delay in the 1.5Mbps access link. That is to say, the average response time is $E[T]=1/(\mu-\lambda)$, where λ is the arrival rate of objects to the access link and μ is the service rate of the access link.

(a) Find the total average response time when no institutional cache is used. (Hint: total delay includes Internet delay, access link delay, and LAN delay)

(b) Now suppose the institutional cache is used. The hit rate for the cache is 0.6. Find the total average response time.

■ **Answer:**

(a) Arrival rate to access link λ = 13/sec

service rate of access link μ = 1.5Mbps /100Kbit

= 15/sec

Thus, the access link delay is = $1/(\mu-\lambda) = 1/2 = 0.5$ sec

For internal Ethernet LAN, the service rate is $\mu 1$ = 10Mbps/100Kbit = 100, thus the LAN delay is = $1/(\mu 1-l) = 1/(100-13) = 1/87 = 0.011$sec

Therefore, the total delay is = 1sec + 0.5 sec + 0.011sec =1.511sec

(b) With cache, 40% requests go outside. Thus for the 1.5Mbps access link, the arrival rate will be $\lambda = 13*0.4 = 5.2$/sec

Thus, the access link delay for the 40% requests is = $1/(\mu-\lambda) = 1/(15-5.2) = 0.102$sec

For internal Ethernet LAN, all requests must go through it no matter whether it goes out or goes to cache, thus the LAN delay is still 0.011sec

The total delay is = $0.4(1+0.102+0.011) + 0.6*0.011 = 0.452$ sec

■ **Example** Consider an enormous file L bytes from host A to host B. assume an MSS of 150 bytes, what is the maximum value of L such that TCP sequence numbers are not exhausted?

■ **Answer** There are 2^{32}= 4,294,967,296 possible sequence numbers. The sequence number does not increment by one with each segment. Rather, is incremented by the number of bytes of data sent. So the size of the MSS is irrelevant – the maximum size file that can be sent from A to B is simply the number of bytes representable by $2^{32} \approx 4.19$Gbytes.

■ **Example** Suppose all of the network sources send data at a constant bit rate. Would packet-switching or circuit-switching be more desirable in this case? Why?

■ **Answer:** Circuit-switching is more desirable here as there are no statistical multiplexing gains to be had, and by using circuits, each connection will get a constant amount of bandwidth that matched its CBR rate. On the other hand, circuit-switching has more overhead in terms of signaling, so there is an argument that packet-switching is better here since there is no call setup overhead. One can say either of them is OK.

■ **Example** Suppose if all the network sources are bursty – that they only occasionally have data to send. Would packet-switching or circuit switching be more desirable in this case? Explain.

■ **Answer:** Packet-switching is better here because there are statistical multiplexing gains – when a source does not have data to send, it will not be allocated bandwidth (that would be idle). Hence this bandwidth is available for use by other sources.

■ **Example** Describe the use of the "If-Modified-Since" header in the HTTP protocol.

■ **Answer:** When a web client or web cache has a copy of previously requested document, its GET request to the server includes an If-modified-Since line that gives the time at which the browser/cache received the copy of the document. If the document has not been modified at the web server since this time, the web server need not (and will not) send a duplicate copy of the document.

■ **Example** What does it mean when we say that control messages are "in-band"?

■ **Answer:** It means that control message and data messages may be interleaved with each other on the same connection. Indeed a single message may contain both control information and data.

■ **Example** What does it mean when we say that control messages are "out-of-band"?

■ **Answer** It means that control and data messages are carried on separate connections. One example of a protocol that has out-of-band control messages (Answer: FTP).

■ **Example** Give example protocols that has in-band control messages.

■ **Answer:** Examples includes HTTP, DNS, TCP, SMTP).

■ **Example** Consider a TCP connection between hosts A and B. Suppose that the TCP segments from A to B have source port number x and destination port number y. What are the source and destination port numbers for the segments traveling from B to A?

■ **Answer:** Source port is y, destination port is x.

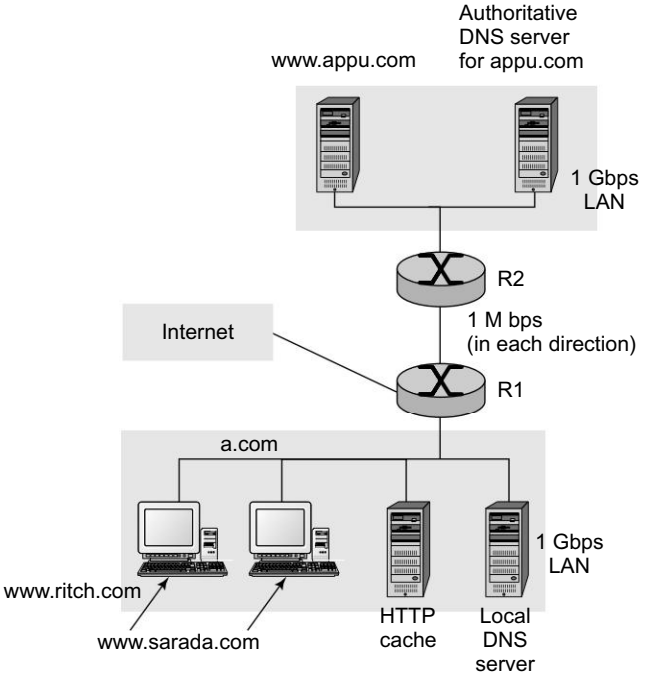
■ **Example** What is the purpose of the connection-oriented welcoming socket, which the server uses to perform an *accept()*? Once the *accept()* is done, does the server use the welcoming socket to communicate back to the client? Explain.

■ **Answer** A connection oriented server waits on the welcoming socket for an incoming connection request. When that connection request arrives a new socket is created at the server for communication back to that client.

■ **Example** Suppose a web server has 1000 ongoing TCP connections. How many server-side sockets are used? How many server-side port numbers are used? Explain.

■ **Answer:** If there are 1000 ongoing connections, and nothing else happening on the server, there will 1001 sockets in use – the single welcoming socket and the 1000 sockets in use for server-to-client communication. The ONLY server-sideport number in use at the server will be the single port number associated with the welcoming socket, e.g., port 80 on a web server.

■ **Example** Consider the networks shown in the figure below. There are two user machines www.ritch.com and www.sarada.com in the network a.com. Suppose the user at www.ritch.com types in the URL www.appu.com/bigfile.htm into a browser to retrieve a 1Gbit (1000 Mbit) file from www.appu.com.



a. List the sequence of DNS and HTTP messages sent/received from/by www.ritch.com as well as any other messages that leave/enter the a.com network that are not directly sent/received by www.ritch.com from the point that the URL is entered into the browser until the file is completely

received. Indicate the source and destination of each message. You can assume that every HTTP request by www.ritch.com is first directed to the HTTP cache in a.com and that the cache is initially empty, and that all DNS requests are iterated queries.

- *www.ritch.com needs to resolve the name www.appu.com to an IP address so it sends a DNS REQUESTmessage to its local DNS resolver (this takes no time given the assumptions below)*
- *Local DNS server does not have any information so it contacts a root DNS server with a REQUEST message (this take 500 ms given the assumptions below)*
- *Root DNS server returns name of DNS Top Level Domain server for.com (this takes 500 ms given the assumptions below)*
- *Local DNS server contacts.com TLD (this take 500 ms given the assumptions below)*
- *TLD.com server returns authoritative name server for appu.com (this takes 500 ms given the assumptions below)*
- *Local DNS server contacts authoritative name server for appu.com (this takes 100 ms given the assumptions below)*
- *Authoritative name server for appu.com returns IP address of www.appu.com. (this takes 100 ms given the assumptions below)*
- *HTTP client sends HTTP GET message to www.appu.com, which it sends to the HTTP cache in the a.com network (this takes no time given the assumptions).*
- *The HTTP cache does not find the requested document in its cache, so it sends the GET request to www.appu.com. (this takes 100 ms given the assumptions below)*
- *www.appu.com receives the GE request. There is a 1 sec transmission delay to send the 1Gbps file from www.appu.com to R2. If we assume that as soon as the first few bits of the file arrive at R1, that they are forwarded on the 1Mbps R2-to-R1 link, then this delay can be ignored.*
- *The 1 Gbit file (in smaller packets or in a big chunk, that's not important here) is transmitted over the 1 Mbps link between R2 and R1. This takes 1000 seconds. There is an additional 100 ms propagation delay.*
- *There is a 1 sec delay to send the 1Gbps file from R1 to the HTTP cache. If we assume that as soon as the first few bits of the file arrive at the cache, that they are forwarded to the cache, then this delay can be ignored.*
- *There is a 1 sec delay to send the 1Gbps file from the HTTP cache to www.ritch.com. If we assume that as soon as the first few bits of the file arrive at the cache, that they are forwarded to the cache, then this delay can be ignored.*
- *The total delay is thus:.5 +.5 +.5 +.5 +.1 +.1 + 1 + 1000 +1+1 = 1105.2 secs (1002.2 is also an OK answer).*

(Note that we have neglected to account for TCP hand-shaking delays for the HTTP exchanges.)

b. Now assume that machine www.sarada.com makes a request to exactly the same URL that www.ritch.com made. List the sequence of DNS and HTTP messages sent/received from/by www.sarada.com as well as any other messages that leave/enter the a.com network that are not directly sent/received by www.sarada.com from the point that the URL is entered into the browser until the file is completely received. Indicate the source and destination of each message. [Hint: make sure you consider caching here]

- *www.sarada.com needs to resolve the name www.appu.com to an IP address so it sends a DNS REQUEST message to its local DNS resolver (this takes no time given the assumptions above)*
- *The local DNS server looks in its cache and finds the IP address for www.appu.com, since www.ritch.com had just requested that that name be resolved, and returns the IP address to www.sarada.com. (this takes no time given the assumptions above)*
- *HTTP client at www.sarada.com sends HTTP GET message to www.b1.com, which it sends to the HTTP cache in the a.com network (this takes no time given the assumptions).*
- *The HTTP cache finds the requested document in its cache, so it sends a GET request with an If-Modified-Since to to www.appu.com. (this takes 100 ms given the assumptions)*
- *www.appu.com receives the GET request. The document has not changed, so www.appu.com sends a short HTTP REPLY message to the HTTP cache in a.com indicating that the cached copy is valid. (this takes 100 ms given the assumptions)*
- *There is a 1 sec delay to send the 1Gbps file from the HTTP cache to www.sarada.com.*
- *The total delay is thus:.1 +.1 + 1 = 1.2 secs*

c. Now suppose there is no HTTP cache in network a.com. What is the maximum rate at which machines in a.com can make requests for the file www.appu.com/bigfile.htm while keeping the time from when a request is made to when it is satisfied non-infinite in the long run? *(Answer: since it takes 1000 secs to send the file from R2tro R1, the maximum rate at which requests to send the file from appu.com to a.com is 1 request every 1000 seconds, or an arrival rate of.001 requests/sec.)*

■ **Example** Why did we need to introduce sequence numbers? Also, why did we need to introduce timers?

■ **Answer:** Sequence numbers are required for a receiver to find out whether an arriving packet contains new data or is

a retransmission. To handle losses in the channel timers are used. If the ACK for a transmitted packet is not received within the duration of the timer for the packet, the packet (or its ACK or NACK) is assumed to have been lost. Hence, the packet is retransmitted.

■ **Example** Consider a datagram network using 8-bit host addresses. Suppose a router uses longest prefix matching and has the following forwarding table:

Prefix Match	Interface
1	0
11	1
111	2
otherwise	3

For each of the four interfaces, give the associated range of destination host addresses and the number of addresses in the range.

■ **Answer:**

Destination Address Range	***Link Interface***
10000000	
through (64 addresses)	*0*
10111111	
11000000	
through(32 addresses)	*1*
11011111	
11100000	
through (32 addresses)	*2*
11111111	
00000000	
through (128 addresses)	*3*
01111111	

■ **Example** Do the routers in both datagram network and virtual-circuit networks use forwarding tables? If so, describe the forwarding tables for both classes of networks.

■ **Answer:** Yes, both use forwarding tables. For a VC forwarding table, the columns are : Incoming Interface, Incoming VC Number, Outgoing Interface, Outgoing VC Number. For a datagram forwarding table, the columns are: Destination Address, Outgoing Interface.

■ **Example** Why is an ARP query sent with a broadcast frame? Why is an ARP response sent within a frame with a specific destination MAC address?

■ **Answer:** An ARP query is sent in a broadcast frame because the querying host does not which adapter address corresponds to the IP address in question. For the response, the sending node knows the adapter address to which the response should be sent, so there is no need to send a broadcast frame (which would have to be processed by all the other nodes on the LAN).

■ **Example** Define switches and routers and explain pros (at least two items) of switches comparing with routers.

■ **Answer:** A router is a layer-3 switch and a switch is a layer-2 packet switch. The pros are switch's plug-and-play, high filtering and forwarding.

■ **Example** Why are acknowledgements used in 802.11 but not in wired Ethernet?

■ **Answer:** The adapter would not be able to detect all collisions due to hidden terminal problem and fading.

■ **Example** Define terminologies such as streaming, packet jitter, policing and real-time.

■ **Answer:**

Streaming: playing-out while downloading a file
Packet jitter: the variation of packet delay.
Policing: the regulation of the rate at which a class or flow is allowed to inject packets into the network
Real-time: response times will be very less.

■ **Example** What are the differences between message confidentiality and message integrity? Can you have one without the other? Justify your answer.

■ **Answer:** Confidentiality is the property that the original plaintext message cannot be determined by an attacker who intercepts the ciphertext-encryption of the original plaintext message. Message integrity is the property that the receiver can detect whether the message sent (whether encrypted or not) was altered in transit. The two are thus different concepts, and one can have one without the other. An encrypted message that is altered in transmit may still be confidential (the attacker cannot determine the original plaintext) but will not have message integrity if the error is undetected. Similarly, a message that is altered in transit (and detected) could have been sent in plaintext and thus would not be confidential.

■ **Example** What is the purpose of a nonce in an end-point authentication protocol?

■ **Answer:** A nonce is used to ensure that the person being authenticated is "live." Nonces thus are used to combat playback attacks.

■ **Example** Compare and contrast between Connection-Oriented communication service and Connectionless communication service?

■ **Answer:** Some of the principle characteristics of the connection-oriented service are:

- Two end-systems first "handshake" before either starts to send application data to the other.
- Provides reliable data transfer, i.e., all application data sent by one side of the connection arrives at the other side of the connection in order and without any gaps.
- Provides flow control, i.e., it makes sure that neither end of a connection overwhelms the buffers in

the other end of the connection by sending to many packets to fast.
- Provides congestion control, i.e., regulates the amount of data that an application can send into the network, helping to prevent the Internet from entering a state of grid lock.

The principle characteristics of connectionless service are:
- No handshaking
- No guarantees of reliable data transfer
- No flow control or congestion control

■ **Example** Explain about network edge, network access, network core.

Answer:
1. Network Edge: where workstations, servers, etc. are hosting the application
2. Network Core : the backhaul networks where routers that facilitate the delivery of data between the network edges
3. Network Access : the physical medium and network access technologies such as dial-up, ADSL, HFC that facilitate the interconnection between the network edge and access router

■ **Example** Consider the web browsing application over unreliable channel. For designing a reliable data transmission protocol, numerate three events that may happen while transmitting which will affect the reliability of your link and how your protocol will response.

■ **Answer** For ensuring reliable data transfer, we need to deliver the whole date correctly and in-order. When we have unreliable channel the may expect packets loss due to either buffer overflow or packet damage. Original packet might be lost or ACKs also might be lost. Then we need to equip our reliable protocol with the following control/information:

Event	Control/information
Packet loss	**Error detection/correction** **Retransmission mechanisms** **Sequence number** **Acknowldgment for previous succesfully received packets**
ACK loss	**Time out procedure** **Sequence number**

■ **Example** Distinguish between Network application and Application-layer protocol.

■ **Answer** The network application is composed of many pieces. For example, the web is a network application which includes several pieces such as standard formats, web browsing. However, the application layer protocol is just one piece that defines the rules for certain communications among application-layer peers.

■ **Example** What is a P2P network? Does it include routers? Why?

■ **Answer:** P2P is a logical network where every peer can acts as a client and as a server. It is an overlay network does not include routers.

■ **Example** Locating the content in P2P network is a non-trivial problem. What do we mean by "locating content"? And why is it a problem?

■ **Answer:** Locating content in the P2P context means how to find the IP address of the peer that can provide this content. It is a problem because the uses are not Always-on. Further, every time a peer is on the net, he may have different IP address.

■ **Example** What do we mean by an overlay network and how is Gnutella overlay network created and maintained?

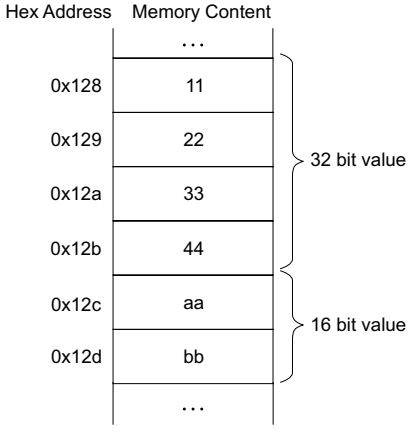
■ **Answer:** The overlay network in a P2P file sharing system consists of the nodes participating in the file sharing system and the logical links between the nodes. With Gnutella, when a node wants to join the Gnutella network, it first discovers ("out of band") the IP address of one or more nodes already in the network. It then sends join messages to these nodes. When the node receives confirmations, it becomes a member of the of Gnutella network. Nodes maintain their logical links with periodic refresh messages.

■ **Example** Briefly explain how TCP demultiplexing takes place?

■ **Answer:** The demultiplexing function takes place at the receiver side at the transport layer. In particular for a TCP connection, the demultiplexing needs to check 4-tuple that constitutes a unique logical TCP connection between the client and the server. These, 4-tuple are source IP address, source port number, destination IP address and destination port number.

■ **Example** In a computer memory an integer (4byte) and a short (2byte) values are stored as shown below. Assuming that the computer is little endian style, how this information is sent over internet? That is, what is network order of this integer and short? Assume numbers in the memory are given in Hex.

■ **Answer:** 44, 33, 22, 11, bb, aa



■ **Example** State the four functions of the data link layer.

■ **Answer:**

a. Framing
b. Reliable data transfer
c. Flow control
d. Channel access control

■ **Example** MAC Protocols can be classified into three broad classes. State these classes and give an example for each one.

■ **Answer:**

a. Channel partitioning (e.g. CDMA, FDMA, TDMA)
b. Random access (e.g. ALOHA, CSMA, etc.)
c. Token turns (e.g. token ring, polling)

■ **Example** What is the access methodology used in Ethernet technology? (please note abbreviation alone is not accepted)

■ **Answer:**

CSMA/CD (Carrier Sense Multiple Access /Collision Detection)

■ **Example** In order to have efficiency of 0.6, what should be the distance between hub and a node in a 100baseT Ethernet LAN. Assume propagation delay is 200m/μsec and frame size is 64 bytes.

■ **Answer:** $t_{prop}=d/200x10^8$

$t_{\_tran}$ = 64bytes/100Mbps = $64*8/100*10^6$ = 5.12μs

efficiency = 1/(1+5a) = 0.6

Therefore, a = 0.133 = t_{prop}/t_{tran}

$0.133 = d/200x10^8/ 5.12\mu s$

Therefore, distance d = 184m

■ **Example** Distinguish from the operational point of view between CSMA/CA and CSMA/CD protocols. Also, explain why CSMA/CD is difficult to apply in wireless environments.

CSMA/CA is carrier sense multiple access/collision avoidance

CSMA/CD is carrier sense multiple access / collision detection. Then CSMA/CA is based on avoiding the collision and detection the collision. Therefore, the protocol takes several measures to assure avoiding collision.

- Using CSMA (by sensing before transmission)
- Using reservation through special requests, namely "ready-to-send" (RTS) and "clear-to-send" (CTS) where the first one is sent by the sending mobile while the other is a response from the receiving mobile.
- Using different Inter frame gaps (DIFS, SIFS)
- Random backoff when collision occurs

Also CSMA/CS requests the receiver to send ACK frame to acknowledging the reception of the frame. This ACK is not part of CSMA/CD

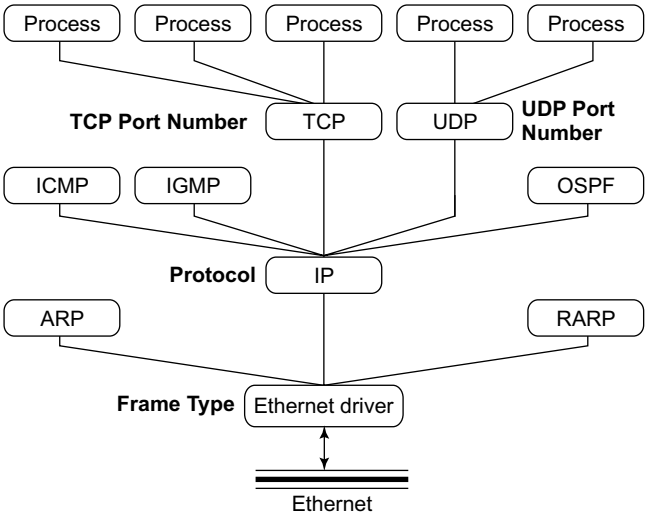
On the other hand, CSMA/CD is design such that collision can be detected therefore, there are restrictions on the length of the frame, network segment length and so on.

CSMA/CD is very difficult to implement is wireless network because of the following

- Very noisy channel
- Reception/transmission should be carried out on the same channel to be able to detect the collision and this is very difficult to implement is wireless network.

■ **Example** Illustrate with a figure where multiplexing and de-multiplexing takes place in the internet layers.

■ **Answer:**



Multiplexing/demultiplexing in the layers.

■ **Example** Explain how multiplexing is done at IP layer?

■ **Answer** In the IP case, each protocol using IP is assigned a unique protocol number, which is carried in the Protocol IP header field in every packet generated by the protocol. By examining the value of this field of an incoming IP datagram, the type of payload can be determined.

■ **Example** Explain how multiplexing is done at DLL?

■ Answer Frame Type

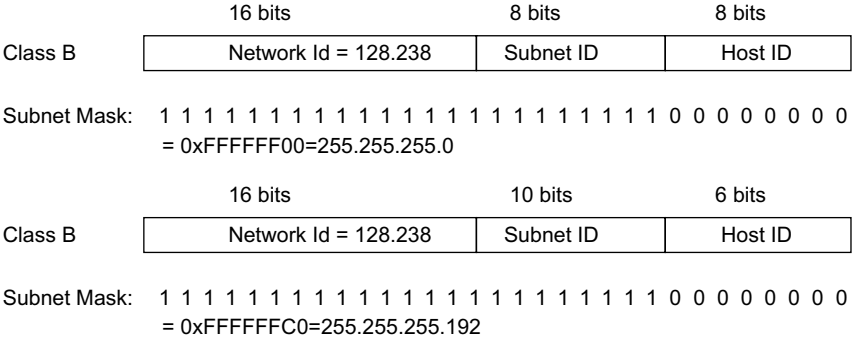
■ **Example** What is the use of subnetting?

■ **Answer:** In order to provide the flexibility in network administration and operation, the subnetting technique was introduced, where an IP address is further divided into three levels: a network ID, a subnet ID, and a host ID. With subnetting, IP addresses can be assigned using a finer granularity, e.g.,a small organisation can be assigned a subnet address that just satisfies its requirement. In addition, with subnetting, an organisation can divide its assigned network

space into a number of subnets, and assign a subnet to each department. The subnets can be interconnected by routers, resulting in better performance, stronger security, and easier management.

■ **Example** Subnet masks of two class B type of networks are given as: 255.255.255.0 and 255.255.255.192. Find out which one supports more number of subnets.

■ **Answer:** If we present these masks in binary fashion, least significant 0's sequence indicates host bits. See the following figure. Also, we know they are class B type addresses, most significant 16 bits indicates the network address. Thus, we know that for first type 8 bits are used for subnetting while for the other 10 bits are used. Thus, second network will be having 2^{10}=1024 subnets.



■ **Example** Explain about error detection that is used in IP, ICMP, IGMP, UDP and TCP.

■ **Answer:** Given protocols use the checksum algorithm (or parity check) to detect bit errors in the header of a received packet. Suppose the checksum header field is K bits long (e.g., K = 16 in IP, UDP, and TCP). The value of the field is first set to 0. Then, the K-bit one's complement sum of the header is computed, by considering the header as a sequence of K-bit words. The K-bit one's complement of the sum is stored in the checksum field and sent to the receiver. The receiver, after receiving the packet, calculates the checksum over the header (including the checksum field) using the same algorithm. The result would be all ones if the header is error free. Otherwise, the header is corrupted and the received packet is discarded.

■ **Example** Explain about error detection in Ethernet.

■ **Answer:** CRC

■ **Example** What is SACK?

■ **Answer:** In practice in TCP layer, a window of TCP segments may be sent and received before an acknowledgement is received by the sender. If multiple segments in this window of segments are lost, the sender has to retransmit the lost segments at a rate of one retransmission per round trip time (RTT), resulting in a reduced throughput. To cope with this problem, TCP allows the use of *selective acknowledgement* (SACK) to report multiple lost segments. While a TCP connection is being established, the two ends can use the *TCP Sack-Permitted* option to negotiate if SACK is allowed. If both ends agree to use SACK, the receiver uses the *TCP Sack* option to acknowledge all the segments that has been successfully received in the last window of segments, and the sender can retransmit more than one lost segment at a time.

■ **Example** Consider a pipelined, reliable transport protocol that uses go-back-N with cumulative acknowledgments. Assume that timeouts trigger retransmissions (duplicate ACKs do not) and that the receiver does not maintain any receive buffer.

a. If the one-way delay between the sender and receiver is 50 ms and every packet is 10,000 bits long, how big must the window be to allow the sender to send at a steady rate of 1 Gb/s under ideal conditions?

■ **Answer:** RTT=2*50ms=0.1 second, so a 1 Gb/s link sends 100M bits per RTT or 10K packets per RTT. So the window size must be at least 10,000 to support a 1 Gb/s rate.

b. Suppose that approximately one packet in 100,000 is lost. If the sender uses a timeout of 500 ms and a window size of 20,000 packets, how often does sender experience a timeout? How many packets will it retransmit when a time out occurs?

■ **Answer:** Assuming that the bottleneck link rate is 1 Gb/s, the sender can still only send 10K packets per RTT. After each loss, it takes half a second for the sender to detect the loss and all packets sent in that half second are effectively wasted (since the receiver discards them in go-back-N). But the window size limits the number of packets sent following the lost packet to 20K. So immediately after each loss, the sender sends 20K packets, pauses for.3 seconds then resends the first 20K packets before sending another 60K, at which point it loses another packet. So, the sender experiences a timeout every 1.3 seconds.

If we do not assume a 1 Gb/s bottleneck link rate, the sender can send 20K packets per RTT. In this case, after

sending the packet that gets lost, it again times out after half a second, before re-sending 20K packets plus 60K new ones (which takes 400 ms). So, the sender has a timeout every 900 ms.

c. Estimate the throughput for this connection, assuming one packet in 100,000 is lost.

■ **Answer:** Assuming a 1 Gb/s bottleneck link rate, the receiver gets 80K new packets every 1.3 seconds, so the throughput is (8/13) Gb/s which is about 620 Mb/s.

If we do not assume a 1 Gb/s bottleneck link rate, the receiver gets 80K new packets every. 9 seconds, so the throughput is (8/9) Gb/s, which is about 890 Mb/s.

■ **Example** A TCP transmission host A to host B, after receiving the ACK with ack number 20, A sends packets with sequence numbers 20, 30, 40, 50, 60, 70, 80, 90 and 100. Sometime later, it receives ACKs with sequence numbers 40, 40, 60, 60, 60, 60, 60. Assume that A sends no additional data segments in the meantime.

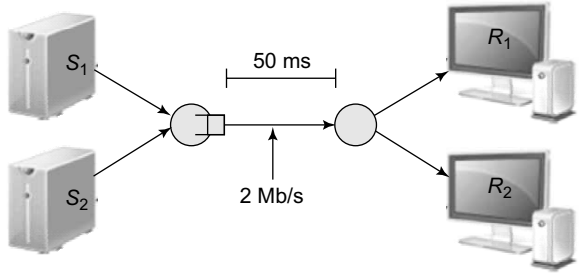
a. What sequence number would you expect to see in the next packet sent by A?

■ **Anwer:** 60

b. What ACK number would you expect in the next ACK? You may assume that all packets sent by A carry 10 bytes of data.

■ **Answer:** 110

■ **Example** The following diagram shows two TCP senders at left and the corresponding receivers at right. Both senders use TCP Reno protocol. Assume that the MSS is 1 KB, that the one-way propagation delay for both connections is 50 ms and that the link joining the two routers has a bandwidth of 2 Mb/s. Let $cwnd_1$ and $cwnd_2$ be the values of the senders' congestion windows and assume that $cwnd_1 = cwnd_2$. What is the smallest value of $cwnd_i$ for which the link joining the two routers stays busy all the time?



■ **Answer:**

We need 200 Kbits per RTT to keep the link busy, or 100 Kbits per sender. That means 12.5 KB.

Assume that the link buffer overflows whenever $cwnd_1+cwnd_2 \geq 36$ KB and that at time 0, $cwnd_1=12$ KB and $cwnd_2= 24$ KB. Approximately, what are the values of $cwnd_1$ and $cwnd_2$ one RTT later? Assume that all packet losses are detected by a triple duplicate ack.

■ **Answer:** 6 KB and 12 KB

a. How many RTTs pass before $cwnd_1+cwnd_2=36$ again? What are the values of $cwnd_1$ and $cwnd_2$ at this point?

■ **Answer:** After 9 more RTTs, we have $cwnd_1= 15$ KB and $cwnd_2 = 21$ KB.

That is, currently $cwnd_1$ and $cwnd_2$ values are 6 and 12. After first RTT, they become 7,13; second RTT they become 8,14. Thus, after 9^{th} RTT they become 15 and 21.

b. Approximately, how many RTTs pass (in total) before $cwnd_2 - cwnd_1 < 2$ KB?

■ **Answer:** Initially, $cwnd_1$ and $cwnd_2$ are 12 and 24. Thus, in the next RTT their values become 6 and 12. After 9 RTTs, they become 15 and 21. In the next RTT, their values become 7 and 10. Their difference is 3 KB and it remains 3 KB for another 9 RTTs when the buffer fills again, at which point their values are 16 and 19. In the next RTT, they will having their difference as 1.5 KB(16/2,19/2). So, approximately 21 RTTs pass before the difference in the congestion windows drops below 2 KB.

■ **Example** A packet switch receives a packet and determines the outbound link to which the packet should be forwarded. When the packet arrives, one other packet is halfway done being transmitted on this outbound link and four other packets are waiting to be transmitted. Packets are transmitted in order of arrival. Suppose all packets are 1,200 bytes and the link rate is 3 Mbps. What is the queuing delay for the packet? More generally, what is the queuing delay when all packets have length L, the transmission rates is R, x bits of the currently-being-transmitted packet have been transmitted, and n packets are already in the queue?

■ **Answer:** The arriving packet must first wait for the link to transmit 5,400 bytes or 43,200 bits. Since these bits are transmitted at 3 Mbps, the queuing delay is 14.3 msec. Generally, the queuing delay is (nL + (L - x))/R.

■ **Example** Consider sending a large file of F bits from Host A to Host B. There are three links (and two switches) between A and B, and the links are uncongested (that is, no queuing delays). Host A segments the file into segments of S bits each and adds 80 bits of header to each segment, forming packets of $L=80 + S$ bits. Each link has a transmission rate of R bps. Find the value of S that minimizes the delay of moving the file from Host A to Host B. Disregard propagation delay. Start solving this problem by calculating the overall delay first.

■ **Answer:** Time at which the 1st packet is received at the destination $\frac{S+80}{R} \times 3$ sec. After this, one packet is re-

ceived at destination every $\frac{S+80}{R}$ sec. Thus delay in sending the whole file =

$$\text{delay} = \frac{S+80}{R} \times 3 + \left(\frac{F}{S} - 1\right) \times \left(\frac{S+80}{80}\right) = \frac{S+80}{R} \times \left(\frac{F}{S} + 2\right)$$

To calculate the value of S which leads to the minimum delay,

$$\frac{d}{ds} delay = 0 \Rightarrow S = \sqrt{40F}$$

■ **Example** Define "flow control" and briefly explain how TCP implements flow control.

■ **Answer:** Flow control assures that the receiver's buffer does not overflow. Under the TCP protocol the receiver notifies the sender of the current remaining free space in its receive buffer using the receive window field in the TCP header. The sender insures that the receive buffer doesn't overflow by guaranteeing that the total size of the sent, but unacknowledged segments is less than the value of the receive window in the last segment that the sender received.

■ **Example** Answer the TCP connection establishment protocol related questions.

a. How many segments are sent?

■ **Answer:** 3

b. Which of these segments are client initiated?

■ **Answer:** 1, 3

c. In the first segment, which flag bits are set?

■ **Answer:** SYN

■ **Example** Consider hosts A and B communicating over a TCP connection. Assume unrealistically that the initial sequence number for each of A and B is 0. Assume that all segments sent between A and B have 20 byte headers. A sends B a segment with a 20 byte payload, B responds with a segment with a 30 byte payload and then another segment with a 40 byte payload, and finally A responds with a segment with a 50 byte payload. Give the value of the sequence number field and acknowledgement number fields for each segment.

■ **Answer:**

Segment 1: Seq = 0 Ack = 0

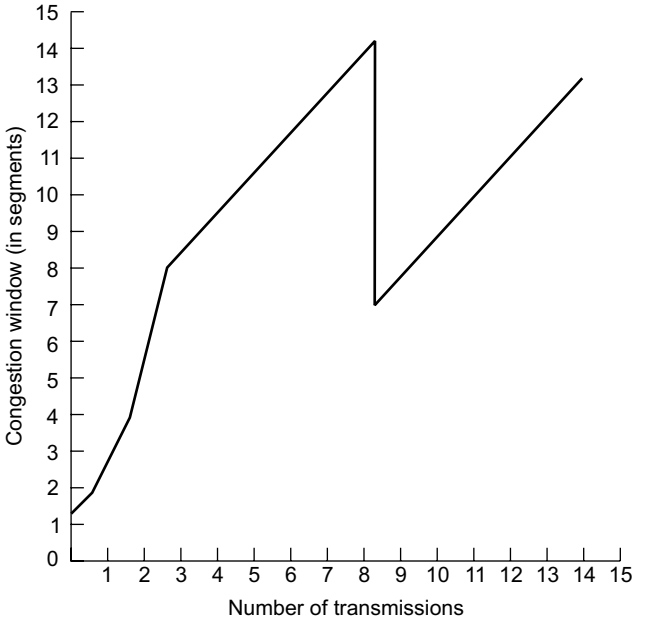
Segment 2: Seq = 0 Ack = 20

Segment 3: Seq = 30 Ack = 20

Segment 4: Seq=20 Ack =70

■ **Example** Congestion window size for a TCP session that uses TCP Reno is 8 and that a loss event will occur when the window size is 14. Show, how congestion window size changes with a figure.

■ **Answer:** The following figure illustrates the working. After 6 RTT's, window size becomes 14. At which point, congestion window is adjusted to 7.



■ **Example** Consider TCP congestion control. Assume we have a round trip time RTT of 2 seconds. Assume that the segment size is 1 kilobyte. Assume that the bandwidth of the connection is 100 kilobits per second. What is the smallest window size for which there will be no stalling?

■ **Answer:**

Using the equation WS/R > RTT + S/R

W 8000/100,000 = 2 + 8000/100,000 for W. The solution to this is W=26.

■ **Example** Assume that you have 3 long lived TCP connections over a single bottleneck link with bandwidth R. The average (over a long time) bandwidth that each connection receives should be about how much?

■ **Answer:** The average of R/3 and (R/3)/2, which is R/4.

■ **Example** Consider two hosts A and B connected by a single link of rate C bits per second. Suppose that the two hosts are separated by d meters. Suppose that the speed of communication on the link is s meters per second. Host A has to send to B a packet of size K bits.

a. What is the propagation delay in terms of C, d, s, and K?

■ **Answer:** d/s

b. What is the transmission delay in terms of C, d, s, and K?

■ **Answer:** K/C

c. What is the end to end delay in terms of C, d, s, and K?

■ **Answer:** d/s + K/C

■ **Example** Assume that you have a base html file with 30 embedded images that is requested by a client. Assume that the base file and all of the images are small enough to fit within one TCP segment. How many round trips are required to retrieve the base file and the images under the

following settings? Assume that the round trip times dominate all other times.

a. HTTP 1.0 with no parallel connections

■ **Answer:** 2 RTT per file for a total of 62

b. HTTP 1.0 with up to 10 parallel connections

■ **Answer:** 2 RTT for the base file, plus 6 RTT for the embedded images, for a total of 8.

c. HTTP 1.1. with no pipelining

■ **Answer:** 2 RTT for the base file, plus 1 RTT for each of the remaining files, for a total of 32.

d. HTTP 1.1. with pipelining

■ **Answer:** 2RTT for the base file, and 1 RTT for the remaining files, for a total of 3 RTT.

■ **Example** Assume that a TCP process A first measures the actual round trip time to another TCP process to be 30 ms, and A thus sets its estimated round trip time to be 30 ms. The next actual round trip time that A sees is 60 ms. In response A increases its estimated round trip to 50 ms. The next actual round trip time that A sees is 40 ms. What is the next estimated round trip computed by A? Justify your answer.

■ **Answer:** TCP uses an exponential weighted moving average, that is,

New EstimatedRTT = x OldEstimatedRTT + (1- x) NewObservedRTT.

So when the estimated RTT of 50 is calculated, we know that 50 = x 30 + (1-x) 60, so x=1/3. Plugging into this formula to compute the new estimated RTT gives

1/3 50 + 2/3 40 = 130/3 = 40 1/3

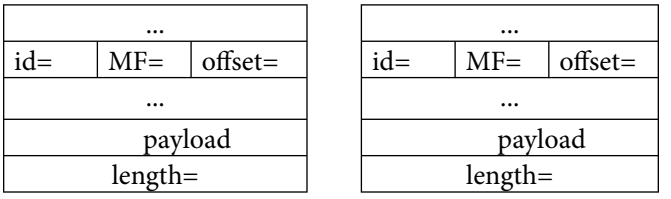
■ **Example** Assume a server A creates sockets by supplying it IP address 1.2.3.5 and port number as 81 to the Socket system call. Then it calls bind, listen system calls. Another host creates socket and calls connect system call by supplying 1.2.3.5 and 81 as parameters. One more host also does the same. Assume on A, connection oriented service is extended. How many sockets are in total created on A, and how many ports are in use on A?

■ **Answer:** 3 and 1

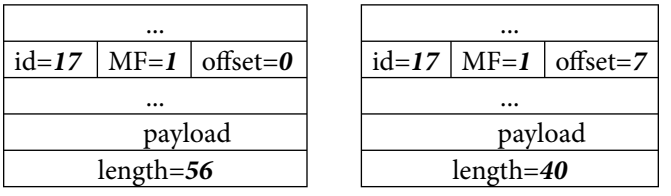
■ **Example** Consider a packet with a total length of 250 bytes (including IPv4 header, with no options) and an id field equal to 17, sent from a host *A* to a host *B*, passing through routers *X* and *Y*. Assume that the subnet where host *A* is connected has an MTU of 500 bytes, the subnet where host *B* is connected has an MTU of 80 bytes and the subnet between *X* and *Y* has an MTU of 120 bytes. Assuming that the "don't fragment" flag is not set, how many fragments does router *X* divide the packet into? What is the length of each fragment?

Answer: The payload has 230 bytes and each packet in the subnet joining X and Y has room for a 100 byte payload, but since 100 is not divisible by 8, we'll have 96 bytes in the initial fragments. So, there are 3 fragments with payload sizes of 96, 96 and 38 giving packet sizes of 116, 116 and 58.

a. Complete the diagram below so that it represents the first two fragments forwarded by router *Y* (*not X*). Fill in all the blanks.



■ **Answer:** First two fragments are the one which can be formed by dividing first fragment having payload length of 96 and coming out from X. As, MTU in the network in which host B is available is 80, this 96 one has to be divided. Do remember that payload sizes has to be multiples of 8 bytes. Thus, 96 is divided into 56 and 40. According two fragments are created with the following field values.



b. Suppose host *A* sends a 50 byte packet with the "don't fragment" flag set. Explain what happens to this packet at each of the two routers.

■ **Answer:** This packet is shorter than the MTU of all subnets, so it will be delivered to B without any fragmentation.

■ **Example** Suppose a TCP message that contains 2048 bytes of data and 20 bytes of TCP header is passed to IP for delivery across two networks of the Internet. The first network uses 14 byte headers and has an MTU of 1024 bytes; the second uses 8-byte headers with an MTU of 512 bytes. Each network's MTU gives the size of the largest IP datagram that can be carried in a link-layer frame. Give the ***sizes*** and ***offsets*** of the sequence of fragments delivered to the network layer at the destination host. Assume all IP headers are 20 bytes. Note, the IP requires that fragmentation should always happen on 8-byte boundaries.

■ **Answer:** Consider the first network. Packets have room for 1024 – 20 = 1004 bytes of IP-level data; because 1004 is not a multiple of 8 each fragment can contain at most 8× floor((1004/8)) = 1000 bytes. We need to transfer 2048 + 20 = 2068 bytes of such data. This would be fragmented into fragments of size 1000, 1000, and 68.

Fragment	Size	Offest
1	1000	0
2	1000	1000
3	68	2000

Over the second network, the 68-byte packet would be unfragmented but the 1000-data-byte packet would be fragmented as follows. The IP header is 20 bytes, leaving 512-20 = 492 bytes for IP-level data. Again rounding down to the nearest multiple of 8, each fragment could contain 488 bytes of IP-level data. 1000 bytes of such data would become fragments with data sizes 488, 488, and 24.

Fragment	Size	Offest
1	488	0
2	488	488
3	24	976
4	488	1000
5	488	1488
6	24	1976
7	68	2000

■ **Example** Host A is on LAN 1 and host B is on LAN 2. The two LANs are interconnected by a router. The MTU of the two LANs are 1000B and 500B, respectively. Suppose an application on host A executes 5000 writes to an application on host B. Each write results in one MTU sized IPv4 packet on LAN 1. Each packet consists of an IP header (20B) and the data. The IP headers do not carry any options. How many packets from host A to host B traverse LAN 2? What are the sizes of the packets?

■ **Answer:** Each IP packet has 20B of header and 980B of payload.

The first fragment will be 500B long, with 20B of header and 480B of payload. The second fragment will have the same header/payload sizes. This leaves 20B of payload left, which will go into the final fragment, making the fragment size 40B. In all, there will be 15,000 packets on LAN 2.

■ **Example** Suppose that a TCP message that contains 1964 bytes of data and 20 bytes of TCP header is passed to IP for delivery across three networks of the Internet (i.e. from the source host to a first router to a second router to the destination host). The first network uses 14-byte headers and has an MTU (maximum transmission unit) of 1024 bytes; the second uses 8-bytes headers with an MTU of 512 bytes; the third uses 16-bytes headers with an MTU of 4500 bytes. Each network's MTU gives the total packet size that may be sent, including the network header. Give the sizes and offsets of the sequences of fragments delivered to the network layer at the destination host. Assume all IP headers are 20 bytes.

■ **Answer:**

TCP message = payload + TCP header = 1964 + 20 = 1984 bytes

First network:

MTU = 1024 = link header + IP header + payload = 14 bytes + 20 bytes + TCP data ➔ TCP data = 990 bytes but 990 is not in multiples of 8. TCP data fragment must be 984 which is in multiples of 8. Therefore the first network generates three fragments (size, offset) = (984, 0), (984, 123)

Second network:

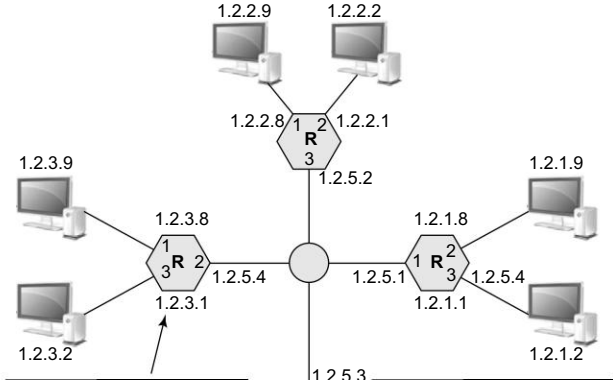
MTU = 512 = link header + IP header + payload = 8 bytes + 20 bytes + TCP data ➔ TCP data = 484 bytes but 990 is not in multiples of 8. TCP data fragment must be 480 that is in multiples of 8. Therefore the second network generates 6 fragments (size, offset) = (480, 0), (480, 60), (24, 120), (480, 123), (480, 183), (24, 243)

Third network:

The third network only forwards 6 fragments (size, offset) = (480, 0), (480, 60), (24, 120), (480, 123), (480, 183), (24, 243) to the destination host because its MTU is 4500 bytes, that is larger than the size of incoming packets.

■ **Example** The diagram below shows a network with 3 routers (shown as hexagons) connected by an Ethernet switch. The routing table for the left-hand router is shown. Complete the routing table for the right-hand router, so that packets will be delivered appropriately (use no more than 5 route table entries).

■ **Answer:**



prefix	next hop	
	output	ip
1.2.2.0/24	2	1.2.5.2
1.2.1.0/24	2	1.2.5.1
1.2.5.0/24	2	–
1.2.3.0/29	3	–
1.2.3.8/29	1	–

prefix	next hop	
	output	ip
1.2.2.0/24		
1.2.3.0/24		
1.2.5.0/24		
1.2.1.0/29		
1.2.1.8/29		

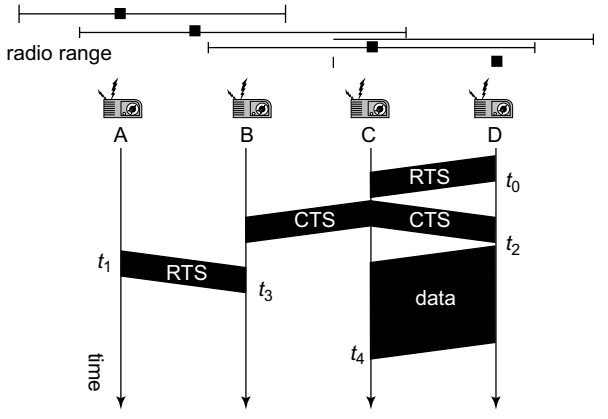
Right hand side routers, routing table looks like:

prefix	next hop	
	output	ip
1.2.2.0/24	1	1.2.5.2
1.2.3.0/24	1	1.2.5.4
1.2.5.0/24	1	–
1.2.1.0/29	3	–
1.2.1.8/29	2	–

a. If a switch was inserted between the host with IP address 1.2.3.2 and its router, how many hosts could be added to that switch, without having to change the routing table entries? What IP addresses would those hosts use?

■ **Answer:** A /29 prefix contains 8 addresses. Since 2 are already in use, so this leaves addresses 1.2.3.0, 1.2.3.3, 1.2.3.4, 1.2.3.5, 1.2.3.6 and 1.2.3.7. So 6 hosts could be added. This is actually not quite right, since there is a special rule for subnets. The first address in any subnet range (the address defined by all zeros) is generally interpreted as the "subnet address" and is not supposed to be used by any host. In addition, the last address in any subnet range (the address defined by all ones) is the "broadcast address" for the subnet and should not be assigned to a host. This rule effectively eliminates hosts 1.2.3.0 and 1.2.3.7, so technically; only 4 hosts can be added.

■ **Example**



Consider the scenario shown in the figure above in which the lines at the top of the figure show the radio range (e.g., so A is heard by B only, and B is heard by A and C but not D). Node D sends an RTS to node C at t0. Node C sends a CTS (which is heard by nodes B and D) in accordance with 802.11 protocol, and node D begins the transmission of its message at t2. In the meantime, node A sends an RTS message to B at time t1.

(a) If node A were to begin transmitting to node B at some point after t3, would A's transmission interfere with the ongoing DATA transmission from D-to-C?

■ **Answer:** No, since A can not reach C.

(b) At t3, can B respond to A's RTS message with a CTS message? Why or why not?

■ **Answer:** No, since B has received a CTS and will have to defer all transmission until after data is sent and ACK is heard.

(c) In the 801.11 protocol, what will node C do at t4, at the end of the receipt of data from node D?

■ **Answer:** C will send an ACK (actually after deferring a bit, but that's not important).

(d) If A were to begin transmitting to node B at some point after t3, would A's transmission be successfully received at B? Justify/discuss your answer in a few sentences.

■ **Answer:** It could interfere with the later ACK transmission from C.

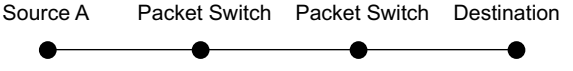
■ **Example** In modern packet-switched networks the source hosts segments long application-layer messages (for example, an image or a music file) into smaller packets and sends the packets into the network. The receiver then reassembles the packet back into the original message. We refer to this process as message segmentation the given figure illustrates the end-to-end transport of a message with and without message segmentation. Consider a message that is $8*10^6$ bits long that is to be sent from source to destination. Suppose each link has a maximum capacity of 2Mbps. Ignore propagation, queuing, and processing delays.

a. Consider sending the message from source to destination *without* message segmentation. How long does it take to move the message from the source host to the first packet switch? Keeping in mind that each switch uses store-and-forward packet switching what is the total time to move the message from source host to destination host?

b. Now suppose that the message is segmented into 4,000 packets, with each packet being 2,000 bits long. How long does it take to move the first packet from source host to the first switch? When the first packet is being sent from the first switch to the second switch, the second packet is being sent from the source to the first switch. At what time will the second packet be fully received at the first switch?

c. How log does it take to move the file from source host to destination host when message segmentation is used? Compare this result with your answer in part (a) and comment.

d. Discuss the drawbacks of message segmentation.



■ **Answer:**

a. Time to send packet from source A to first packet switch=$8x10^6/2x10^6$=4sec. As the packet needs to take 3 hops to reach destination and all links are of same capacity and length, it needs 3x4=12sec for packet to reach destination.

b. Time needed to send small 2000 bit packet from source to first packet switch = $2000/2x10^6$=1ms. While first 2000 bit packet is sent from packet switch 1 to 2, packet switch 1 will be receiving next packet from source. At the end of 2ms, this gets completed.

c. Small packet of 2000bits needs 3ms to reach destination. After that for every 1ms one 2000bit packet will be arriving. As the file size is 4000 packets in total, we will be needing 3999ms + 3ms = 4002ms = 4.002ms

d. Packets need to arranged at destination and control overhead is more.

■ **Example** Suppose within your Web browser you click on a link to obtain a Web page. The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address. Suppose that n DNS servers are visited before your host receives the IP address from DNS; the successive visits incur an RTT of RTT1, …, RTTn. Further suppose that the Web page associated with the link contains exactly one object, consisting of a small amount of HTML text. Let RTT0 denote the RTT between the local host and the server containing the object. a. Assuming zero transmission time of the object, how much time elapses from when the client clicks on the link until the client receives the object? Now suppose the HTML file references eight very small objects on the same server. Neglecting transmission times, how much time elapses with b. Non--persistent HTTP with no parallel TCP connections? c. Non-persistent HTTP with browser configured for 5 parallel connections? d. Persistent HTTP?

■ **Answer:**

a. The total amount of time to get the IP address is RTT1+RTT2+….+RTTn. Once the IP address is known, RTT0 elapses to set up the TCP connection and another elapses to request and receive the small object. The total response time is=2RTT0+RTT1+RTT2+….+RTTn.

b. RTT1+…..+RTTn+2RTT0+8.2RTT0=18RTT0+RTT1+RTT2+…RTTn

c. RTT1+…..+RTTn+2RTT0+2.2RTT0=6RTT0+RTT1+RTT2+…RTTn

d. RTT1+…..+RTTn+2RTT0+RTT0=3RTT0+RTT1+RTT2+…RTTn

■ **Example** Consider sending a large file from a host to another over a TCP connection that has no loss. a. Suppose TCP uses AIMD for its congestion control without slow start. Assuming congestion window increases by 1 MSS every time a batch of ACKs is received and assuming approximately constant round--trip times, how long does it take for congestion window to increase from 5 MSS to 11 MSS (assuming no loss events)? b. What is the average throughput (in terms of MSS and RTT) for this connection up through time = 6 RTT?

■ **Answer:** a. It takes 1 RTT to increase CongWin to 6 MSS; 2 RTTs to increase to 7 MSS; 3 RTTs to increase to 8 MSS; 4 RTTs to increase to 9 MSS; 5 RTTs to increase to 10 MSS; and 6 RTTs to increase to 11 MSS.

b. In the first RTT 5 MSS was sent; in the second RTT 6 MSS was sent; in the third RTT 7 MSS was sent; in the forth RTT 8 MSS was sent; in the fifth RTT, 9 MSS was sent; and in the sixth RTT, 10 MSS was sent. Thus, up to time 6 RTT, 5+6+7+8+9+10 = 45 MSS were sent (and acknowledged). Thus, we can say that the average throughput up to time 6 RTT was (45 MSS)/(6 RTT) = 7.5 MSS/RTT.

■ **Example** Suppose Client A initiates an HTTP session with web server S. At about the same time, Client B also initiates an HTTP session with web server S. Provide possible source and destination port numbers for:

a. The segments sent from A to S.

b. The segments sent from B to S.

c. The segments sent from S to A.

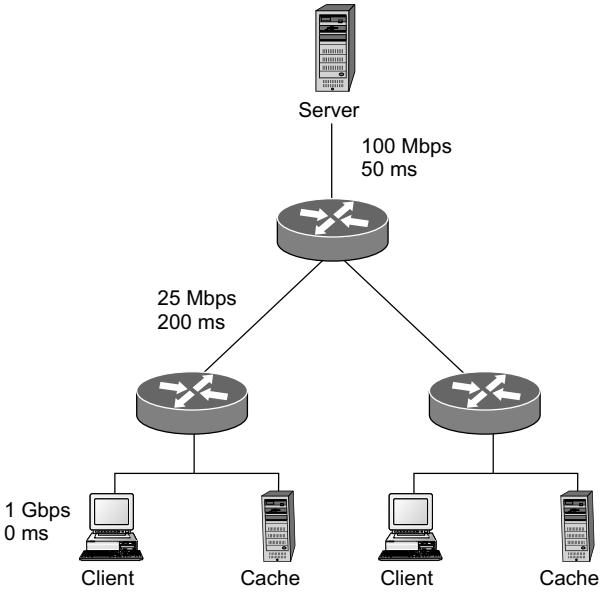
d. The segments sent from S to B.

e. If A and B are different hosts, is it possible that the source port number in the segments from A to S is the same as that from B to S?

■ **Answer:**

source port numbers	destination port numbers
(a) A → S 467	23
(b) B → S 513	23
(c) S → A 23	467
(d) S → B 23	513

■ **Example** Consider the scenario in the given shown in which a server is connected to a router by a 100Mbps link with a 50ms propagation delay. Initially this router is also connected to two routers, each over a 25Mbps link with a 200ms propagation delay. A 1Gbps link connects a host and a cache (if present) to each of these routers and we assume that this link has 0 propagation delay. All packets in the network are 20,000 bits long.



a. What is the end-to-end delay from when a packet is transmitted by the server to when it is received by the client? In this case, we assume there are no caches, there's no queuing delay at the routers, and the packet processing delays at routers and nodes are all 0.

■ **Answer:** If all packets are 20,000 bits long it takes 200 usec to send the packet over the 100Mbps link, 800 usec to send over the 25Mbps link, and 20 usec to send over the 1Gbps link. Sum of the three-link transmission is 1020 usec. Thus, the total end-to-end delay is 251.02 msec.

b. Here we assume that client hosts send requests for files directly to the server (caches are not used or off in this case). What is the maximum rate at which the server can deliver data to a single client if we assume no other clients are making requests?

■ **Answer:** Server can send at the max of the bottleneck link: 25Mbps.

c. Again we assume only one active client but in this case the caches are on and behave like HTTP caches. A client's HTTP GET is always first directed to its local cache. 60% of the requests can be satisfied by the local cache. What is the average rate at which the client can receive data in this case?

■ **Answer:** We assume that requests are serially satisfied. 40% of the requests can be delivered at 25Mbps and 60% at 1Gbps. So the average rate is 610Mbps.

■ **Example** Suppose two nodes, A and B, are attached to opposite ends of an 1200m cable, and that they each have one frame of 1,500 bits (including all headers and preambles) to send to each other. Both nodes attempt to transmit at time t=0. Suppose there are four repeaters between A and B, each inserting a 40-bit delay. Assume the transmission rate is 100 Mbp, and CSMA/CD with backoff intervals of multiples of 512 bits times is used. After the collision, A draws K=0 and B draws K=1 in the exponential backoff protocol. Ignore the jam signal in this case.

a. What is the one--way propagation delay (including repeater delays) between 8 A and B in seconds? Assume the signal propagation speed is 2*10 m/sec.

Answer: $1200m/2x10^8 m/sec + 4* 40bits/100x10^6 bps$ =7.6μsec

b. At what time (in seconds) is A's packet completely delivered at B?

■ **Answer:** First note, the transmission time of a single frame is given by 1500/(100Mbps)=15 micro sec, longer than the propagation delay of a bit. • At time t = 0, both A and B transmit.

At time t = 7.6μ sec, both A and B detect a collision, and then abort.

At time t =15.2μ sec last bit of B 's aborted transmission arrives at A.

At time t = 22.8μ sec first bit of A 's retransmission frame arrives at B. 1500bits At time t = 22.8μ $+1500bits/100x10^6 bps$= 37.8μ sec A 's packet is completely delivered at B.

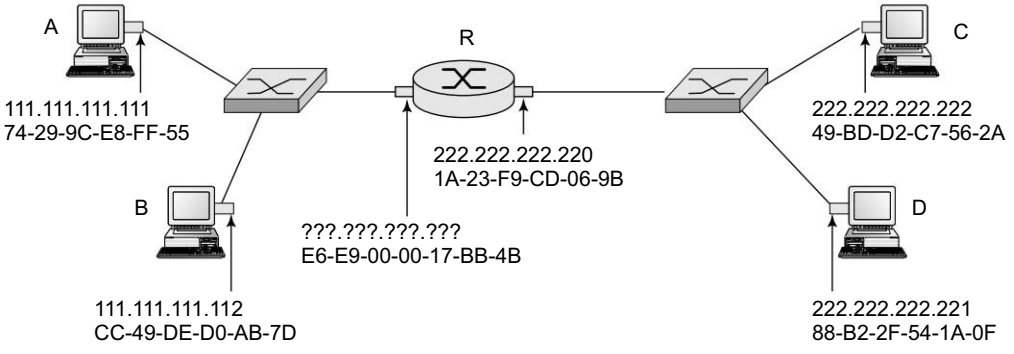
c. Now suppose that only A has a packet to send and that the repeaters are replaced with switches. Suppose that each switch has a 20-bit processing delay in addition to a store-and-forward delay. At what time, in seconds, is A's packet delivered at B?

■ **Answer:** The line is divided into 5 segments by the switches, so the propagation delay between switches or between a switch and a host is given by $1200m/2x10^8 m/sec$ = 1. 2microsec. The delay from Host A to the first switch is given by 15micosec (transmission delay), longer than propagation delay. Thus, the first switch will wait 16.4=15+1.2+0.2 (note, 0.2 is processing delay) till it is ready to send the frame to the second switch. Note that the store-and-forward delay at a switch is 15 microsec. Similarly, each of the other 3 switches will wait for 16.4 microsec before ready for transmitting the frame. The total delay is: 16.4*4 + 15+0.8=81.4 micro sec.

■ **Example** Where and why does packet loss occur within a router?

■ **Answer:** packet loss occurs in buffer in either the input or output line cards, because the memory in the buffer is finite, and the input rate to the buffer exceed the output rate of the buffer over some period of time.

■ **Example** Consider LAN shown in the given figure with its routers and switches.



Now answer each question briefly

(a) Assign an IP address to the leftmost interface of the router, given that the subnet part of IP addresses are 24 bits.

■ **Answer:** Any address starting with 111.111.111.* is fine (except for * being 111 and 112).

(b) Suppose A wants to send an IP datagram to B and knows B's IP address. Must A also know B's MAC address to send the datagram to B? If so, how does A get this info? If not, explain why not.

■ **Answer:** Yes, since B is on the same subnet, it will need to know B's MAC address. This will be done through the ARP protocol.

(c) Suppose A wants to send an IP datagram to C and knows C's IP address. Must A also know C's MAC address to send the datagram to C? If so, how does A get this info? If not, explain why not.

■ **Answer:** No, A will forward the frame to the router, and the router will then de-capsulate the datagram and then re-encapsulate the datagram in a frame to be sent over the right subnet. R will need to run ARP in this case to get C's MAC address, but A will not).

(d) Suppose that R has a datagram (that was originally sent by A) to send to C. What are the MAC addresses on the frame that is sent from R to C? What are the IP addresses in the IP datagram encapsulated within this frame?

■ **Answer:** Source IP: 111.111.111.111, dest IP: 222.222.222.222. source MAC: 1A-23-F9-CD-06-9B (right interface of R), dest MAC: 49-BD-D2-C7-56-2A (node C).

(e) Suppose the switches above are learning switches and suppose that the switch has just been turned on. Suppose now A send an Ethernet frame to B.

a. On how many outgoing switch interfaces will this first frame be carried?
   Answer: Two interface (to B and to router)
b. Now suppose that B replies to A and A sends a second frame to B. On how many outgoing switch interfaces will this second frame be carried?
   Answer:
   one interface (to B) since switch has learned where B is.
   Suppose now (for the two questions below) that the router is removed from the scenario above

(f) Can the nodes keep their IP addresses the same as shown in the picture above? Explain in one or two sentences.

■ **Answer:** If the subnet is /24 or actually anything except no network bits, then the answer is NO, since nodes on the same subnet need to have the same network part of their address.

(g) Suppose that the network manager wants to assign A and C to the same VLAN and B and D to a different VLAN. When a frame is forwarded between switches, how does the receiving switch know which VLAN the frame is destined to?

■ **Answer:** It's contained in the VLAN tag in the Ethernet frame header.

■ **Example:** A TCP machine is sending windows of 65,535 bytes over a 1-Gbps channel that has a 10-msec one way delay. What is the maximum throughput achievable? What is the line efficiency?

■ **Answer:** Transmission time for the sending window size, tw = window size / channel speed = (65,535 × 8 bits) / 10^9 bps = 0.52428 msec.

Propagation delay, tp = RTT/2 = 10 msec.

An ACK packet is sent only when the last bit of the data in the sending window is received. For simplicity, we assume that ACK packets are extremely small (so that we can ignore the transmission time). Then the time for sending an ACK message is tp. Thus total time for sending data in the sending window with TCP,

T = tw + tp + tp = = tw + 2 × tp = 20.52428 msec.

The throughput for the TCP machine,

U = the amount of data sent / total time to send data (T) = (65,535 × 8 bits) / 0.02052428 sec ≅ 2554438 bps ≅ 2.55 Mbps.

The line efficiency E = 2554438 /10^9 = 0.002554438 ≅ 2.55 %.

■ **Example** Suppose there are 5 users whose traffic is being multiplexed over a single link with a capacity of 1 Mbps.

(a) Suppose each user generates 100 kbps when busy, but is only busy (i.e., has data to send) 10% of the time. Would circuit-switching or packet-switching be preferable in this scenario? Why?

■ **Answer:** Here, circuit switching is preferable since each of the users will each get a dedicated allocation of 100 kbps.

(b) Now suppose that the link capacity is still 1 Mbps, but the amount of traffic each user has to send when busy is increased to 1 Mbps, and that each of the 5 users still only has data to send 10% of the time. Would circuit-switching or packet-switching be preferable in this scenario? Why?

■ **Answer:** Here, packet switching is preferable. We cannot allocate 1Mbps per user in circuit switching mode. Packet switching will work well since the aggregate average traffic rate is 0.5 Mbps and the link is a 1 Mbps link

■ **Example** Suppose a TCP SYN message is sent from a client with IP address 128.119.40.186 and client port number 5345 to a server with IP address 41.123.7.94 and server port number 80 (HTTP)

(a) Once the TCP connection has been established, what will be the client-side IP address, client-side port number, server-side IP address and server-side port number of the TCP segment carrying the HTTP GET message.

■ **Answer:** Exactly as specified in the problem statement.

2) Will the TCP SYN message and the HTTP GET message be directed to the same socket at the server? Explain in one or two sentences.

■ **Answer:** No. The GET will be directed to the new socket that was created when the TCP SYN messages was accepted (i.e., the socket returned from the wait on the. accept() on the welcoming socket). Note that the TCP SYN and the GET will both be addressed to port 80 on the server, however.

■ **Example** In network-assisted congestion control, how is congestion in the network signaled to the sender?

■ **Answer:** The routers may set bits in passing packets, and may send messages to the source or destination to indicated congestion.

■ **Example** Name one protocol that uses a network-assisted approach.

■ **Answer:** ATM ABR uses a network-assisted approach.

■ **Example** In end-end congestion control, how is congestion in the network signaled to the sender?

■ **Answer:** Congestion is inferred at the end hosts (sender or receiver, either as a the result of packet loss or increased delay).

■ **Example** Name one protocol that uses an end-end approach.

■ **Answer:** TCP uses an end-end approach.

■ **Example** Briefly describe (in a few sentences) how caching is used in the DNS.

■ **Answer:** A host will maintain a cache of recent DNS-translated address/name pairs. If a name is found in the cache, the DNS system will not be consulted to perform the mapping.

■ **Example** Are values returned from a DNS cache always guaranteed to be up to date? Explain.

■ **Answer:** No. Cached values have a time-to-live value and will remain in the cache until they time out. If the name/address pair is changed in the DNS, the cached value will not change, and the current mapping will only become known after the old mapping times out of the cache, and a new mapping is retrieved from the DNS.

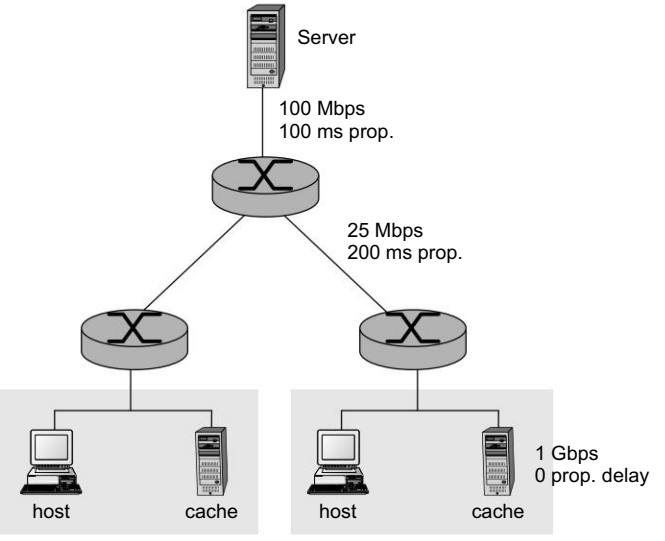
■ **Example** Does the Internet checksum always detect errors in the transmitted segment? Explain your answer in a sentence or two.

■ **Answer:** No. For example if two 16-bit word values are swapped, this would not be detected since the sum is unchanged.

■ **Example** What is meant by demultiplexing a protocol data unit up to an upper-level protocol?

■ **Answer:** this refers to passing the decapsulated data unit up to the appropriate higher level protocol. This is done by looking at the upper-layer protocol field.

■ **Example**



Consider the scenario in the figure above in which a server is connected to a router by a 100 Mbps link, with a 100 ms propagation delay. That router in turn is connected to two routers, each over a 25 Mbps link with a 200 ms propagation delay. A Gbps link connects a host and a cache (when present) to each of these routers; this link, being a local area network, has a propagation delay that is essentially zero. All packets in the network are 10,000 bits long.

(a) What is the end-to-end delay from when a packet is transmitted by the server to when it is received at a host? Assume that there are no caches, that there is no queueing delay at a link, and that the node (router) packet-processing delays are also zero.

■ **Answer:** If all packets at 10,000 bits long, it takes 100 usec to send the packet over a 100Mbps link, 400 usec to send over a 25Mbps link, and 10 usec to send over a gigabit link. The sum of the three link transmission times is thus 510 usec. The sum of the propagation delays is 200+100=300 msec. Thus the total end-end delay is 300.510 msec.

(b) First assume that client hosts send requests for files directly to the server (i.e., the caches are off). What is the maximum rate at which the server can deliver data to a single client, assuming no other clients are making requests.

■ **Answer:** 25 Mbps, the bottleneck link speed.

(c) Again assume that only one client is active, but now suppose the caches are HTTP caches and are turned on. A client HTTP GET is always first directed to its local cache. 50% of the requests can be satisfied by the local cache. What is the maximum rate at which this client can receive data in this scenario?

■ **Answer:** We assume that requests are serially satisfied. 50% of the requests can be delivered at 25Mbps and 50% of the requests can be delivered at 1 Gbps. So the average rate is 512.5 Mbps.

(d) Now suppose that the clients in both LANs are active and the HTTP caches are on, as in c) above. 50% of the requests can be satisfied by the local cache. What is the maximum rate at which each client can receive data, in this scenario?

■ **Answer:** The 25 Mbps remains the bottleneck link, which is not shared between clients. So the answer is the same as (c) above. Note that we assume that the 100Mbps is shared at a fine grain, so that each client can get up to 50Mbps over that link.

(e) Now suppose the 100 Mbps link is replaced by a 25 Mbps link. Repeat question d) above in this new scenario.

■ **Answer:** The two clients must now share the 25Mbps bottleneck link, each getting 12.5 Mbps. 50% of the requests from a client are delivered at 12.5 Mbps and 50% of the requests can be delivers a 1 Gbps. So the average rate is 506.25 Mbps.

■ **Example** (a) What is the purpose of the receiver-advertised window in TCP?

■ **Answer:** This allows the receiver to tell the sender how much unacknowledged data can be in flight.

(b) Consider two TCP sessions that must share a link's bandwidth. One of the TCP connections has been running for quite some time and has built up a large TCP sending window. The second connection then starts up with an initially small window. Long term, what will be the relative throughput achieved by these two TCP sessions? Explain.

■ **Answer:** We saw that TCP will cause two senders to eventually fairly share the links bandwidth, so each will eventually have the same size window (assuming their RTT is the same).

(d) Consider a TCP session and a UDP session that must share a link's bandwidth. Of course, both sessions would ideally like to send as fast as they can. Long term, what will be the relative throughput achieved by these two sessions? Explain.

■ **Answer:** Since UDP can send as fast as it wants, it can use all of the bandwidth (e.g., in the limit that it sends infinitely fast, as soon as buffer space becomes free in a router, that buffer will be filled by a UDP segment. TCP segments will always be lost, causing TCP to keeps its window at 1 segment, which when sent is always lost.

(e) Suppose we want to modify TCP so that it counts the number of segments that are lost in transit. What are the difficulties of doing this in TCP?

■ **Answer:** The key complications here are (i) that ACKs are cumulative (and so the sender may never see some packets individually acked); (ii) premature timeouts result in segments being resent that were never lost in the first place, and (with acks being cumulative) one can't tell how many of the retransmitted segments are lost.

- Because ACKs are cumulative, if the sender gets no ACK for x but gets an ACK for x+1, it doesn't know if x was received and the ack for x was lost, or if x was lost but a later retransmission of x was received.
- A lack of an ACK for x could just mean that the ACK for x is delayed. Suppose x is retransmitted. The sender may never see an ACK for the retransmitted x because of cumulative ACKs for segments with a higher sequence number than x.
- Because of the ACK-every-other segment behavior of TCP receivers, an ACK may never be generated for x is x+1 is received soon after x.

■ **Example** Design a reliable byte-stream protocol that uses a sliding window (like TCP). This protocol will run over a 100-Mbps network. The RTT of the network is 80 ms, and the maximum segment lifetime is 300 seconds. How many bits would you include in the AdvertisedWindow and SequenceNum fields of your protocol header?

■ **Answer:** Bandwidth = 100 Mbps, RTT = 80 ms, MSL = 300 seconds

Advertised Window Size = BW * RTT

= 100Mbps * 80 ms= 2^{20} Bytes

Therefore, Advertised Window Size = 20 bits

Now considering the maximum segment lifetime,

Size = BW * MSL

= 100 Mbps * 300sec = 3750 MB < 4GB = 2^{32} B

Therefore, we need 32 bits in the Sequence Number Field.

■ **Example** Suppose host A sends two TCP segment back to back to host B over TCP connection. The first segment has sequence number 90; the second has sequence number 110.

(i) How much data is in the first segment?

(ii) Suppose that the first segment is lost but the second segment arrives at B. In the Acknowledgement that host B sends to host A, what will be the acknowledgement number?

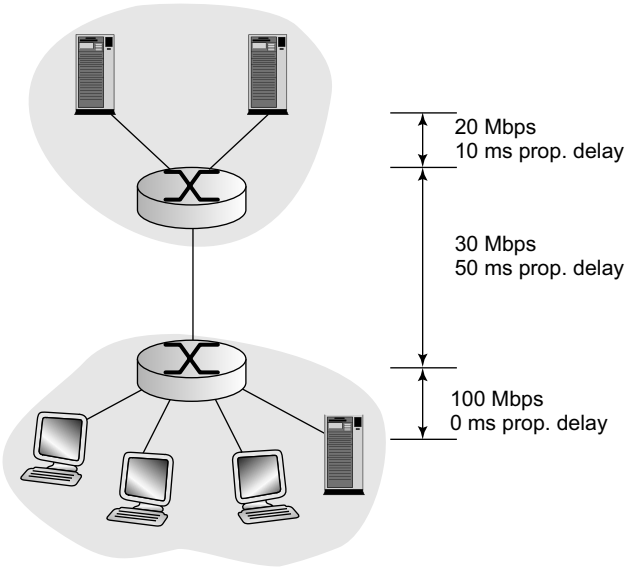
■ **Answer:**

i. Data Size in the First Segment: 110 –90 = 20 bytes

ii. Acknowledgement Number = 90

■ **Example** Consider the scenario in the given figure, in which (from the bottom up) three hosts and a local logging server (that stores information that is sent to it) are connected to a router and to each other by a 100 Mbps link,

with a near-zero ms propagation delay. That router in turn is connected to another router over a 30 Mbps link with a 50 ms propagation delay, and that latter router is connected to two remote logging servers, each over a 20 Mbps link with a10 ms propagation delay.



(a) Suppose a host sends a logging message directly to one of the remote logging servers. The logging message is 10K bits long. What is the end-to-end delay from when the logging message is first transmitted by the host to when it is received at the remote server? Assume that the request goes directly to the server, that there are no queueing delays, and that node (router) packet-processing delays are also zero.

■ **Answer:** Given the 10K bit packet, it takes.0005 secs to send this packet over a 20 Mbps link. 0.000333 secs to send over a 30 Mbps link, and.0001 secs over the 100 Mbps link. The total transmission time end-to-end is this .0009333 secs. The total propagation delay is 60 ms. Therefore the total end-end delay is .0609333 secs.

(b) Assume that each of the three hosts generate logging messages at the same rate; each host is equally like to send a logging message to either of the two remote servers. No traffic is directed to the local logging server. What is the maximum rate at which the clients can send logging messages to the remote servers?

■ **Answer:** The link between routers is the bottleneck link, allowing 30 Mbps to be delivered to the two servers combined, or 15 Mbps to be delivered to each server. Since each message is 10K bits, this is 1.5K logging messages per second.

(c) Now assume that the local logging server is ON and only one host is active (generating) logging messages and that host is only sending messages to one of the remote logging servers. Suppose that 50% of the logging messages are directed locally and the other 50% directed to this remote server. What is the maximum rate at which this host can generate and send logging messages (both local and remote combined, given there is a 50/50 ratio of local/remote transmissions) in this scenario?

■ **Answer:** The maximum rate at which the host can generate remote logging messages is 20 Mbps or 2K logging messages per second. Local messages can be generated that the same rate, so the overall rate is 40 Mbps or 4K logging messages per second.

■ **Example** Consider a router with N input lines, each with input link rate R and an internal switching fabric that is 2N times faster than R. Where packet queue will be formed in this router? Explain your answer.

■ **Answer:** Queueing will only occur at the output ports. Since the switch is more than N times fast than the input rate, all arriving packets in a slot can be move from input port to output port in that slot.

■ **Example** Suppose BGP router A sends a BGP path vector to BGP peer router B. BGP peer B is connected to BGP peer C. Must B advertise that path to C?

■ **Answer:** No. It is up to B's internal policy about what routes to advertise to others. An ISP will generally only carry traffic to/from its customers, and not carry transit traffic (i.e., traffic that is both sources and destined in non-customer networks).

■ **Example** In IP forwarding, it's possible for two packets to take different paths from a common router to the same destination, based on their source IP address.

■ **Answer:** False.

■ **Example** In MPLS (multi-protocol label switching) forwarding, it's possible for two packets to take different paths from a common router to the same destination, based on their source IP address.

■ **Answer:** True.

■ **Example** Briefly describe how Ethernet's exponential backoff works. What is onereason why Ethernet's exponential backoff might be better than randomizing retransmission attempts over a fixed-length time interval?

■ **Answer:** Ethernet maintains an interval of time T over which is will randomize when it will attempt a retransmission. After each collision for the same packet, it doubles the length of T up to some fixed max. This is better than just a single, fixed value of T since when there are a lot of collisions the interval over which randomization is done will be large, allowing just one node to successfully being transmitting. When there are only a small number of colliding nodes, the retransmission will be randomized initially over a small T, allowing a node to transmit more quickly.

■ **Example** Consider four Internet hosts, each with a TCP session. These four TCP sessions share a common

bottleneck link – all packet loss on the end-to-end paths for these four sessions occurs at just this one link. The bottleneck link has a transmission rate of *R*. The round trip times, *RTT,* for all fours hosts to their destinations are approximately the same. No other sessions are currently using this link. The four sessions have been running for a long time.

(a) What is the approximate throughput of each of these four TCP sessions? Explain.

■ **Answer:** R/4 since TCP shares bandwidth fairly.

(b) What is the approximate size of the TCP window at each of these hosts? Explain briefly how you arrived at this answer.

■ **Answer:** You know that

throughput = W/RTT or W = throughput * RTT = R*RTT/4.

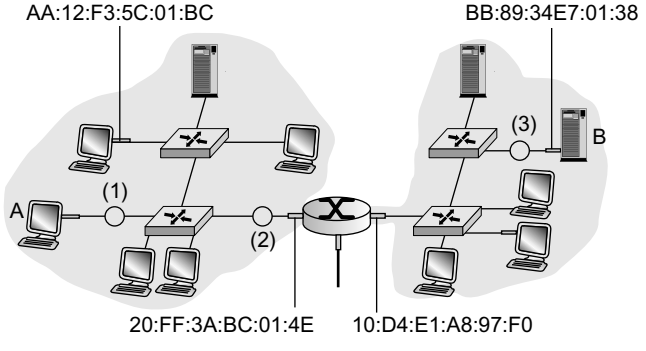
(c) Suppose that one of the sessions terminates. What is the new throughput achieved by each of the three remaining sessions? Briefly describe how this new throughput is reached (i.e., what do the TCPs in the remaining three hosts do that results in this new throughput being achieved).

■ **Answer:** R/3 since TCP shares bandwidth fairly

(d) Now suppose that one of the three hosts starts a second session that also crosses this bottleneck link. What is the throughput achieved (in aggregate) by the one host with two sessions, and by each of the two hosts with one session each?

■ **Answer:** Each session will again get R/4, so the one host with two sessions will get R/2 in aggregate and the other two hosts will each get R/4.

■ **Example:** Consider the following network with routers, hosts.



(a) Assign IP address ranges to the subnets containing hosts A and B, and assign IP addresses in these ranges to hosts A and B. (You don't have to assign IP addresses to any hosts except A and B, but you do need to specify the address range being used by each subnet). Your subnet addressing should use the smallest amount of address space possible.

■ **Answer:** Because there are less than eight but more than 4 nodes in each subnet, we'll need three address bits for each subnet. So let's assign the left subnet XX.YY.ZZ.xxxx0***/29, where the XX.YY.ZZ are 8 bit numbers. Each x is a bit and the three *'s correspond to the three address bits for this network. For the right subnet, well use XX.YY.ZZ.xxxx1***/29. A will have an IP address of XX.YY.ZZ.xxxx0000 and B will have an IP address of XX.YY.ZZ.xxxx1000.

(b) What IP address range can the router advertise to the outside for all of the hosts reachable in these two subnets? Again, you should choose your answer in a) above so that the minimum-size address space is advertised here.

■ **Answer:** XX.YY.ZZ.xxxx/28

(c) Does the router interface with link-layer address 20:FF:3A:BC:01:4E have an IP address? If so, what is the role of the IP address of the router's IP interface in forwarding datagrams through the router.

■ **Answer:** Yes. That's the address that a host in the left network will use to determine the MAC address to send frames to, containing datagrams that need to be routed through the router. The router address however, won't appear in the IP datagram.

(d) Consider an IP datagram being sent from A to B using Ethernet as the link layer protocol in all links in the figure above. What are the (i) Ethernet source and destination addresses and (ii) IP source and destination addresses of the IP datagram encapsulated within the Ethernet frame at points (1), (2), and (3) in the above example for a datagram going from A to B.

■ **Answer:**

(1): ETH source: aa:12;F3:5C:01:BC, ETH dest: 20:FF:3A:BC:01:4E

IP source: XX.YY.ZZ.xxxx0000; IP dest: XX.YY.ZZ.xxxx1000 see (a) above

(2) same as (1)

(3) ETH source: 10:D4:E1:A*:97:FO, ETH dest: BB:89:34:E7:01:3B

IP source: XX.YY.ZZ.xxxx0000; IP dest: XX.YY.ZZ.xxxx1000 same as (1) above

(e) Suppose all switches in the above example are learning switches. Consider the datagram being sent from A to B; neither A nor B have sent any frames or datagrams in the network before. How many of the 11 hosts in the network receive the frame containing the datagram sent by A? Explain your answer briefly.

■ **Answer:** All 11, since no switch knows where B is located (since B hasn't sent anything), all switches will broadcast the frame containing the IP datagram from A. Note that different frames are broadcast on the left and right subnets (e.g., the frames have different source and destination MAC addresses, see above), but both contain the datagram from A.

Suppose the server in the upper part of the left network sends a datagram to A shortly after the A-to-B datagram is sent. How many of the 11 hosts in the network receive the frame containing the datagram sent by this server? Explain your answer briefly.

■ **Answer:** Only A will receive that, since all of the switches know the outgoing port leading to A, as a result of learning where A is, as a result of the initial A-to-B transmission.

(f) Suppose A sends out an ARP request, and this ARP request is in the very first frame sent in the network above (i.e., even before the original A-to-B datagram). How many of the 11 hosts in the network receive the frame containing this ARP request? Explain your answer briefly.

■ **Answer:** ARP broadcasts are restricted to a subnet and generally do not pass through the router, so all 5 other hosts in the left network will receive the ARP broadcast (as will the leftmost interface on the router).

■ **Example** Assume we have a 1 kilometer fiber optic link between A and B with bandwidth 1 megabit per second. Information on this link travels at the speed of light, which is $3*10^8$ meters per second. A sends a 1 kilobyte packet to B.

a. Give an expression for the propagation delay.

■ **Answer:** 1000 meters/($3*10^8$ meters/second) = 3.33μs

b. Give an expression for the transmission delay.

■ **Answer:** Transmission delay=propagation delay + frame time=3.33 μs + $1000x8/10^6$=8003.33 μs

■ **Example** Answer the following questions about TCP with short answers.

a. What is the purpose of the receive window? That is, explain what problem it is aimed toward fixing?

■ **Answer:** Flow control, that is the receiver buffer should not overflow.

b. Who sets the receiver window, the sender or the receiver?

■ **Answer:** Receiver

c. How does the other entity respond to this setting of the receive window?

■ **Answer:** The sender makes sure that the number of sent but unacknowledged bytes is at most the receive window.

■ **Example** Consider the TCP Reno protocol.

a. What are the two possible loss events for TCP Reno?

Answer: Time-out and triplicate ACK

b. How does TCP Reno change the congestion window in response to each of these types of loss events? Assume that the congestion window is of size X segments just prior to the loss event at time t. Assume X is quite large. Give the size of the congestion window at times t, t + RTT, t + 2RTT, for each possible loss event. Here RTT is the round trip time.

■ **Answer:** After a time-out: 1, 2, 4
After a triplicate ACK: X/2, X/2+1, X/2 +2

■ **Example** Assume that a TCP process A first measures the actual round trip time to another TCP process to be 30 ms, and A thus sets its estimated round trip time to be 30 ms. The next actual round trip time that A sees is 90 ms. In response A increases its estimated round trip to 70 ms. The next actual round trip time that A sees is 50 ms. What is the next estimated round trip computed by A?

■ **Answer:**
70= alpha * 30 + (1-alpha)90
Thus alpha=1/3
Then next estimated round trip is (1/3)70 + (2/3)50=56.7

■ **Example** Consider that a browser on host A wants to retrieve a html document D, and an embedded image I, on a host B. Assume that A does not initially know the IP address of B, but A's local name server S does know B's IP address. Assume that the browser on A uses HTTP/1.0. Show the chronological sequence of transport layer segments (TCP, or UDP) sent and the application layer data type (DNS or HTTP) included by filling in the following table. Also state when any of the SYN, FIN and/or ACK bits in the TCP header are set.

■ **Answer:**

Source	Destination	Transport Layer Protocol	Application Layer Protocol
A	S	UDP	DNS
S	A	UDP	DNS
A	B	TCP SYN=1	
B	A	TCP SYN=ACK=1	
A	B	TCP ACK=1	HTTP
B	A	TCP	HTTP
A	B	TCP FIN=1	
B	A	TCP FIN=ACK=1	
A	B	TCP ACK=1	
A	B	TCP SYN=1	
B	A	TCP SYN=ACK=1	
A	B	TCP ACK=1	HTTP
B	A	TCP	HTTP
A	B	TCP FIN=1	
B	A	TCP FIN=ACK=1	
A	B	TCP ACK=1	

■ **Example** Consider TCP congestion control. Assume we have a round trip time RTT of 4 seconds. Assume that the segment size is 3 kilobyte. Assume that the bandwidth of the connection is 500 kilobits per second. What is the smallest window size for which there will be no stalling?

■ **Answer:**
WS/R > RTT + S/R. By substitution,
W * 3 * 8/500 > 3*8/500 + 4.
Solving for W gives W = 84 segments

■ **Example** Here we consider TCP Reno connections through a bottleneck router. Assume that capacity of the router measured in TCP segments is R, where R is quite large.
a. Assume that you have only one TCP connection. What will be the approximate average long term throughput for this connection as a function of R?

■ **Answer:** 3R/4

b. Assume that you have 2 TCP connections with equal round trip times. What will be the approximate average long term throughput for each connection?

■ **Answer:** (R/2)(3/4)

c. You have 3 TCP connections A, B, and C with round trip times 1, 2, and 3 respectively. What will be the approximate average long term throughput for each connection?

■ **Answer:** Let X be A's share of the bandwidth. Then B's share is X/2 and C's share is X/3. Hence X + X/2 + X/3 = R. Therefore X=6R/11. Now taking multiplicative decrease into account, we get an answer of (3/4)(6R/11)

■ **Example** Consider sending a file of F Kbytes in the following settings.

Setting 1 consists of two computers A and B, each equipped with a modem that is capable of sending/receiving at 33.3 kbps. For A to send a file to B, it must first establish a dial-up connection with B, which takes 30 seconds. It can then send the file in 128-byte packets, with a 1 byte checksum attached to each packet. The propagation delay of the phone line is negligible. Assume that A and B are directly connected, i.e., there are no intermediate routers.

Setting 2 consists of two computers C and D, connected by an established wireless connection that can transmit at 8 kbps through a satellite, with a 0.25s total propagation delay from C to D. In setting 2, files are transmitted without being split into packets. You may assume there are no errors during each transmission and you may ignore acknowledgments, i.e., consider only bits flowing from the sending computer to the receiving computer.

If F = 16, how long does it take to send the file from A to B?

■ **Answer:** Total number of bytes transmitted = size of file + number of checksum bytes
= 16K + 16K/128 = 16.125K bytes
Total Time = time for connection setup + time for tx
= 30s + (16.125K * 8bits)/33.3k (note that kbps is kilo bits per sec) = 33.97 sec
How long does it take to send the file from C to D?

■ **Answer:** Total Time = propagation delay + time for transfer
= 0.25 sec + (16K*8bits)/8k sec= 16.634sec

■ **Example** Why is it difficult to implement persistent connections for CGI scripts and dynamic content in general?

■ **Answer:** In persistent http, the client must be able to tell different objects apart from each other. For this to happen, the server must include the size of the content in the header. However, the output size of the CGI scripts and the size of the dynamic content cannot be determined in advance.

■ **Example** Suppose you are downloading a page with m large objects, m > n, and a large number of small objects. Assume the client requests the objects in the order it finds them in the page. What problem does the above persistent connections scheme pose in this situation?

■ **Answer:** Since the large objects will take a lot of time to download, the user will not see any content for a long time and get an impression that the server is not responding. So he will refresh the page and the same thing will start over again.

Describe a simple client side (i.e., in the browser) solution to address this problem and improve the overall response time for the Web page.

The client can reserve one connection for large objects and use the rest for small objects. The client can use the HEAD method of http and request the meta-data of the objects (here images) including the size of the object. Then it can use the GET method to retrieve the large-sized images on one connection and the smaller objects on the other connections.

■ **Example** Suppose you are designing a sliding window protocol for a 1Mbps point-to-point link to the Moon, which has a one way latency of 1.25s. Assuming that each frame carries 1KB of data, what is the minimum number of bits you need for the sequence number?

■ **Answer:** RTT=Round Trip Time = 1.25 * 2 = 2.5 s
Total packet out = 2.5 * 1M bps / 1KB + 1 = 305
Therefore, we need $\log_2(305)+1= 9$ bits

■ **Example** Assume that two hosts are trying to communicate over a 100Mbps Ethernet segment. Many other hosts also connected to this segment. The hosts wish to exchange a 40B IP packet (including IP, TCP and application headers and payload). What is the minimum number of padding bytes that the sending host's Ethernet adaptor will add? Assume that electromagnetic signals travel at the speed of 108 m/s over 100Mbps Ethernet cable. Also assume that the maximum allowed segment size is 1000m. Suppose nodes A and B are on the same 10Mbps Ethernet segment. Say the propagation delay between them is 250 bit times.

■ **Answer:** For a collision to be effectively detected, in this case, the sending packet should be able to last for at least $1000 / 10^8 * 2 = 2*10^{-5}$ s
Thus, it should be at least
$100 * 10^6 * 2*10^{-5} = 1000$ bits = 250 byte => Need to pad 250 – 40 = 210 bytes

Suppose at time t = 0, B starts to transmit a frame. Suppose A also transmits at t = x, but before completing its transmission, it receives bits from B (hence, a collision occurs at A). Assuming node A follows the CSMA/CD protocol, what is the maximum value of x?

■ **Answer:** Node A senses empty channel from t=0 to t = 250, so A can transmit at any time during this interval. At time t=250, A senses a busy channel and will refrain from transmitting. So x=249 bit times and is equivalent to 24.9 microseconds.

Suppose at time t = 0, both nodes start to transmit a frame. At what time do they detect a collision? Assuming both nodes transmit a 48-bit jam signal after detecting a collision, at what time (in bit times) do A and and B sense an idle channel? How many seconds is this for 100Mbps Ethernet?

■ **Answer:** Both nodes detect a collision at t=250. At t=250+48, both nodes stop transmitting their jam signals. The last bit of jam signal from B arrives at A at t=298+250=548 bit times. Similarly, the last bit of A's jam signal arrives at B at 548 bit times. This is when both nodes sense an idle channel. At 100Mbps, 548 bit times is equivalent to 5.48 microseconds.

■ **Example** Video applications typically run over UDP rather than TCP because they cannot tolerate retransmission delays. However, this means video applications are not constrained by TCP's congestion control algorithms.

(a) What impact does this have on TCP traffic? Be specific about the consequences.

■ **Answer:** Since UDP does not have congestion control, thus they could not lower the transmission rate when congestion occurred. On the other hand, TCP would lower its transmission rate. In this case, most of the bandwidth would be occupied by UDP, which would significantly worsen the TCP performance.

(b) Assume these video applications uses RTP, which results in RTCP "receiver reports" being sent from the sink back to the source. These reports are sent periodically (e.g. once a second) and include the percentage of packets successfully received in the last reporting period. Describe how the source might use this information to adjust its rate in a TCP compatible way.

■ **Answer:** The percentage of successful received packets gives the source the indication of current network congestion conditions. When the percentage is low, the source should lower its sending rate accordingly, and when the percentage is high, it can steadily increase its sending rate until the percentage starts dropping. This mechanism is very similar to that used in TCP like AIMD.

■ **Example** Calculate the buffering needed to ensure that TCP fully utilises a link if:

(a) TCP were to change its multiplicative decrease to decrease the window 3/4th the size after a loss.

■ **Answer:** To fully utilize the link, we need to ensure that the average transfer amount equals bandwidth-delay product (BDP). Therefore, with ¾ multiplicative decreasing, the buffer size should satisfy 3(BDP+Q)/4>=BDP, thus the minimum buffer size is BDP/3.

(b) TCP were to change its additive increase to 2 packets per RTT

■ **Answer:** If the additive increase is 2 packets per RTT, and multiplicative decrease is still ½, then the buffer size needed would be BDP as in normal case. Essentially, the buffer size needed only relates to the multiplicative decreasing rate regardless the additive increasing rate.

■ **Example** Consider 10 flows with arrival rates of 1, 2,..., 10 Mbps that traverse a link of 50Mbps. Compute the max-min fair share on this link. What is the fair share if the link capacity is 60 Mbps?

■ **Answer:** Since, the total bandwidth is 50Mbps, originally each flow will be allocated 5Mbps. Thus the first five flows will be allocated 1Mbps, 2Mbps, 3Mbps, 4Mbps, and 5Mbps respectively, since they do not exceeds the limit. Then there would be 35Mbps left for 5 flows. Thus 6th and 7th flow can also be guaranteed. After that 22Mbps will be evenly allocated by the other 3 flows. So the final result would be

No.	1	2	3	4	5	6	7	8	9	10
Size(Mbps)	1	2	3	4	5	6	7	7.33	7.33	7.33

If the total bandwidth increases to 60Mbps, in the similar manner, we can calculate that the allocation would be like the following table:

No.	1	2	3	4	5	6	7	8	9	10
Size(Mbps)	1	2	3	4	5	6	7	8	9	10

which makes sense because currently the resource is enough for all the users.

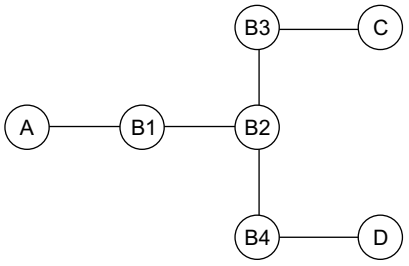
■ **Example** Two flows A and B arrive at a router with a WFQ (weighted fair queue) scheduling policy. The WFQ scheduling is modeled after GPS. Flow A has reserved 1/3 of the bandwidth on the outgoing link. Flow B has reversed 2/3 of the bandwidth on the outgoing link. Flow A's packets are one third the size of flow B's packets. What are the first 6 packets to leave the link?

■ **Answer:** Suppose we have constant and equal flows for both A and B, and they arrived at the same time, but A was served first. Therefore, based on the WFQ policy, the first 6 packets would be A, B, A,B, A, A or A, B, A, A, B, A.

■ **Example**

(1) Consider the arrangement of learning bridges shown in the following figure. Assuming all are initially empty, give the forwarding tables for each

of the bridges B1-B4 after the following transmissions: D sends to C; A sends to D; C sends to A



■ **Answer:** When D sends to C, all bridges see the packet and learn where D is. That is, B4 sends packet to B2. B2 sends the same to both B1 and B3 (other than the line from which packet is arrived). Thus, all the bridges will be learning about D. However, when A sends to D, the packet is routed directly to D and B3 does not learn where A is. That is, packet from A arrives to B1 which already knows where D is, so it sends packet to B2. B2 knows D can be reached via B4, it will send packet to B4 only; not to B3. Similarly, when C sends to A, the packet is routed by B2 towards B1 only, and B4 does not learn where C is. After these three transmissions, what every bridge know about other bridges is summarised below.

The forwarding table for Bridge B1

Destination	Next Hop
A	A-Interface
C	B2-Interface
D	B2-Interface

The forwarding table for Bridge B2

Destination	Next Hop
A	B1-Interface
C	B3-Interface
D	B4-Interface

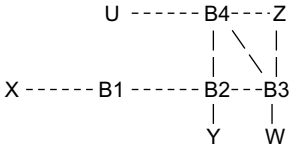
The forwarding table for Bridge B3

Destination	Next Hop
C	C-Interface
D	B2-Interface

The forwarding table for Bridge B4

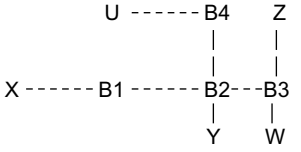
Destination	Next Hop
A	B2-Interface
D	D-Interface

■ **Example** Consider hosts X, Y, Z, W, U and learning bridges B1, B2, B3, B4 with initially empty forwarding tables, connected in the following manner:



a. Assuming B1 has the lowest bridge identifier, and B4 the highest, show the spanning tree for the above network.

Answer:



b. Say B4 is connected to U, B2, B3 and Z via its ports 1, 2, 3 and 4, respectively. Assuming no failures, which of these ports will B4 *never* forward on (i.e. which ports are *blocked*)?

Answer: 3, 4

c. Suppose X sends its first packet to Z. Which bridges learn where X is? Does Y's network interface see this packet?

Answer: All bridges will learn. Yes, Y's interface will see this packet.

d. Suppose Z now sends its first packet to X. Which bridges learn where Z is? Does Y's network interface see this packet?

Answer: B1, B2, B3 will learn. Y will not see this packet.

e. Suppose Y now sends to X. Which bridges learn where Y is? Does Z's network interface see this packet?

Answer: B1, B2 will learn. Z will not see this packet.

■ **Example** Explain about longest prefix match algorithm.

■ **Answer:** Longest prefix match (also called Maximum prefix length match) refers to an algorithm used by routers in Internet Protocol (IP) networking to select an entry from a routing table. Each entry in a routing table may specify a network, one destination address may match more than one routing table entry. The most specific table entry is the one with the highest subnet mask. Thus is called as the longest prefix match because it is the entry where the largest number of leading address bits in the table entry match those of the destination address.

A router has 3 interfaces: 172.21.10.237/28, 172.21.10.66/27, and 172.21.10.193/29. Find out which interface is selected if the destination address of a packet that is arrive is 172.21.10.68?

We apply longest prefix matching algorithm for the given address and the gives three interface addresses as shown below.

1010	**1100.0001**	**0101.0000**	**1010.0100**	**0100**	**172.21.10.68**
1010	**1100.0001**	**0101.0000**	**1010.1110**	**1101**	**172.21.10.237**
1010	**1100.0001**	**0101.0000**	**1010.0100**	**0100**	**172.21.10.68**
1010	**1100.0001**	**0101.0000**	**1010.0100**	**0010**	**172.21.10.66**
1010	**1100.0001**	**0101.0000**	**1010.0100**	**0100**	**172.21.10.68**
1010	**1100.0001**	**0101.0000**	**1010.1100**	**0001**	**172.21.10.193**

We found that the given address is having largest number of prefix bits matching with 172.21.10.66. Thus, this is selected.

■ **Example** The following table is a routing table using CIDR. Address bytes are in hexadecimal.

Net/MaskLength	Next hop
C4.50.0.0/12	A
C4.5E.10.0/20	B
C4.60.0.0/12	C
C4.68.0.0/14	D
80.0.0.0/1	E
40.0.0.0/2	F
00.0.0.0/2	G

State to what next hop the following will be delivered.

(A) C4.5E.20.87 (B) C4.5E.1A.09

(C) C3.41.80.02 (D) C4.6D.31.2A

(E) C4.6B.31.2B

■ **Answer:**

The following table illustrates the interfaces and their prefix bits.

Net/MaskLength	Next hop	Binary Address
C4.50.0.0/12	A	1100.0100.0101
C4.5E.10.0/20	B	1100.0100.0101.1110.0001
C4.60.0.0/12	C	1100.0100.0110
C4.68.0.0/14	D	1100.0100.0110.10
80.0.0.0/1	E	1
40.0.0.0/2	F	01
00.0.0.0/2	G	00

The following are the binary versions of given addresses. We choose the address with longest match in the addresses of routing table.

Binary Address	Next Hop	No. of Bits Match
C4.5E.20.87 = 11000100.01011110.00100000.10000111	=> A;	12
C4.5E.1A.09 = 11000100.01011110.00011010.00001001	=> B;	20
C3.41.80.02 = 11000011.01000001.10000000.00000010	=> E;	1
C4.6D.31.2A = 11000100.01101101.00110001.00101010	=> C;	12
C4.6B.31.2B = 11000100.01101011.00110001.00101011	=> D;	14

■ **Example** An organisation has a class C network 196.10.10 and wants to form subnets for five departments, which host as follows:

(A) 55 hosts
(B) 50 hosts
(C) 45 hosts
(D) 25 hosts
(E) 20 hosts

There are 195 hosts in all. Design a possible arrangement of subnets to make each department in a different subnet. For each subnet, give subnet mask and range of IP addresses.

■ **Answer:** Class C network: 196.10.10 indicates, we can have at most 254 hosts only in total. Our total, 195 is less than 254, we can easily achieve the required subnetting. In principle, we can have many solutions. The following is one of them.

Department	Subnet Mask	Subnet ID	Range of Address
A: 55 Hosts	255.255.255.192	196.10.10.0	196.10.10.0 – 196.10.10.63
B: 50 Hosts	255.255.255.192	196.10.10.64	196.10.10.64 – 196.10.10.127
C: 45 Hosts	255.255.255.192	196.10.10.128	196.10.10.128 – 196.10.10.191
D: 25 Hosts	255.255.255.224	196.10.10.192	196.10.10.192 – 196.10.10.223
E: 20 Hosts	255.255.255.224	196.10.10.224	196.10.10.224 – 196.10.10.255

■ **Example** Which of the following prefixes are contained in the CIDR prefix 201.10.0.0/21?

a. 201.10.4.0/24
b. 201.10.8.0/23
c. 201.10.24.0/22
d. 201.10.6.0/23

■ **Answer:** Binary code of the given IP address:1100 1001 0000 1010 0000 0000 0000 0000. We will check most significant 21 bits of each of given address with the most significant 21 bits given CIDR address, 1100 1001 0000 1010 0000 0 as shown here:

(a) 201.10.4.0/24------**1100 1001 0000 1010 0000 0**100 0000 0000 Matching
(b) 201.10.8.0/23------**1100 1001 0000 1010 0000 1**000 0000 0000
(c) 201.10.24.0/22---- **1100 1001 0000 1010 0001 1**000 0000 0000
(d) 201.10.6.0/23----- **1100 1001 00001010 00000**110 0000 0000 Matching

■ **Example** All router activities related to TTL decrement are always handled in the fast path of the router (i.e., in special purpose silicon chips on the input ports) and the slow path of the router (i.e. the router processor) is never involved.

■ **Answer:** No. When TTL goes to zero, CPU must generate ICMP error.

■ **Example** Does copies of router forwarding table are kept at all input ports?

■ **Answer:** Yes. This enables fast distributed look-ups and prevents route processor from becoming a central bottleneck.

■ **Example** Does IP address lookup in a router takes fewer clock cycles than Ethernet MAC address lookup in a switch (assume that the lookup tables have the same total number of entries).

■ **Answer:** True. For the same length tables, IP address lookup will take lesser time since IP addresses are hierarchical. This is despite the fact that IP lookup needs longest prefix match, masking etc. Also, the fact that Ethernet addresses are long and require an exact match makes Ethernet address lookup cumbersome.

■ **Example** Explain about multihomed hosts? Explain above routing in them.

■ **Answer:** If a host is attached to multiple subnets it is called as multihomed. Typically end-user machines are not multihomed. However servers are multihomed for the following reasons:

- The ability to connect the server to as many different client subnets as possible
- The ability to provide redundant connectivity. The server will still be able to maintain connectivity with the rest of the network even when one of its interfaces fails.

Note that a multihomed server will not typically forward traffic among its interfaces. It must participate in routing in order to be able to forward traffic among its interfaces. The standard IP Forwarding procedure is used with minor changes. We will understand this with a simple example.

At initialisation time, the server will apply the bitwise logical AND operation to each of its interfaces. It will store the subnet prefix and the mask of each of its interfaces. For example.

interface E1: 172.21.11.199/25 (mask: 255.255.255.128 in the dotted-decimal notation)
interface E2: 172.21.11.62/26 (mask: 255.255.255.192 in the dotted-decimal notation)
interface E3: 172.21.13.3/23 (mask: 255.255.254.0 in the dotted-decimal notation)
Default Gateway: 172.21.11.129

interface E1:

10101100.00010101.00001011.10000111	**172.21.11.199**
11111111.11111111.11111111.10000000	**255.255.255.128**
10101100.00010101.00001011.10000000	**172.21.11.128**

The subnet prefix of interface E1 is: 172.21.11.128

interface E2:

10101100.00010101.00001011.00111110	**172.21.11.62**
11111111.11111111.11111111.11000000	**255.255.255.192**
10101100.00010101.00001011.00000000	**172.21.11.0**

The subnet prefix of interface E2 is: 172.21.11.0

interface E3:

10101100.00010101.00001101.00000011	**172.21.13.3**
11111111.11111111.11111110.00000000	**255.255.254.0**
10101100.00010101.00001100.00000000	**172.21.12.0**

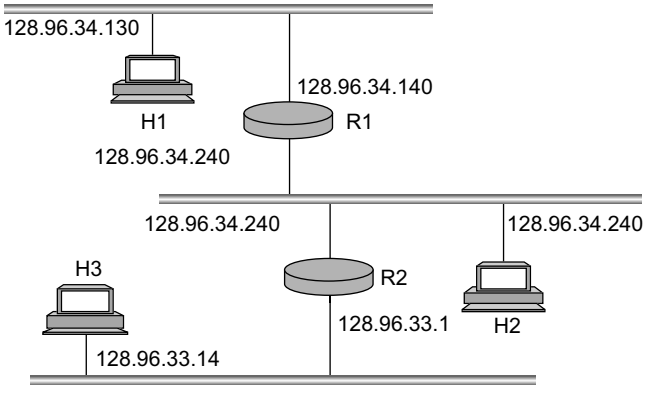
The subnet prefix of interface E3 is: 172.21.12.0

Packet Forwarding Changes:

The server must still determine which outgoing interface to use when it has a packet to forward. When sending the packet, the server will put the IP address of the interface over which the packet will be forwarded in the source address of the IP packet.

- If the destination address of the IP packet to be forwarded is within the subnet prefix of interface E1 (172.21.11.128/25, i.e., address range from 172.21.11.128 to 172.21.11.255), then the packet is sent from interface E1. When sending the packet, the server puts interface E1 IP address in the source address field.
- If the destination address of the IP packet to be forwarded is within the subnet prefix of interface E2 (172.21.11.0/26, i.e., address range from 172.21.11.0 to 172.21.11.63), then the packet is sent from interface E2. When sending the packet, the server puts interface E2 IP address in the source address field.
- If the destination address of the IP packet to be forwarded is within the subnet prefix of interface E3 (172.21.12/23, i.e., address range from 172.21.12.0 to 172.21.13.255), then the packet is sent from interface E3. When sending the packet, the server puts interface E3 IP address in the source address field.
- If the destination address of the IP packet to be forwarded is not within the subnet prefix of one of its interfaces: E1, E2 or E3, the packet is sent to the default gateway. Again the server has only a single default gateway, which has an IP address assigned to it from the interface E1 subnet. When sending the packet, the server puts interface E1 IP address in the source address field.

■ **Example** A company has a network as shown in figure



The hosts H1, H2, H3 has the subnet masks 255.255.255.192, 255.255.255.192, and 255.255.255.0 respectively.

a. What network class does this network belong to? What is network id of this network?

b. How many subnets does this network have? What are subnet Ids?

c. H1 has a routing table. Fill in the entries of routing table as follows:

Subnet Number	Subnet Mask	Next hop (router)
		Interface 0
		R1
		R1

(d) R1 has a routing table. Fill in the entries of routing table as follows:

Subnet Number	Subnet Mask	Next hop (router)
		Interface 0
		Interface 1
		R2

■ **Answer:**

(a) 1. Class B because the first two bits of 128.96.34.240 are "10"

2. Network id is 128.96.

(b)

1. 3 subnets

3. Subnet Ids are 128.96.34.128 (H1), 128.96.34.192 (H2), 128.96.33.0 (H3)

(c)

Subnet Number	Subnet Mask	Next hop (router)
128.96.34.128	255.255.255.192	Interface 0
128.96.34.192	255.255.255.192	R1
128.96.33	255.255.255.	R1

(d)

Subnet Number	Subnet Mask	Next hop (router)
128.96.34.128	255.255.255.192	Interface 0
128.96.34.192	255.255.255.192	Interface 1
128.96.33.0	255.255.255.0	R2

■ **Example** What is the network prefix of an IP address 128.130.4.150/22?

■ **Answer:** The network prefix is 128.130.0.0/22

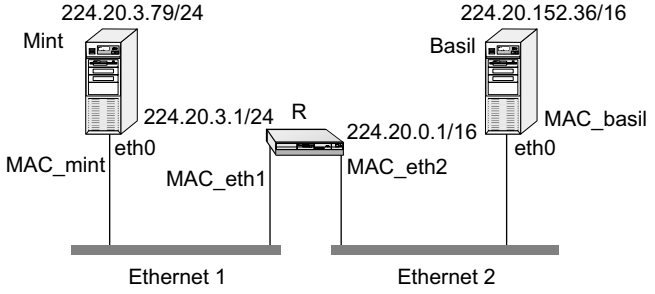
■ **Example** What is an IP address prefix? How many IP addresses will match the address prefix 10.0.0.0/23? How many IP addresses will match the address prefix 0.0.0.0./0?

■ **Answer:** An IP address prefix is used to specify a range of IP addresses who have the same prefix bits of a certain length.

2^9 = 512 IP adddresses match the prefix 10.0.0.0/23.

All IP addresses, i.e. 2^{32} addresses, match the prefix 0.0.0.0/0.

■ **Example** The host Mint and Basil are connected by a router R as shown in the following figure. R has turned on the proxy ARP. The figure also shows the IP addresses and the Ethernet addresses of Mint, Basil, and the router R. Mint and Basil have R configured as their default router. Answer the following questions.



a. Suppose the ARP caches on both PCs and the router are empty. A command "ping-c 1 224.20.152.36" is issued at Mint. Is Mint going to send an ARP request? If yes, what is the source and destination Ethernet address of the ARP request? What is the content of the ARP request? For all questions asking the content of an ARP message, specify the source hardware address, source protocol address, target hardware address, and target protocol address of the ARP message.

■ **Answer:** Yes, Mint will send an ARP request.

The source Ethernet address of the ARP requst is MAC mint, the destination Ethernet address is ff.ff.ff.ff.ff.ff

The source hardware address of the ARP request is MAC mint, the destination hardware address is 00.00.00.00.00.00,

The source target address is 224.20.3.79,

The destination target address is 224.20.3.1.

b. Who will send an ARP reply to Mint? What is the source and destination Ethernet address of the ARP reply? What is the content of the ARP reply?

■ **Answer:** Router R will send an ARP reply to Mint.

The source hardware address of the ARP reply is MAC eth1, the destination hardware address is MAC mint, the source target address is 224.20.3.1, the destination target address is 224.20.3.79.

c. After Mint receives the ARP reply, what is the content of Mint's ARP cache? What is the content of R's ARP cache?

■ **Answer:** Mint's ARP cache is: 224.20.3.1 is at MAC eth1;
R's ARP cache is: 224.20.3.79 is at MAC mint;

d. After Mint receives the ARP reply, it sends the ping message. What type of message does ping send?

■ **Answer:** It sends an ICMP echo request message

e. On Ethernet 1, what are the source and destination Ethernet address of the ping message? What are the source and destination IP address of the ping message?

■ **Answer:** Source Ethernet address: MAC mint;
destination Ethernet address: MAC eth1;
source IP address: 224.20.3.79;
destination IP address: 224.20.152.36.

f. When the router R receives the ping message, how does it decide via which interface to forward the ping message?

■ **Answer:** R looks up its routing table and uses the longest prefix match to forward the ping packet to the destination through interface MAC eth2.

g. Is R going to send an ARP request? If so, what's the content of the ARP request?

■ **Answer:** Yes, R will send an ARP request.

The source hardware address of the ARP request is MAC eth2; the destination hardware address is 00.00.00.00.00.00; the source protocol address is 224.20.0.1; the destination protocol address is 224.20.152.36.

h. What are the source and destination Ethernet address of the ping message on Ethernet 2?

■ **Answer:** The source Ethernet address: MAC eth2; the destination Ethernet address: MAC basil.

i. When Basil receives the ping message, it will send a reply. Is Basil going to send an ARP request? If so, what is the content of the ARP request?

■ **Answer:** Yes, Basil will send an ARP request.
The source hardware address of the ARP request is MAC basil;
the destination hardware address is 00.00.00.00.00.00;
the source protocol address is 224.20.152.36;
the destination protocol address is 224.20.3.79.

j. Who is going to reply to the ARP request? What's the content of the ARP reply? Note proxy-arp is turned on at R.

■ **Answer:** The router R is going to reply to the ARP request.
The source hardware address of the ARP reply is MAC eth2;
the destination hardware address is MAC basil;
the source protocol address is 224.20.3.79;
the destination protocol address is 224.20.152.36.

k. A second command "ping -c 2 224.20.152.36" is issued right after the first one terminates. Is Mint going to send an ARP request this time? Explain why?

■ **Answer:** No, Mint will not send an ARP request this time, because it already has in its ARP cache the Ethernet address of the router R's interface with the IP address 224.20.3.1.

■ Example Ali works in a company whose network topology is shown in the following figure. The company's network has three LANs. Router 2 is the border router of the company that connects directly to an ISP. Ali has to configure the network.

a. LAN0 has at most 200 hosts; LAN1 and LAN2 each has at most 100 hosts. Ali needs to obtain a block of addresses from ARIN so that she can assign IP addresses to hosts. A block of addresses are succinctly represented by an address prefix *address/n*, where n is the number of prefix bits all

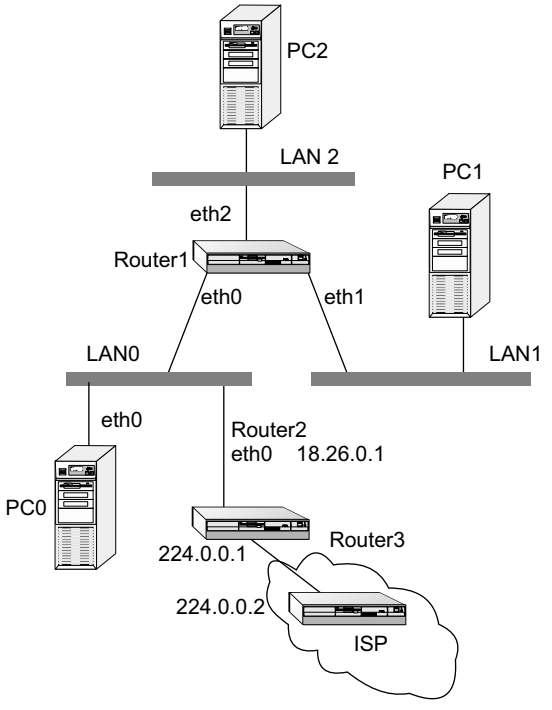
addresses in the block share in common. What is the length of the longest address prefix Ali needs?

■ **Answer:** 23

Since the network has 200 + 100 + 100 = 400 hosts, it requires at least 9 bits to denote host ID, which can represent $2^9 = 512$ hosts. Therefore, the length of the longest address prefix Ali needs is 32 – 9 = 23.

b. Suppose Ali has obtained the longest address prefix. Next, Ali needs to assign addresses to hosts. Specify the maximum length of the network prefix of each LAN.

■ **Answer:** The maximum length of the network prefix of LAN0 is 24; the maximum length of the network prefix of both LAN1 and LAN2 is 25.



c. Suppose Ali has assigned the longest possible network prefix to each LAN. She then assigns IP addresses to hosts and routers on each LAN. As shown in the figure, the IP address she assigns to the eth0 interface of Router 2 is 18.26.0.1. What is the network prefix Ali assigns to LAN0?

■ **Answer:** 18.26.0.1/24.

d. What is the address prefix Ali obtains from ARIN?

■ **Answer:** 18.26.0.0/23.

e. Assign network prefixes to LAN1 and LAN2, and assign IP addresses and network masks to the three interfaces of Router 1, PC0, PC1 and PC2. Your assignment must be consistent with what is already assigned to Router 2.

■ **Answer:**

Network prefix of LAN1: 18.26.1.0/25
Network prefix of LAN2: 18.26.1.128/25;

Interface	IP Address	Netmask
Router1 - eth0	18.26.0.2	225.225.255.0
-eth1	18.26.1.1	255.255.255.128
-eth2	18.26.1.129	255.255.255.128
PC0 - eth0	18.26.0.3	255.255.255.0
PC1 - eth0	18.26.1.2	255.255.255.128
PC2 - eth0	18.26.1.130	255.255.255.128

■ **Example** What is route aggregation?

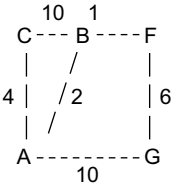
■ Answer For a group of subnets, one interface is selected at the router for forwarding packets. This is called as route aggregation. Route aggregation, if not configured properly, may result in routing loops.

To avoid looping, we usually configure a static route pointing to the aggregate route, with destination a Null interface on the Router. Any packet going to the Null interface is discarded. The Null interface acts as a Pit Bucket. So a packet with destination within an address prefix in the aggregate that does not have a route, will be routed using the static route to the Null interface.

■ **Example** A certain router B in a network receives the following link state updates from other nodes in the same network. Show the network topology, with nodes, edges and link costs, as constructed by B after it receives the above link state packets.

Update from	Sequence#	Neighbour/cost
C	5	B/8, A/4
A	20	B/2, C/4, G/10
C	10	B/10, A/4
F	50	B/1, G/6
G	100	F/6, A/10

■ **Answer** From the given information, we find that for station C, we consider most recent sequence number related delay. That is, 10 as the delay to C from B. Similarly, for B to A as 4. In the case of F, delay is 1(from link state information from F with sequence number 50). Like this, station B updates its details as shown below.



Suppose that the A—B link goes down. Say, A and B quickly re-compute shortest paths to other network nodes, but their link state updates are yet to reach their neighbours. Is there a possibility of packets looping in the network? Why?

■ **Answer:** Yes. There is a possibility of looping. After A and B do the re-computation, the shortest A—B path is through C, but C still thinks that the shortest C—B path is through A. So packet will be transiently stuck in an A—C—A… loop.

■ **Example** Consider a LAN with a maximum distance of 2 km. At what bandwidth would propagation delay (at a speed of 2×10^8 m/s) equal transmit delay (insertion delay) for 512 byte packets? What about 2000 byte packets?

■ **Answer:** Case (a): Packet Size = 512 bytes
Maximum Distance = 2 km
Speed of Light = 2×10^8 m/s
Propagation = Distance / Speed of Light
= 2000 m / 2×10^8 m/s
Transmit = Size / Bandwidth
= 512 x 8 bits / Bandwidth
Equating both,
Distance / Speed of Light = Size / Bandwidth
Bandwidth = Size x Speed of Light / Distance
= (512 x 8) bits x 2×10^8 m/s / 2000 m
= 409.6 Mbits/sec=409.6Mbps
Case (b): Packet Size = 2000 bytes
Bandwidth = Size x Speed of Light / Distance
= (2000 x 8) bits x 2×10^8 m/s / 2000 m
= 1600 Mbits/sec=1600Mbps

■ **Example** Suppose a 100-Mbps point-to-point link is being set up between the earth a new lunar colony. The distant from the moon to the earth is approximately 385,000 km, and data travels over the link at the speed of light – 3 x 108 m/s.

■ **Answer:**
(a) Calculate the minimum RTT for the link
Minimum RTT = 2 x Propagation
Propagation = Distance / Speed of Light
= 2 x 385000 km / 3 x 108 m/s = 2.57 sec
(b) Using the RTT as the delay, calculate the delay x bandwidth product for the link.

■ **Answer:** Delay x Bandwidth = 2.57 sec x 100 Mbits/sec
= 32MB
(c) What is the significance of the delay x bandwidth product computed in (b)?

■ **Answer:** This represents the amount of data the sender can send before it would be possible to receive a response.
(d) A camera on the lunar base takes pictures of the earth and saves them in digital format to disk. Suppose Mission Control on earth wishes to download the most current images, which is 30MB. What is the minimum amount of time that will elapse between the request for the data goes out and the transfer is finished?

■ **Answer:** We require at least one RTT before the picture could begin arriving at the ground (TCP would take two RTTs). Assuming bandwidth delay only, it would then take = 30MB/100Mbps = 2.4 sec to finish sending,

Thus, for a total time of 2.4 + 2.57 = 4.97 sec until the last picture bit arrives on earth.

■ **Example** Hosts A and B are connected to a switch (S) via 10-Mbps separate links(A----S----B). The propagation delay on each link is 40μs. S is a store-and-forward device; it begins retransmitting a received packet 20μs after it has finished received it. Calculate the total time required to transmit 10,000 bits from A to B
(a) as a single packet
(b) as two 5,000-bit packets sent one right after the other.

■ **Answer:** **Case (a): as a single packet**
Per link Transmit Delay = Size / Bandwidth
= 10^4 bits / 10 x 10^6 bits/sec
= 1000 μs
Total Transmission Time = (2 x 1000 + 2 x 40 + 20) μs = (2000 + 80 + 20) μs = 2100 μs

Case (b): as two 5,000-bit packets sent one right after the other
Transmit delay for 5000bit packet = 5000 bits/ 10 x 10^6 bits/ sec = 500μs
When sending as two packets, here is the possible sequence of events in accordance with time.
T=0 start
T=500μs A finishes sending packet 1, starts packet 2
T=540μs packet 1 finishes arriving at S
T=560μs packet 1 departs for B
T=1000μs A finishes sending packet 2
T=1060μs packet 2 departs for B
T=1100μs bit 1 of packet 2 arrives at B
T=1600μs last bit of packet 2 arrives at B
Total Transmit time = (3 x 500 + 2 x 40 + 1 x 20) μs
= (1500 + 80 + 20) μs = 1600 μs

■ **Example** A bit string, 0111101111101111110, needs to be transmitted at the data link layer. What is the string actually transmitted after bit stuffing?

■ **Answer:**

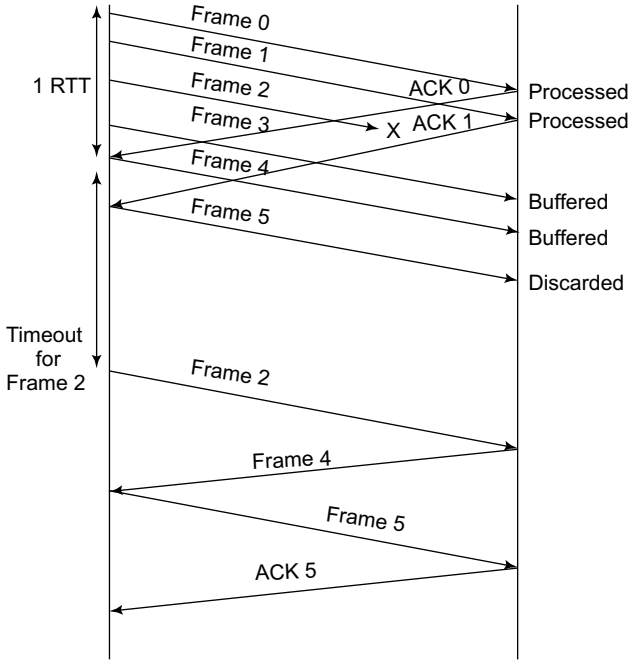
01111011111	0	011111	0	10
(data)	(Inserted 0)	(data)	(Inserted 0)	(data)

When five consecutive 1s have been transmitted from the body of the message, the sender inserts a 0 (as shown above) before transmitting the next bit.

■ **Example** Draw a timeline diagram (up to frame 7) that for the sliding window algorithm with SWS=4 frames and RWS=3 frames, when the third frame (frame 2) is lost. The

receiver use cumulative ACKs. Use a timeout interval of about 2 x RTT. Assuming that the transmit time (insertion delay) of a frame is equal to 0.25 RTT and the frames can be processed instantaneously if they arrive in order. On each data frame and ACK frame, you need to indicate the sequence number (start from 0). In addition, you need to indicate what action is taken by the receiver when it is received, for example, processed, buffered, and discarded.

■ **Answer:**



7.7 Objective Questions

1. What is the one service/guarantee that UDP provides to application layer protocols?
- A. Error detection
- B. Security
- C. Connection oriented service
- D Piggybacking

2. Packetisation of messages
- A. Is an encryption mechanism that enforces network security
- B. Helps prevent a single sender from monopolizing a shared communication link for an arbitrary amount of time
- C. Is an algorithm for data encoding in the physical layer

3. The traditional phone network is
- A. Circuit switched
- B. Datagram packet switched
- C. Neither of the above

4. The layered network architecture
- A. Refers to a type of network hardware
- B. Defines an interconnection topology of multiple communication networks
- C. Is a way of creating multiple levels of communication abstractions

5. The Internet is large and heterogeneous. However, it has a few invariables that define its architecture, such as
- A. The used hardware platform
- B. The used addressing scheme
- C. The used physical layer protocol

6. Circuit switching has the advantage of
- A. Requiring a signalling phase
- B. Very efficient allocation of shared resources
- C. Offering guaranteed connection bandwidth if needed

7. "Store and forward" most commonly refers to
- A. A congestion control algorithm
- B. A way of handling packets at network nodes
- C. A signalling protocol in virtual circuit packet switched networks

8. In network terminology, ATM most generally refers to
- A. A type of network adaptor
- B. A set of standards that define a network architecture
- C. A physical communication medium

9. The Internet is predominantly
- A. Circuit switched
- B. Datagram packet switched
- C. Neither of the above

10. Which of the following is generally NOT a function of the network layer?
- A. Forwarding messages from source to destination
- B. Interpretting destination addresses for routing purposes
- C. Ensuring that every packet is received reliably

11. What option below about "Store-and-Forward" technology is correct?
- A. The host(s) is used to store and forward the packets
- B. There is a dedicated path maintained in a network for this purpose
- C. It is mainly used in a packed switching network
- D. It is mainly used in a circuit switching network
- E. The major reason to use this technique is for network security.

12. Which of the following issues about STDM and FDM are true?

A. Both are the techniques used for network routing purposes
B. The max number of network flow is adjustable and resizable
C. The data flow for these two techniques are subject to under utilization
D. STDM is used for circuit switching and FDM is used in VC networking
E. STDM and FDM's data flow is transferring on demand.

13. Which option below about digital encoding could be a potential problem?
A. Encode makes 0 as low signal and 1 as high signal
B. Encode is to modulate electromagnetic waves
C. Signals propagate over a physical medium
D. Low signal (0) may be interpreted as no signal
E. There could be alternative encoding methods

14. Which option below about representing network as graph is wrong?
A. Nodes are part of the components of the graph
B. Node costs is used to represent values related to physical distance, capacity, delay, etc
C. Routers is used to support the protocols and algorithm in the dynamic approach
D. The assumption of edge cost in dynamic approach is known.
E. The distance vector algorithm maintaining the next hop value for every destination.

15. Which of the following option about ATM is wrong?
A. ATM is a connection-oriented packet-switched network
B. The Cells in ATM network are variable length
C. ATM used in both WAN and LAN settings
D. Different ATM AAL (ATM Adaptation Layer) provides different services for network applications
E. Segmentation and Reassembly (SAR) are necessary processes in ATM

16. Which one below about limitation of a Learning Bridge is wrong?
A. Learning Bridge uses Spanning Tree Algorithm to remove loops
B. There is limited scalability capability for Learning Bridge
C. There could be a transparency problem for Learning Bridge
D. Learning Bridge use VLAN to support scalability
E. It is easy for different types of network structures can be interconnected using Learning Bridge.

17. Which one below about Datagram Switching Network is wrong?
A. Connection setup phase is needed
B. Every packet contains the complete destination address
C. This is sometimes called connectionless model
D. Each Switch maintains a routing table for packets
E. Source host has no way of knowing if the network is capable of delivering a packet or if the destination host is even up

18. Which one below about VC (Virtual Circuit) Network is wrong?
A. Typically wait full RTT for connection setup before sending first data packet.
B. If a switch or a link in a connection fails, the connection is broken and a new one needs to be established.
C. Each data packet contains only a small identifier
D. Include only Outgoing Interface for packets transfer
E. It contains connection setup and data transfer phases

19. Which one below about Network Channels is wrong?
A. Request/reply (client/server) channel used by the file transfer and digital library application
B. Message Stream channel could be used by both video-on-demand and videoconferencing applications
C. Request/reply (client/server) channel has no need to guarantee all messages are delivered
D. Message Stream channel support both one or two-way traffic and delay properties
E. Message Stream channel needs to ensure that messages are delivered arrives in the same order in which they were sent.

20. Which one below is NOT a network function?
A. Provide Connectivity
B. Addressing
C. Resource Sharing
D. Switching and Data Forwarding
E. Server Application Supports

21. Four hosts (H1, H2, H3 and H4) are connected to one hub. H1's IP address is 192.168.7.33/24; H2's IP address is 192.168.120.7/16; H3's IP address is 192.168.7.43/16; H4's IP address is 192.1.168.7/28. Mark all pairs that can ping each other.

A. H1-H2 B. H1-H3
C. H1-H4 D. H2-H3
E. H2-H4 F. H3-H4

22. A hub that enables us to gather network performance information is
A. Hub B. Intelligent hub
C. Bridge D. Gracious hub

23. Correct point about spanning tree algorithm is
A. Calculates shortest distance between links
B. Ensure frames are not transmitted in an endless loop.
C. Forward frames along the most efficient route
D. None

24. _____is used for compressed audio and video where the data rate depends on the level of compression that can be achieved.
A. Constant Bit Rate (CBR) service
B. Variable Bit Rate (VBR) service
C. Available Bit Rate (ABR) service
D. None of the given

25. A form of addressing; in which a set of computers is assigned one address.
A. Broadcast B. Multicast
C. Packet Switch D. CRC
E. Bandwidth

26. _____ Program sends a message to a remote computer and reports whether the computer responds.
A. Ping B. Traceroute
C. ICMP D. None of the above

27. A Bridge can
A. Filter a frame B. Forward a frame
C. Extend a LAN D. Do all the above

28. A network with throughput T and delay D has a total of _____ bit in transit at any time.
A. T / D B. T x D
C. T + D D. None of the given

29. _____ is used for audio and video, since these have predefined maximum data rates.
A. Constant Bit Rate (CBR) service
B. Variable Bit Rate (VBR) service
C. Available Bit Rate (ABR) service
D. None of the given

30. Most NICs contain _____ circuitry that allows the NIC to operate independent of the CPU.
A. DMA(Direct Memory Access)
B. Multiplexer
C. Transceiver
D. None of the given

31. If the ASCII character G is sent and the character D is received, what type of error is this?
A. Single-bit B. Multiple-bit
C. Burst D. Recoverable

32. Computers attached to an ether use ____ in which a computer waits for the ether to be idle before transmitting a frame.
A. CSMA/CD B. CSMA/CA
C. TOKEN PASSING D. None of the given

33. The maximum size of an Ethernet segment is_____
A. 250 meters B. 500 meters
C. 700 meters D. None of the given

34. In Direct point to point communication adding the Nth computer requires_____ new connections.
A. N2 B. N-1
C. (N2 –N)/2 C. None of the given

35. The number of connections needed for N computer in direct point to point communication is equal to
A. (N2-N)/2 B. N (N-1)
C. N2 D. None of the given

36. Hardware that calculates a CRC uses two simple components.
A. AND unit and XOR unit
B. Shift register and XOR unit
C. Shift register and AND unit
D. None of the given

37. One repeater _____two repeaters _____ the maximum cable length limitation.
A. Doubles, cancel
B. Doubles, triple
C. Square roots, cude roots
D. And, triple

38. A Bridge forwards or filters a frame by comparing the information in its address table to the frame's
A. Layer 2 source address
B. Source node's physical address
C. Layer 2 destination address
D. Layer 3 destination address

39. ATM assigns each VC a _____ identifier that is divided two parts to produce a hierarchy.
A. 21-bit B. 22-bit
C. 23-bit D. 24-bit

40. Basic LAN technologies such as Ethernet, Token Ring, and FDDI use a
A. Connectionless service paradigm
B. Connection-oriented service paradigm
C. Both Connectionless and Connection-oriented service paradigm
D. None of the given

41. The operation of subnet is controlled by
A. Network Layer B. Data Link Layer
C. Data Layer D. Transport Layer

42. Multiplexing and Demultiplexing of Network connections is by _____Layer.
A. Network Layer B. Data Layer
C. Data Link Layer D. Transport Layer

43. In a network with 25 computers, which topology would require the most extensive cabling.
A. Bus B. Mesh
C. Star D. Tree

44. A cable break in a _____ topology stops all transmissions.
A. Bus B. Mesh
C. Star D. Primary

45. Which topology requires a central controller or hub?
A. Mesh B. Star
C. Bus D. Ring

46. Which topology requires a multipoint connection?
A. Mesh B. Star
C. Bus D. Ring

47. In _____ transmission, the channel capacity is shared by both communicating devices at all times.
A. Simplex B. Half-duplex
C. Full-duplex D. Automatic

48. A configuration of N LANs is linearly connected by 4 bridges. By the time we reach the Nth LAN how many discovery frames will be circulating?
A. N4 B. 4
C. 4N D. N

49. In _____ all frames are given to the computer, not to those addressed.
A. Promiscuous mode B. Miscues mode
C. Normal mode D. Special Mode

50. In transparent bridges the failures are handled by_____.
A. Host B. Bridge
C. Network layer D. Router

51. _____bridge operates in promiscuous mode.
A. Transparent bridge B. Selective flooding
C. Source routing D. Remote bridges

52. Source routing bridges in the same LANs must have _____ bridge number.
A. Same B. Different
C. Source D. Destination

53. Which type of switching uses the entire capacity of a dedicated link?
A. Circuit switching
B. Datagram packet switching
C. Virtual circuit packet switching
D. Message switching

54. The channel efficiency of bit-map protocol at low load is _____.
A. d/(N+d)
B. d/(d+1)
C. d/(d+log2N)
D. N/(d+log2d)

55. The channel efficiency of bit-map protocol at high load is _____.
A. d/(N+d) B. d/(d+1)
C. d/(d+log2N) D. N/(d+log2d)

56. The channel efficiency of binary-countdown protocol is _____.
A. d/(N+d) B. d/(d+1)
C. d/(d+log2N) D. N/(d+log2d)

57. The first collision free protocol is _____.
A. Binary countdown B. Basic bitmap
C. Reservation protocol D. SAP

58. The problem of a station not being able to detect a potential competitor for the medium because it thinks that there is activity between them is called
A. Exposed Station Problem
B. Collision Avoidance Problem
C. Hidden Station Problem
D. Access Grant Problem

59. You have determined that there is network congestion caused by the portion of your organisation's network that is used by research engineers. What device can help you reduce the congestion between the research engineers' network and the rest of the organisation's network?
A. A router B. A bridge
C. A speed filter D. A bandwidth filter

60. Which of the following are protocol stacks?
A. TCP/IP B. 802.3
C. 100BASET D. FTP
E. AppleTalk

61. Traditional LANs eg: Ethernet Bus, Token Ring use
A. Circuit Switching B. Packet Switching
C. Frame Relay D. Packet Broadcast

62. Collision Detection in Baseband Ethernet Bus CSMA/CD (IEEE 802.3) LANs is based on
A. voltage threshold, voltage on (passive) bus exceeds a certain level

B. Different frequencies, as in case of human communication in a group
C. CD stands for Code Division (multiplexing) hence no collision detection
D. Logic using active circuits, than amplitude (voltage), as in Star Networks.

63. In LAN terminology, 10 Base 5 signifies
A. Ethernet Bus 10 metre long, digital, 5Mbps
B. Ethernet Bus 10 Mbps, digital, 500 metre
C. Token Bus 10 Mbps, digital, 500 metre
D. Ethernet or Token Bus, 10 Mbps, digital, 500 metre

64. Wavelength Division Multiplexing (WDM) optical networks is based on
A. Asynchronous Time Division Multiplexing, fixed time slots, anyone can transmit
B. Synchronous Time Division Multiplexing, fixed time slots, all slots reserved
C. Space Division Multiplexing—Each physical link carries only one wavelength
D. Same principle as frequency division multiplexing different colours on same link

65. CRC(cyclic redundancy check) is good for serial lines
A. For byte oriented transmissions only (each character has start stop, parity bits)
B. For checksum purposes, it is checksum of the information eg: WDRL $50
C. For bit serial transmissions only
D. For Quadrature Phase Shift Keying (QPSK) analog transmission, using polynomials

66. A carrier with the frequency of 1200 khz carries modulating sine wave of 100 khz. The resultant spectrum of AM SSB (upper sideband) with transmitted carrier is
A. 1100 khz, 1200 khz, 1300 khz
B. 1100 khz, 1200 khz
C. 1300 khz
D. 1200 khz, 1300 khz

67. LRC (Longitudinal Redundancy Check) is good
A. For byte oriented transmissions only (each character has start stop, parity bits)
B. For checksum purposes, it is checksum of the information
C. For bit serial transmissions only
D. For parallel lines

68. In LAN terminology, 10 Base 5 signifies
A. Ethernet Bus 10 metres long, digital, 5Mbps
B. Token Bus, 10 Mbps, digital, 500 metres
C. Ethernet Bus 10 Mbps, analog, 500 metres
D. Ethernet Bus 10 Mbps, digital, 500 metres

69. A Token Ring LAN uses the following type of encoding
A. Differential Manchester
B. Manchester
C. NRZ-I
D. Bipolar-AMI

70. In regards to Statistical Time Division Multiplexing, which statement is true?
A. Multiple signals may be transmitted simultaneously
B. Time slots may be wasted when a particular source has nothing to send
C. Additional addressing information is included in each time slot
D. Time slots are divided equally amongst users/sources

71. Identify the class of the following IP address 5.5.6.7
A. CLASS A B. CLASS B
C. CLASS C D. CLASS D

72. Identify the class of the following IP address 229.1.2.3.
A. CLASS A B. CLASS B
C. CLASS C D. CLASS D

73. Identify the following IP address 169.5.0.0.
A. Host IP address
B. Direct broadcast address
C. Limited broadcast address
D. Network address

74. Identify the following IP address 160.5.255.255.
A. Host IP address
B. Direct broadcast address
C. Limited broadcast address
D. Network address

75. A device has two IP addresses. This device can be
A. A computer B. A router
C. A gateway D. All

76. A device has two IP addresses. One address is 192.123.46.219. The other address can be
A. 192.123.46.220 B. 192.123.46.0
C. 192.123.47.219 D. None

77. A private network with 300 computers wants to use a netid. What is a good choice?
A. 10.0.0 B. 172.16
C. 192.68.0 D. None

78. IP address is
A. 32 bits B. 48 bits
C. 16 bits D. None

79. Socket is
A. 16 bits B. 32 bits
C. 48 bits D. None

80. Why are packets divided?
A. To take care of noise
B. For not to monopolize channel
C. To manage with less buffer overheads
D. All

81. In a 3-bit sequence number field (eg: I frame -Ns, Nr, 2^3 =8; S = W +1), the size of the _transmit_ window in Go-Back-N protocol is
A. 3 B. 8
C. 7 D. 1

82. 10 Base 2 means:
A. Ethernet BUs(CSMA/CD) 10 metres, digital, 2 Mbps
B. Token Ring 10 Mbps, digital, 200 metres (actually 185m)
C. Ethernet Bus or Token Bus:10 Mbps, digital, 200 metres (actually 185m)
D. Ethernet Bus (only) 10 Mbps, digital, 200 metres (actually 185m)

83. Digital and Analog signal representations are handled at the following layer
A. Data Link B. Physical
C. Network D. Applications

84. Layer dynamically monitors traffic situation, and increases/decreases end-end credits (admittance control, number of packets into network)
A. Transport B. (Intra-)Network
C. Inter-network D. Data Link

85. Layer's role is dialogue management-usually half duplex, your turn etc., synchronisation, error recovery
A. Transport B. Session
C. Presentation D. Application

86. The physical layer is concerned with the transmission of ___ over physical medium.
A. Programs B. Dialogs
C. Protocols D. Bits

87. As a data packet moves from the lower to the upper layers, headers are _____.
A. Added B. Subtracted
C. Rearranged D. Modified

88. When data is transmitted from device A to B, the header from A's layer 5 is read by B's __ layer.
A. Physical B. Transport
C. Session D. Presentation

89. Which layer functions as a liaison between user support layers and network support layers?
A. Network layer B. Physical
C. Transport D. Session

90. What is the main function of the transport layer?
A. Node-to-node delivery
B. Process-to-process message delivery
C. Synchronisation
D. Updating routing tables

91. Session layer check points
A. Allow just a portion of a file to be resent.
B. Detect and recover errors.
C. Control the addition of headers.
D Are involved in dialog control.

92. Which of the following are application layer service?
A. Network virtual terminal
B. File transfer, access and management
C. Mail service
D. All

93. When a host on network A sends a message to a host on network B which address does the router look at?
A. Port B. IP
C. Physical D. None

94. To deliver a message to the correct application program running on a host, the __ address must be specified?
A. Port B. IP
C. Physical D. Both A & B

95. IPV6 has ___ bit address.
A. 16 B. 32
C. 128 D. Variable

96. ICMPV6 includes
A. IGMP B. ARP
C. RARP C. Both A & B

97. In a noisy environment the best transmission medium is
A. Twisted pair B. Coaxial cable
C. Optica fibre D. The atmosphere

98. What protects the coaxial cable from noise?
A. Inner conductor
B. Diameter of the cable
C. Outer conductor
D. Insulating material

99. The inner core of an optical fiber is ___ in composition.
A. Glass or plastic B. Copper
D. Bimetallic D. Liquid

100. Radio communication
A. 3KHz to 300KHz B. 300KHz to 3 GHz
C. 3KHz to 300 GHz D. 3 KHz to 3000 GHZ

101. 10Base2 uses ___ cable while 10Base5 uses
A. Thick coaxial, thin coaxial
B. Twisted pair, thick coaxial
C. Thin coaxial, thick coaxial
D. Fibre optic, thin coaxial

102. ___ specifies a star topology featuring a central hub and unshielded twisted pair wire as the medium.
A. 10Base5 B. 10Base 2
C. 10BaseT D. None

103. The __ houses the switches in token ring.
A. NIC B. MAU
C. 9-pin connector D. Transreceiver

104. A station in a token ring
A. Examines destination address
B. Regeneration of the frame
C. Passes the frame to next station
D. All

105. FDDI is more efficient than a regular ring as
A. No times are needed
B. The node that has the token can send more than one frame
C. The AC field is eliminated
D. The added priority and reservation option

106. Which is suitable for a dedicated link for best results?
A. Circuit switching
B. Datagram approach to packet switching
C. VC approach for packet switching
D. Message switching

107. In which of the following all the datagrams of a message follow the same route?
A. Datagram approach to packet switching
B. VC approach for packet switching
C. Both A & B
D. None

108. X.25 uses ___ for end-to-end transmission.
A. Message switching
B. Circuit switching
C. The datagram approach to packet switching
D. The VC approach to packet switching

109. The physical layer protocol used in X.25 protocol is
A. RS-232 B. X.21
C. DB-15 D. DB-37

110. X.25 requires error checking at ___ layer.
A. Physical B. Data link
C. Network D. Both B & C

111. Frame relay needs error checking at __ layer.
A. Physical B. Data link
C. Network D. None

112. Why is so much error checking is needed in X.25?
A. X.25 uses large bandwidth
B. When X.25 is started medium was very unreliable
C. When X.25 is started the switching is very unreliable
D. When X.25 is started the multiplexing was not available

113. In frame relay __ bit in the address filed is set to signify the last address byte?
A. DE B. EA
C. C/R D. FECN

114. In X.25 Permanent VC address is
A. EA B. FECN/BBCN
C. DE D. DLCI

115. ATM uses
A. Twisted pair B. Coaxial cable
C. Fibre D. Atmosphere

116. As ATM is _____, cells do usually arrive out of order because the cells follow the same path.
A. Asynchronous B. Synchronous
C. Multiplexed D. A network
E. Connection oriented

117. ___ layer in ATM has 53-byte cells as an end product.
A. Physical
B. ATM
C. Application adaptation
D. Cell transformation

118. Which application adaptation layer type can process a data stream having a non constant bit rate?
A. AAL1 B. AAL2
C. AAL3/4 D. AAL5

119. ___ support constant bit rate.
A. AAL1 B. AAL2
C. AAL3/4 D. AAL5

120. The end product of the AAL5 SAR is a data packet that is
A. Variable length B. 48 bytes
C. 44 to 48 bytes D. > 48 bytes

121. _____ field in a cell header in ATM determines whether the cell can be dropped or not.

A. VPI B. VCI
C. CLP D. GFC

122. Find odd one out
A. Bridge B. Transceiver
C. Router D. Repeater

123. Gateway
A. Protocol conversion B. Packet re-sizing
C. Data rate adjustment D. All

124. Gateways function in which OSI layer?
A. Lower 3 B. Upper 3
C All 7 D. None

125. A bridge has access to the ___ address of a station on the same network.
A. Physical B. Network
C. SAP D. All

126. A packet from an Ethernet requires a ___ before it can be routed to an FDDI network.
A. Repeater B. Bridge
C. Router D. Gateway

127. Piggybacking
A. Physical layer B. Data link layer
C. Network layer D. Transport layer

128. Find odd-one out
A. Network layer B. Physical
C. Datalink D. Transport

129. Flow control is in
A. Transport B. Datalink
C. Physical D. Application

130. Why framing has to be done at a DLL connected to a link?
A. Such that no user service monopolizes the link
B. Buffer requirements reduces
C. To have data transfer even on the erroneous channel with some error rate
D. All

131. Repeaters are
A. SW entities B. HW entities
C. In physical layer D. Both B & C

132. Baud and bit rates of a channel are same if the number of symbols used are equal to
A. 1 B. 2
B. 4 D. None

133. Find odd one out
A. Data link B. Transport
C. Network C. None

134. Parity bit stuffing is
A. Error detection
B. Error correction
C. Both A & B
D. None

135. Application layer
A. Frame
B. Packet
C. Message
D. None

136. A 10V 2khz user sine signal is modulating a 30V 200 khz carrier sine signal. The resultant fourier spectrum under a Double Side Band with Transmitted Carrier (DSBTC) is
A. 10V 2 khz, 40V 200 khz
B. 10V 198 khz, 30V 200 khz, 10V 202 khz
C. 5V 198 khz, (0V) nothing at 200 khz, 5V 202 khz
D. 5V 198 khz, 30V 200 khz 5V 202 khz

137. The above data (in Q51) under a DSB SC (suppressed Carrier) is
A. 10V 2 khz, 40V 200 khz
B. 10V 198 khz, 30V 200 khz, 10V 202 khz
C. 5V 198 khz, (0V) nothing at 200 khz, 5V 202 khz
D. 5V 198 khz, 30V 200 khz 5V 202 khz

138. The data in Q51, using Single Sideband (SSB) Transmitted Carrier, Upper Sideband is
A. 5V 198 khz,
B. 5V 198 khz, 30V 200 khz
C. 30V 200 khz, 5V 202 khz
D. 5V 202 khz

139. The data in Q51, using Single Sideband (SSB) Suppressed Carrier, Lower Sideband is
A. 5V 198 khz,
B. 5V 198 khz, 30V 200 khz
C. 30V 200 khz, 5V 202 khz
D. 5V 202 khz

140. Data in Q1 with minor modification (user signal = cosine):- A 10V 2khz user cosine signal is modulating a 30V 200 khz carrier sine signal. The resultant fourier spectrum under a Double Side Band with Transmitted Carrier (DSBTC) is
A. -5V 198 khz, 30V 200 khz, 5V 202 khz
B. 5V 198 khz, 30V 200 khz, -5V 202 khz
C. -5V 198 khz, 30V 200 khz, -5V 202 khz
D. 5V 198 khz, 30V 200 khz, 5V 202 khz

141. A QPSK 16 levels modem is currently at +90 degrees 2V. The next group of data is 0111 (5V, + 135 degrees). The output is

A. 7V, +225 degrees B. 5V, +225 degrees
C. 5V, +135 degrees D. 7V, +135 degrees

142. The encoding scheme for the bits 00000 under NRZ-L is
A. Low, 0V
B. A transition at the beginning of interval for each 0
C. High, 5V
D. For each bit 0, alternates between high +5V, and low -5V, returning to 0V after each bit 0.

143. The encoding scheme for the bits 00000 under NRZ-I, intially, at high 5V s
A. Low, 0V
B. A transition at the beginning of interval for each 0
C. High, 5V
D. For each bit 0, alternates between high +5V, and low -5V, returning to 0V after each bit 0.

144. The encoding scheme for the bits 1111 under NRZ-I, intially, at high 5V is
A. High, high, low, low B. Low, high, low, high
C. Low, low, high, high D. High, low, high, low

145. Manchester encoding is used in disks over NRZ-I or NRZ-L due to
A. Manchester code has less transitions rate than NRZ's
B. Manchester code has transition in the middle for each 0 or 1, hence this makes it self-clocking
C. Manchester code has transition in the middle for each 0 or 1, hence this allows collison detection, on voltage threshold
D. The transitions in Manchester encoding alternate for 0's and 1's hence this makes it suitable for alternating current (AC)

146. Bi-polar AMI signals have
A. More transitions than NRZ-L, NRZ-I, more power than Manchester
B. Less transitions than NRZ-L, NRZ-I, more power than Manchester
C. Less transitions than NRZ-L, NRZ-I, less power than Manchester
D. Least average power, due to each bit 1 alternating between +5V, -5V (cancellation effect in average amplitude)

147. Traditional LANs such as Token Ring, Ethernet Bus use
A. Circuit Switching
B. Packet Switching
C. Radio Frequency (RF) wireless
D. Packet Broadcasting

148. Traditional LANs such as Token Ring (IEEE 802.5), Ethernet Bus (IEEE 802.3) use
A. NRZ-I
B. Bi-polar AMI
C. Manchester
D. Differential Manchester

149. Ethernet Bus is based on
A. Reservation bus
B. Collision bus--collision detect, random backoff, LWT
C. Collision avoidance (CA) bus-using tokens
D. Collision avoidance (CA) bus-allocated (fixed) time slots

150. Collision detection in IEEE 802.3 (CSMA/CD) is based on
A. Frequency --bus detects 2 different frequencies of users
B. Clocking -- bus detects clock transitions for 2 stations
C. Logic -- bus uses logic to detect to detect 2 stations are active
D. Amplitude--voltage level exceeds a threhold.

151. Coding method in IEEE 802.3 (CSMA/CD) -Ethernet Bus is based on
A. NRZ-I
B. Bi-polar AMI
C. Manchester
D. Differential Manchester

152. Manchester encoding in IEEE 802.5 Token Ring is based on
A. Minimum transitions
B. Clocking -- stations are kept in synchronisation due to self-clock
C. Logic--stations use the self-clocking info to decide who has token
D. Amplitude--each 0 or 1 has a high, hence voltage threshold can be used for collision detection.

153. WDM is based on
A. Time Division Multiplexing(TDM)
B. Space Division Multiplexing (SDM)
C. Wavelength Division Multiplexing--principle same as FDM, different colours (optical-easier to represent as wavelengths, than frequency)
D. Frequency Division Multiplexing (FDM)--in the Radio Frequency Frequency Modulation eg: 101.1 Mhz range

154. Collision Avoidance is NOT done in

A. Ethernet Bus (CSMA/CD) IEEE 802.3
B. Token Ring IEEE 802.5
C. FDDI (optical, dual-ring, one ring normal, other backup)
D. Token Bus IEEE 802.4

155. Collision Avoidance in a bus topology is used in
A. Ethernet Bus (CSMA/CD) IEEE 802.3
B. Token Ring IEEE 802.5
C. FDDI (optical, dual-ring, one ring normal, other backup)
D. Token Bus IEEE 802.4

156. LLC/MAC layers in LANs are finer subdivisions at the following layer
A. Physical B. Data Link
C. Network D. Transport

157. LANs do not have
A. Physical B. Data Link
C. Network D. Transport

158. LLC (Logical Link Layer)
A. is independent of MAC. LLC is same for Ethernet bus, token ring. LLC is a layer below MAC and LLC interfaces to PL.
B. is unique to each MAC
C. is grouped for bus, ring, wireless LAN's. That is, all bus MAC's use one type of LLC, all ring MAC's use another LLC.
D. is independent of MAC. Same LLC is emploted in Ethernet bus, token ring.

159. A 16 levels extended QPSK modem uses a 2400 hz carrier(W) on the telephony network. The bit rate is
A. 20Kbps B. 19.2Kbps
C. 1Mbps D. None

160. A channel uses a 2400 hz carrier. The signal to noise ratio snr = 30db. The max bit rate
A. 2400 B. 24000
C. 24Mbps D. None

161. A Stop and Wait protocol has the following data: Frame size (L)= 1000 bits; Transmission Speed (R) = 1 Mps Distance (D) = 10 kms; Velocity of Propagation (V) = 2*10^8 m/sec. Calculate the utilization of the link, ignoring effects of ACK, and CPU.
A. 1 B. 0.9
C. 0.5 D. None

162. Calculate the utilisation of the above link(Q86), probability of error, P=0.2.
A. 0.9 B. 1.0
C. 0.72 D. None

163. Calculate the utilisation of the link in Q86, under a Selective Repeat Protocol, Window, W=7, with no errors.
A. 0.9 B. 1.0
C. 0.72 D. None

164. Calculate the utilisation of the link in Q86, under a Selective Repeat Protocol, Window, W=7, with probability of error, P =0.2.
A. 1.0 B. 0.9
C. 0.8 D. None

165. An 8 level extended QPSK modem has a bit rate (C) = 12,000 bps. Calculate the analog signalling frequency.
A. 2000 B. 4000
C. 8000 D. None

166. For the analog frequency in Q90, with signal to noise ratio snr = 20db. Calculate the theoretical maximum bit rate.
A. 2000 B. 10000
C. 13400 D. None

167. A Stop and Wait protocol has the following data: Frame size (L)= 1000 bits; Transmission Speed (R) = 10 Mps ACK frame size (Lack) = 100 bits. Distance (D) = 100 kms; Velocity of Propagation (V) = 2*10^8 m/sec. Calculate the utilisation of the link, ignoring effects of CPU.
A. 1.0 B. 0.9
C. 0.09 D. None

168. Calculate the utilisation of the above link(92), probability of error, P=0.2
A. 1.0 B. 0.09
C. 0.072 D. None

169. Calculate the utilisation of the link in Q92, under a Selective Repeat Protocol, Window, W=7, with no errors.
A. 1.0 B. 0.09
C. 0.054 C. 0.072

170. Number of frames sent at a time in ARQ are
A. 1 B. w
C. 3 D. None

171. Time slot width in pure Aloha is
A. One B. Half
C. Frame time D. None

172. Ethernet addresses are
A. 2 bytes B. 6 bytes
C. Both A & B D. None

173. Padding
A. Ethernet B. IEEE 802.5
C. IEEE 802.6 D. None

174. Priority
A. IEEE 802.4 B. IEEE 802.5
C. Both A & B D. None

175. Minimum Frame size limitation is in
A. IEEE 802.3 B. IEEE 802.4
C. IEEE 802.5 D. All

176. Frame time in T1 carrier
A. 1Micro second B. 125 micro seconds
C. 100 ms D. None

177. Number of bits for common channel signalling in T1 carrier is
A. 1 B. 24
C. 25 D. None

178. Number of bits for common channel signalling in E1 carrier is
A. 16 B. 32
C. 48 D. None

179. Number of bits for channel associated signaling in E1 carrier are
A. 16 B. 32
C. 48 D. None

180. Date rate of E2 carrier
A. 1.054 Mbps B. 1.554 Mbps
C. 2.048 Mbps D. 8.848 Mbps

181. Why frame time in T1 carrier should be 125 micro seconds?
A. To support real time digital voice
B. It is design parameter
C. To have delay
D. None

182. Charging in circuit switching
A. Based on connection time
B. Based on number of connections
C. Based on amount of data transferred
D. None

183. A TSI switch is having memory with 100 ns access time. How many lines it can support?
A. 100 B. 625
C. 1024 D. None

184. Propagation delay is more in
A. Twisted pair
B. Coaxial cable
C. Satellite channels
D. Fibre

185. ATM data rate
A. 1.554 Mbps B. 2.048 Mbps
C. 622.08 Mbps D. 155.52 Mbps
D. Both C and D

186. In Q51 A 10V 2khz user signal is modulating a 30V 200 khz carrier signal. The fourier spectrum under a frequency modulated (FM) is most likely
A. 5V 198 khz, 30V 200 khz 5V 202 khz
B. A triangle centred at 40V 200khz, dropping to 0V at 198,202 khz
C. Rectangle, centred at 40V 200khz, ending at 198, 202 khz (40V)
D. Rectangle, centred at 30V 200khz; end frequencies, (symmetric) of rectangle depends on how user signal signal amplitude modulates carrier frequency (width of rectangle has no relevance to user frequency)

187. Routing of actual IP packets is done through
A. Network address B. Socket address
C. IP address D. None

188. Netid in class C type of networks occupies __ bits (excluding class type bits)
A. 7 B. 21
C. 14 C. None

189. Number of hosts which class A type of network can support
A. 16777216 B. 16777116
C. 16000000 D. None

190. Practically how many class A type networks are possible?
A. 256 B. 126
C. 100 D. None

191. The two left most bits of class B network are
A. 00 B. 10
C. 01 D. 11

192. If the first number of an IP address represented in dotted decimal is 193 then network is
A. Class A B. Class B
C. Class C D. Class D
E. Class E

193. In an IP packet
A. Hostid part in source address can not be 0's
B. Hostid part in destination address can not be 0's
C. Hostid part in source address can not be 1's
D. Hosted part in destination address can be all 1's
E. All are valid

194. Destination address of an IP packet which is to be broadcasted in local LAN from which the IP packet originated is

A. All 1's in hostid B. All 1's in netid
C. All 0's in hostid D. All 0's in netid
E. All 32 bits to be 1's

195. Destination address of an IP packet which is to be broadcasted in remote LAN
A. All 1's in hostid B. All 1's in netid
C. All 0's in hostid D. All 0's in netid
E. All 32 bits to be 1's

196. Loopback addresses
A. 127.x.x.x
B. To test the software on the machine without really having physical network
C. Class A type address
D. Can be used as a destination address only

197. All 32 bits are 0's in an IP address
A. Is used by diskless machines
B. Class A type address
C. Can be only source address
D. Can be recognised by bootstrap server
E. All

198. What destination address can be used to send a packet from a host with IP address 189.1.1.2 to all hosts on the same network?
A. 189.0.0.0 B.. 189.255.255.255
C. 255.255.255.255 D. None

199. What is the netmask for class C type network?
A. 255.0.0.0
B. 11111111111111110000000000000000
C. 11111111111111111111111100000000
D. 255.255.255.255
E. 255.255.255.0

200. Which is true about the following IP address 242.1.2.4?
A. Netid is 241 B. Class E
C. Hostid is 1.2.4 D. None

201. A host with an IP address of 144.2.2.1 needs to test internal software. What is the destination address in the packet?
A. 127.1.1.1 B. 144.0.0.0
C. 144.255.255.255 D. 127.0.0.0
E. Both A & D

202. Unguided media is
A. Coaxial cable B. Twisted pair
C. Fiber D. Microwave

203. Frequency limiting
A. Is the property of the medium
B. Is used by using filter to limit the amount of bandwidth available to customer
C. A & B
D. None

204. Baud is a
A. Signaling speed
B. Rate is same as bit rate for binary valued channel
C. Is voltage transitions for unit time
D. All

205. Limiting the bandwidth limits the data rate
A. For noisy channels B. For perfect channels
C. Both A & B D. None

206. Voice grade line is
A. 3-4KHz B. 3-4MHz
C. 5-6MHz D. None

207. Bandwidth of a coaxial cable
A. Is proportional to its inner core diameter
B. Is proportional to outer mesh spacing
C. Depends on the length of the cable
D. None

208. Error rates in fiber is
A. 1 in 1000 B. 1 in 10000
C. 1 in 100 D. Almost zero

209. Fiber has
A. Huge bandwidth B Less error rate
C. Fast D. Uni-directional
E. All

210. Fiber is preferred over copper wire as
A. Bandwidth is higher
B. Low attenuation
C. No of repeaters required will be scarce
D. Security against wiretapping is better
E. All

211. Propagation delay is more in
A. Twisted pair B. Coaxial cable
C. Fiber D. Satellites
E. None

212. V.32 bis data rate is
A. 9600 B. 2400
C. 14400 D. None

213. V.32 modem data rate is
A. 9600 B. 2400
C. 14400 D. None

214. Constellation diagram of V32 bis contains ___ no. of points.
A. 16 B. 32
C. 64 D. None

215. 28800 bps is
A. V.34 B. V.32
C. V.32bis D. None

216. ___is prone to noise very much.
A. Frequency B. Amplitude
C. Wavelength D. Phase
E. None

217. Current day modems employs
A. Ascii coding B. Ebcdic coding
C. Trellis coding D. Differential pulse code
E. None

218. Null modem will have ___ circuits.
A. 1 B. 2
C. 3 D. None

219. Codec is
A. Used in end office
B. Coder-decoder
C. Works on multiple channels
D. Does A/D and D/A conversion
E. All

220. Overhead in T1 carrier is
A. 193 bits B. 25 bits
C. 1 bit D. None

221. Out-of-band signaling bits in T1 carrier are
A. 193 B. 25
C. 1 D. 24
E. None

222. Why frame time in T1 carrier is 125 micro seconds?
A. It is standard
B. Users interest
C. For realtime digital voice commuinication

223. E1 carrier data rate is
A. 1.544Mbps B. 2.048Mbps
C. 6.468Mbps D. None

224. Circuit switching
A. Charging is based on connection time
B. Every packet traverse in the same route
C. No need of packet buffering & congestion control
D. All

225. For realtime performance ____ is preferred.
A. Circuit switching B. Packet switching
C. Both D. Message switching

226. Physical copper path is created in
A. Circuit switching B. Packet switching
C. Message switching D. Both A & C
E. None

227. Store-and-forward network uses
A. Circuit switching B. Packet switching
C. Message switching D. None

228. Telegrams uses (in olden days)
A. Circuit switching B. Message switching
C. Packet switching D. None

229. ___ suitable for interactive traffic.
A. Circuit switching B. Packet switching
C. Message switching D. None

230. Packets reach in same order in
A. Circuit switching B. Message switching
C. Packet switching C. None

231. Bandwidth utilisation is better in
A. Circuit switching B. Message switching
C. Packet switching D. None

232. For real time traffic ___ is suitable.
A. Circuit switching B. Message switching
C. Packet switching D. None

233. Main drawback of message switching is
A. Cost
B. Delays because of monopolisation of channel by one application
C. Complexity
D. None

234. A TSI switch is used which is equipped with RAM of 100 nano seconds. Frame period is 125 micro seconds. Then no. of lines which it can support is
A. 100 B. 500
C. 625 D. None

235. ATM uses
A. Circuit switching B. Message switching
C. Packet switching D. None

236. PSTN and N-ISDN uses ___
A. Circuit switching B. Message switching
C. Packet switching D. None

237. Virtual circuits
A. Actual service is offered through packet switching
B. Actual service is offered through circuit switching
C. Are either permanent or switched
D. None

238. Setup is not needed in
A. Circuit switching
B. Permanent virtual circuit
C. Switched virtual circuit
D. None

239. In ATM cells

A. Not necessarily come from alternate sources
B. 53 bytes size
C. Empty data cells are acceptable
D. All

240. Does ATM requires only fiber?
A. Yes B. No

241. Television channels are 8MHz wide. How many bps can be sent if four-level digital signals are used. Assume channel is ideal.
A. 32Mbps B. 32MB/s
C. 16Mbps C. 16MB/s

242. Responsibilities of data link layer is
A. Framing
B. Deframing
C. Acknowledgement management
D. Piggybacking
E. All

243. No. of frames sent at a time in ARQ protocol.
A. 1 B. 2
C. 3 D. Many

244. Does sequence numbers are necessary in ARQ?
A. Yes B. No

245. _____is error correction technique.
A. Bit stuffing B. CRC
C. Single parity bits D. None

246. Flow control has
A. Data Link layer B. Physical layer
C. Network layer D. Transport layer

247. Routing is done at
A. Data Link layer B. Physical layer
C. Network layer D. Transport layer

248. If the timer is set to a small value in DLL then
A. It expires too frequently thus retransmissions will takes place frequently
B. Bandwidth gets wasted
C. Both A & B
D. Both

249. The data bits are 1101011011, generator polynomial is 10111 then checksum is
A. 1001 B. 1110
C. 0000 D. None

250. The data bits are 11010110111110, generator polynomial is 10111 then checksum is
A. 1001 B. 1110
C. 0000 C. None

251. Do we require any protocol if the channel is ideal, stations are powerful, having infinite buffer space?
A. Yes B. No

252. Sliding window protocols are
A. Also known as pipelining
B. Useful in satellite networks
C. Needed where propogation delay is very high
E. All

253. Size of the acknowledgment frame when piggybacking is used is
A. Very small
B. No ack at all
C. Only out bound traffic is not existing ack is sent
D. Both B & C
E. None

254. Nak frames are seen in
A. ARQ B. PAR
C. Goback n D. Selective repeat
E. None

255. If the bit string 0111001111001 is bit stuffed then stuffed string is

256. Error-correcting codes
A. ASCII B. Hamming
C. EBCDIC D. DPCM

257. A code has following code words then the distance is __ 0000000000, 0000011111, 1111100000, and 1111111111.
A. 2 B. 5
C. 4 D. None

258. The above code can correct ___ errors.
A. Single B. Five
C. Double D. None

259. HDLC uses
A. Information frame B. Supervisory frame
C. Unnumbered frame D. All

260. Address field in HDLC bit oriented frame is
A. 8 bytes B. 32 bits
C. 8 bits D. None

261. HDLC uses
A. ARQ B. PAR
C. Sliding window D. None

262. HDLC uses ___ sequence number.
A. 3 bit B. 1 bit
C. n bit D. None

263. In Shell account
A. www browsing is not possible
B. Mail can be sent
C. Character user interface only supported
D. All

264. A channel has a bit rate of 4kbps and propagation delay of 20msec. For what range of frame sizes does stop-and-wait protocol can give an efficiency of 50%?

265. A 100km long cable runs at T1 data rate. The propagation speed in the cable is 2/3 the speed of light. How many bits can fit into the cable ?

266. Frames of 1000bits are sent over a 1-Mbps satellite channel using ARQ protocol. Acknowledgments are always piggybacked. The headers are very short. Calculate channel utilisation.

267. Maximum efficiency of pure ALOHA is

A. 0.18 B. 0.36
C. 1 D. None

268. Maximum efficiency of slotted ALOHA is

A. 0.18 B. 0.36
C. 1 D. None

269. The expected number of transmission of a slotted ALOHA with offered load G is

A. e^{G} B. e^{-G}
C. e^{2G} D. None

270. If the offered load G is 1 then probability of success in slotted ALOHA is

A. 26% B. 37%
C. 74% D. None

271. If the offered load is G is 1 then number slots involve in collision in slotted ALOHA are

A. 26% B. 37%
C. 74% D. None

272. Find odd one out

A. ALOHA B. Slotted ALOHA
C. Bit-map D. CSMA/CD

273. Binary countdown algorithm is used in

A. IEEE 802.3 B. CSMA/CD
C. IEEE 802.4 D. Slotted ALOHA

274. IEEE 802.3 is

A. 1-persistent B. p-persistent
C. Persistent D. None

275. Thin Ethernet is

A. 10Base2 B. 10Base5
C. 10BaseF D. None

276. In IEEE 802.3 address field to be made as ___ for broadcast.

A. All 0's B. All 1's
C. Nulls D. None

277. In IEEE 802.3 minimum frame size should have

A. Frame time = propogation delay
B. Frame time= round trip propagation delay
C. Frame time can be any thing
D. None

278. For a 10Mbps IEEE 802.3 LAN with maximum of length 2500 m the minimum allowed frame is

A. 51.2 micro second B. 100 micro seconds
C. 120 micro second D. None

279. For a 1Gbps IEEE 802.3 LAN with maximum length of 250 m should have frame size of atleast

A. 6400 bytes B. 640 bytes
C. 64 bytes D. None

280. Priority is supported in

A. IEEE 802.3 B. IEEE 802.4
C. IEEE 802.5 D. Both B & C

281. Beacon frame is used in

A. IEEE 802.3 B. IEEE 802.4
C. IEEE 802.5 D. Both B & C

282. ____collision free.

A. IEEE 802.3 B. IEEE 802.4
C. IEEE 802.5 D. Both B & C
E. None

283. Acknowledgment is available in

A. IEEE 802.3 B. IEEE 802.4
C. IEEE 802.5 D. Both B & C
E. None

284. Monitor

A. IEEE 802.3 B. IEEE 802.4
C. IEEE 802.5 D. Both B & C
E. None

285. Padding

A. IEEE 802.3 B. IEEE 802.4
C. IEEE 802.5 D. Both B & C
E. None

286. Monitor

A. Generates token
B. Takes care orphan frames
C. Checks breaks in ring
D. All

287. Ethernet data rates ___ Mbps

A. 1 B. 10
C. 100 D. All above
E. None

288. Smallest possible Ethernet packet size (assume addresses are 6 bytes long) is

A. 72 bytes B. 1526
C. 1500 D. None

289. Largest possible Ethernet packet size (assume addresses are 6 bytes long) is

A. 72 bytes B. 1526
C. 1500 D. None

290. Largest possible padding bytes in Ethernet (assume addresses are 6 bytes long) is
A. 26 B. 56
C. 72 D. 46
E. None

291. CRC polynomial used in Ethernet is
A. CRC-12 B. CRC-16
C. CRC-32 D None

292. Imagine that the length of a 10Base5 LAN cable length is 2500m and it is working at 1Gbps. In worst case how many padding bits are needed? (assume addresses are 6 bytes long)
A. 46 B. 614
C. 6474 D. None

293. Fast Ethernet is
A. 100base-T4 B. STP (two pairs)
C. Two pairs of UTP D. 100Base-XF
E. All

294. Fast Ethernet differs with simple Ethernet in
A. Speed (data rate) B. Collision domain
C. Both A & B D. None

295. Multiple access of stations
A. IEEE 802.3 B. IEEE 802.4
C. IEEE 802.5 D. None

296. Token ring supports __ data rates.
A. 4Mbps B. 16Mbps
C. B & C D. None

297. CDDI is
A. Copper version of FDDI
B. Other version of Ethernet
C. Fast Ethernet
D. None

298. A seven-bit code can generate ___ possible characters.
A. 7 B. 128
C. 256 D. None

299. Each asynchronous character is preceeded by a __ bit which has the opposite polarity to the idel condition.
A. Start B. Stop
C. Neutral D. None

300. Statistical mutiplexers may also be classed as
A. TDM B. Concentrators
C. FDM D. None

301. All communication satellites are
A. Geosynchronous
B. Geostationery
C. at about 36000Km orbit
D. Both A & C
E. Both B & C

302. ___ is used in high speed modems.
A. AM B. FM
C. QAM D. PSK

303. At physical level of IEEE 802.3 ___ code is used.
A. Binary B. Bipolar binary
C. NRZ D. Manchester

304. Effectiveness of Error detection codes are usually measured in ___distance.
A. Euclidian B. Chessboard
C. Hamming D. None

305. Sliding window protocols are ___ protocol.
A. Flow control B. Congestion
C. MAC D. None

306. In pure ALOHA the frames are of ___ length.
A. Fixed B. Variable
C. Protocol dependent D. None

307. The maximum number of repeaters that can be connected for the largest configuration of Ethernet is
A. 1 B. 2
C. 3 D. 4
E. None

308. Using repeaters the maximum cable length in Ethernet can be ___ Km.
A. 1 B. 2.5
C. 1.2 D. 10

309. X.25 normally assumes the use of ___ as the physical layer standard.
A. X.25 B. X.21
C. PPP D. SLIP

310. In X.25, packet transferred across DTE/DCE interface begins with a __ header.
A. 26 bytes B. 3 bytes
C. 4 bytes IP address D. None

311. For the 2.048 Mbps the ISDN channel structure will have___no. of B channels and one 64Kbps D channel.
A. 32 B. 24
C. 30 D. None

312. The light accepting capability of an optical fiber is measured in
A. Reflectance
B. Refraction IFOV
C. Numerical aperture

313. Which of the following is not guided transmission line?

A. Pair of wires B. Waveguides
C. Laser beam D. Fiber optic channel

314. In twisted pair cables the mode of communication is
A. FDM B. TDM
C. CDMA D. TDMA

315. Which of the following is a balanced interface?
A. RS-232C B. RS-449
C. RS-423 D. RS-422

316. Data networks are
A. Simplex B. Half duplex
C. Full duplex D. None

317. Double bit errors can be identified through
A. LRC B. VRC
C. CRC D. None

318. Which of the following requires higher bandwidth?
A. Speech B. Digital speech
C. CD quality audio D. None

319. Which of the following requires higher bandwidth?
A. Speech B. Facsimile
C. Video D. Music

320. An example for base band communication is
A. Radio B. TV
C. Telephone D. Satellite links

321. A twisted wire can not be suitable for
A. Analog voice B. Digital voice
C. Analog video D. None

322. When an analog signal is bandlimited to 5KHz and is to be transmitted using a 8-bit PCM encoder, the minimum data rate required is
A. 8Kbps B. 16Kbps
C. 32Kbps D. 64Kbps

323. The downlink and uplink channels of a satellite are separated in
A. Time B. Space
C. Frequency D. None

324. In digitized voice with 8000 samples/sec ___ frequencies are lossed.
A. Below 20Hz B. Above 4KHz
C. Above 2KHz D. None

325. CD quality audio bandwidth
A. 4KHz B. 10KHz
C. 22.05KHz D. None

326. CD quality audio (stereo) bandwidth
A. 64Kbps B. 705.6Kbps
C. 1.411Mbps D. None

327. Does a T1 carrier adequate to carry a single CD quality audio channels?
A. Yes B. No

328. MIDI
A. Supports 127 instruments
B. Reduces the bandwidth requirement to play sound on an instrument
C. Employs codes
D. All

329. In JPEC ___ transform is used.
A. DCT B. FFT
C. Fourier D. Hadamand

330. To route packets to destination LAN ___ addresses are needed.
A. IP addresses B. Subnet
C. Data link addresses D. Network card addresses

331. Physical address
A. Subnet address
B. IP address
C. Network card address
D. Socket address

332. ____ address is more related to a process.
A. Subnet B. IP
C. Network card D. Socket

333. Jumbograms
A. Data link layer B. IPV4
C. IPV6 D. None

334. IPV4 suffers
A. Wastes addresses
B. No encryption & authentication is provided
C. Unavailability of resource reservation for real time audio & video
D. All

335. IPV6 supports
A. Larger address space
B. More security
C. Resource reservation freedom
D. Better header format
E. All

336. IPV6 supports
A. Unicast B. Anycast
C. Multicast D. All

337. Node identifier in IPV6 is
A. 48 bits B. 8 bits
C. 16 bits D. 32 bits

338. Subnet address in IPV6 is
A. 48 bits B. 8 bits
C. 16 bits D. 32 bits

339. Internet provider address in actual address part of IPV4 occupies
A. 48 bits B. 8 bits
C. 16 bits D. 32 bits
E. None

340. Loopback addresses in IPV6
A. Not supported
B. All 0's followed by 1 at the end
C. Starts with 127
D. None

341. Find odd one out
A. INTERNIC B. RIPNIC
C. NIC D. APNIC

342. Reserved addresses in IPV6 starts with
A. 1 B. 0
C. 00000000 D. None

343. In IPV6, link local addresses starts with
A. 00000000 B. 11111111
C. 11111110

344. In IPV6 hop limit is used. It's equavalent in IPV4 is
A. TTL filed B. Priority
C. Flow label D. None

345. Does header checksum is needed in IPV6?
A. Yes B. No

346. Record rout option is
A. Supported in IPV4 B. Supported in IPV6
C. Both A & C D. None

347. Which of the following is a necessary part of IPV6 datagram?
A. Base header
B. Extension header
C. Data packet from the upper layer
D. Both A & C

348. The ___ field in the base header of IPV6 restricts the lifetime of a datagram.
A. Version B. Priority
C. Next header D. Hop limit

349. In IPV6 When a datagram needs to be discarded in a congested network the decision is based on the __ field of base header.
A. Version B. Priority
C. Next header D. Hop limit

350. In IPV6, the___ field in the base header and sender IP address combine to indicate a unique path identifier for a specific flow of data.
A. Version B. Priority
C. Next header D. Hop limit

351. The source address in the base header always contains the address of the __.
A. Last router B. Next router
C. Original sender D. Any of the above

352. For IPV6, for a maximum number of hops, set the hop limit field to decimal___.
A. 16 B. 15
C. 42 D. 0

353. For IPV6 for time sensitive data, assign the priority field a value of decimal ___.
A. 0 B. 7
C. 8 to 15 D. 16

354. A datagram in IPV6 with a priority of ___ will be discarded before a datagram with a priority of 12.
A. 11 B. 7
C. 0 D. Any of the above

355. A 6000-byte packet needs to routed through an Ethernet LAN, what extension header must be used in IPV6?
A. Source routing B. Fragmentation
C. Authentication D. Destination option

356. An IP datagram in IPV6 is 80000bytes. What extension header must be used?
A. Hop-by-hop B. Fragmentation
C. Authentication D. Destination option

357. The maximum size for an IPV6 datagram is __ bytes.
A. 65535 B. 65575
C. 2^{32} D. $2^{32}+40$

358. When the hop count field reaches zero and the destination has not been reached, a __ error message is sent.
A. Destination unreachable
B. Time exceeded
C. Parameter problem
D. Packet too-big

359. When all fragments of a message have not been received with the designated amount of time, a __ error message is sent.
A. Destination unreachable
B. Time exceeded
C. Parameter problem
D. Packet too-big

360. The __ field in the socket structure is usually set to 0 for a process using the services of TCP or UDP.
A. Family B. Type

C. Protocol D. Local socket address

361. The field ___ in the socket structure defines the protocol group.
A. Family B. Type
C. Protocol D. Local socket address

362. The ___ socket I sued with a connectionless protocol.
A. Stream B. Datagram
C. Raw D. Remote

363. The ___ socket is sued with a protocol that directly uses the services of IP.
A Stream B. datagram
C. Raw D. Remote

364. On an IBM mainframe computer the starting address in memory for the IP address 7.8.9.11 contains
A. 00000111 B. 00001000
C. 00001001 D. 00001011

365. On a PC the starting address in memory for the IP address 7.8.9.11 contains
A. 00000111 B. 00001000
C. 00001001 D. 00001011

366. The___function provides information about the host.
A. Hostent B. Hostname
C. Gethostbyname D. Getnameofhost

367. A connectionless process issues the ___ system call to receive, from the incoming queue, the datagrams sent by a remote process.
A. Listen B. Receive
C. Bind D. Recvfrom

368. The ___ system call converts a socket into a passive socket.
A. Convert B. Listen
C. Socket D. Bind

369. In TCP/IP suite port numbers are
A. 16 bit numbers
B. 0 to 65535
C. Used to map onto running processes
D. All

370. In TCP/IP well known ports are the one's at which popular services available
A. 0-1023 B. Both A & B
C. None

371. Telnet uses port
A. 23 B. 11
C. 21 D. 19

372. ___ is responsible for delivery of the message to the appropriate application program.
A. TCP B. DLL
C. PL D. Computer network
E. Network layer

373. TCP
A. Connection oriented
B. Reliable
C. Adds the above features to services of IP
D. Connection establishment, managment
E. All

374. Find odd one out
A. IP delivers packets to destination computer
B. IP requires IP addresses
C. TCP delivers packets to respective process on the destination machine
D. IP packets contains socket address in their destination fields

375. HTTP service
A. 80 B. 20
C. 21 D. 23

376. Socket address
A. IP address B. Port number
C. Both A & B D. None

377. TCP services
A. Stream data service B. Full duplex service
C. Reliable service D. None

378. Find odd one out
A. Diskless machine B. Bootp
C. Rarp D. 67
E. None

379. Find odd one out in terms of padding
A. IP B. TCP
C. Ethernet D. None

380. Sliding window protocol is used in
A. X.25 B. DLL
C. TCP flow control D. All

381. Find odd one out
A. Urgent data
B. Out of band data
C. URG bit
D. Urgent data is dealt seperately
E. All

382. The ___ address uniquely identifies a running program
A. IP address B. Host
C. NIC C. Socket

383. Find odd one out

A. IBM mainframe
B. Apple Macintosh (Motorla based)
C. Sun sparc
D. IBM PC

384. What is the maximum no. of no-operation options in one 32-bit word?
A. 1 B. 2
C. 3 D. 4

385. In ___, data is sent or processes at a very inefficient rate such as one byte at a time.
A. Nagles syndrome
B. Silly window syndrome
C. Sliding window syndrome
D. Delayed acknowledgement

386. Which option define the maximum number of bytes in a TCP segment?
A. Maximum segment size
B. Window scale factor
C. Timestamp
D. No operation

387. Any mobile agent can get service from LAN by presenting its identification. (Y/N)

388. Monitor cannot
A. Generate token
B. Handle orphan frames
C. Allow others to become monitor while it is active
D. Support priority
E. Introduce delays

389 Which of the following is connecting device in DLL?
A. Transponder B. Transciever
C. Codec D. Bridge
E. Router

390. Find odd man with respect to calculating checksum.
A. UDP B. TCP
C. IP D. Ethernet

391. The worst case padding required for a 10Mbps Ethernet protocol with round trip propagation delay 51.2 micro second is _____ (Assume 6 byte addresses).

392. Find odd one out
A. ICMP B. RARP
C. BOOTP D. ARP
E. DHCP

393. Find odd one out
A. TCP is reliable service
B. IP provides only best-effort connection less packet transfer
C. TCP is connection oriented service
D. Gateways may discard packets when they encounter congestion
E. IP guarantees for QoS

394. Padding is not needed in
A. Ethernet data frame B. IP header
C. UDP header D. Token ring

395. DF bit is set then the IP packet then it may be coming from _____ and traversing to _____

396. A packet arrives at a router and is forwarded to a network with MTU value of 576 bytes. The IP header of the packet is 20 bytes and data is 1484 bytes. What should be the fragment offset of second fragment if the first fragment is at its maximum possible size.

397. Find odd one out
A. Source quench B. Parameter problem
C. Time exceeded D. Timestamp reply
E. NAK

398. Pushed data and urgency data are same. (Y/N)

399. The default maximum segment size in TCP is
A. 536 bytes B. 65536
C. 576 D. 65495

400. Find odd one out
A. In TCP segment urgent pointer is set
B. The value of the urgent pointer field added to get the last byte of urgent data
C. The start of the urgent data is not defined explicitly
D. Urgent data is handled by IP
E. Urgent pointer is 16 bit field

401. If the URG bit is set and sequence number is 2048 and urgent pointer value is 0000000000000101 then total size of the urgent data is

402. A token ring working at 4Mbps with 20 stations each separated by 100m with delays b=2.5bits and signal propagation speed is 200000000 m/sec. What will be the efficiency of the ring if frame size 400 bits. Also calculate the efficiency assuming b=1bit. Calculate the same for a ring working at 16Mbps.

403. Suppose the IP header consists of (11111111 11111111 11111111 00000000 11110000 11110000 11000000 11000000) then the Internet checksum is _____.

404. Socket is
A. 48 bit address
B. 32 bit address
C. 16 bit address
D. A file
E. A stream

405. Efficiency of the token ring is (find odd man out)
- A. More if heavily loaded
- B. Very less load is less
- C. Less because of walk time
- D. More because of delays at interface units
- E. Less because of contention during ring initialisation

406. Padding is needed in Ethernet as _____

407. Entire message cannot be sent as a single communication data unit as
- A. One may monopolize channel
- B. Utilisation may become poor
- C. Buffer requirements increases
- D. All
- E. None

408. If we happened to manufacture a noise less channel, do we still need link control protocols? (Y/N)

409. Why checksum field in Ethernet is 4 bytes?

410. Required S/N value of the channel with 4Khz bandwidth which carry digital voice in real time is _____

411. Can we go on send frames if the channel is ideal channel such as super conductor? (Y/N)

412. Data Link layer responsibility does not include
- A. Framing & de-framing
- B. Error correction & detection
- C. Delivery of data from source machine to destination machine
- D. Piggybacking
- E. Flow control

413. Efficiency of a stop-and-wait protocol is at least 0.5 if
- A. Frame time is Γ (propagation delay)
- B. Frame time is 2Γ (round trip propagation delay)
- C. Frame time is at least 2Γ (round trip propagation delay) & error rate is 1 in 2ΓC, where C is data rate
- D. Never
- E. None

414. CD quality audio bandwidth
- A. 4Khz
- B. 8Khz
- C. 23KHz
- d. 300Khz
- E. None

415. Layer in TCP/IP which synchronises after communication failures is
- A. Data link
- B. Transport
- C. Session
- D. Network

416. What is wrong about a station in CSMA/CD (Ethernet LAN)
- A. It gets blocked
- B. It can sense channel
- C. It can identify collision
- D. It supports exigency (priority) service
- E. Calculates checksum

417. What is true about IEEE 802.3 LAN?
- A. Stations concludes a frame is sent if it did not see collision before transmitting last bit of the frame
- B. A station sends more than one frame at a time
- C. 6 byte addresses used as IPV6 addresses
- D. Acknowledgements are used
- E. Amount of padding increases with reduce in network spread

418. Min frame size for Ethernet LAN with 250m spread and 51.2micro sec (round trip propagation delay) and working at 10Mbps is
- A. 64 bytes
- B. 6.5 bytes
- C. 640 bytes
- D. 6400 bytes
- E. None

419. Hamming distance for the code book with code words 01100001, 11000001, 11110111, and 00001011 is
- A. 2
- B. 3
- C. 4
- D. 5

420. Stop-and-Wait is not efficient on optical fiber lines as
- A. Propagation delay is very less
- B. Propagation delay is very large
- C. Data rate is very high
- D. Both A & C
- D. Both B & C

421. Frames sent in a Go back n protocol which employ 3 bit sequence numbers are 4,5,6,7,0,1,2 then next frames that will be sent if frame 7 gets spoiled is
- A. 3,4,5,6,7,0,1
- B. 0,1,2,3,4,5,6
- C. 7,0,1,2,3,4,5
- D. None

422. Number of bits in a frame in T1 carrier are
- A. 1
- B. 25
- C. 193
- D. None

423. End of the frame is indicated by
- A. By sending special bit sequence
- B. By sending special character sequences
- C. By violating physical layer protocols by not having any transition in the middle of clock cycle
- D. All

424. What bandwidth is required to multiplex (TDM) 24 CD quality audio channels. Assume 12 bit samples are employed.

ANSWER KEY

1. A	**2.** B	**3.** A	**4.** C
5. C	**6.** C	**7.** B	**8.** B
9. B	**10.** C	**11.** C	**12.** C
13. D	**14.** B	**15.** B	**16.** E
17. A	**18.** D	**19.** C	**20.** E
21. B,D	**22.** B	**23.** B,C	**24.** B
25. A	**26.** A	**27.** D	**28.** B
29. A	**30.** A	**31.** C	**32.** A
33. B	**34.** B	**35.** A	**36.** B
37. B	**38.** C	**39.** D	**40.** A
41. A	**42.** D	**43.** B	**44.** A
45. B	**46.** C	**47.** C	**48.** B
49. A	**50.** B	**51.** A	**52.** B
53. A	**54.** D	**55.** B	**56.** C
57. B	**58.** A	**59.** A	**60.** A
61. D	**62.** A	**63.** B	**64.** D
65. C	**66.** D	**67.** A	**68.** B
69. A	**70.** B	**71.** A	**72.** D
73. D	**74.** B	**75.** D	**76.** B,C
77. B	**78.** A	**79.** D	**80.** D
81. C	**82.** B	**83.** B	**84.** A
85. B	**86.** D	**87.** B	**88.** C
89. C	**90.** B	**91.** A	**92.** D
93. B	**94.** D	**95.** C	**96.** D
97. C	**98.** C	**99.** A	**100.** C
101. C	**102.** C	**103.** B	**104.** D
105. B	**106.** A	**107.** B	**108.** C
109. D	**110.** B	**111.** D	**112.** B
113. B	**114.** D	**115.** C	**116.** D
117. B	**118.** B	**119.** A	**120.** B
121. C	**122.** B	**123.** D	**124.** C
125. A	**126.** D	**127.** B	
128. D(Peer to Peer)		**129.** B	**130.** D
131. D	**132.** B	**133.** C	**134.** C
135. C	**136.** D	**137.** C	**138.** C
139. A	**140.** D	**141.** B	**142.** C
143. C	**144.** B	**145.** B	**146.** D
147. D	**148.** C	**149.** B	**150.** D
151. C	**152.** B	**153.** C	**154.** A
155. D	**156.** B	**157.** C	**158.** D
159. B	**160.** B	**161.** B	**162.** C
163. D	**164.** C	**165.** B	**166.** D
167. C	**168.** C	**169.** N.A	**170.** A

171. D	**172.** C	**173.** A	**174.** C
175. A	**176.** B	**177.** A	**178.** B
179. B	**180.** D	**181.** A	**182.** A
183. B	**184.** C	**185.** E	**186.** D
187. A	**188.** A	**189.** B	**190.** B
191. B	**192.** C	**193.** E	**194.** E
195. A	**196.** All	**197.** E	**198.** C
199. A	**200.** B	**201.** E	**202.** D
203. C	**204.** D	**205.** C	**206.** A
207. A	**208.** D	**209.** E	**210.** E
211. D	**212.** C	**213.** A	**214.** A
215. A	**216.** B	**217.** C	**218.** C
219. E	**220.** B	**221.** C	**222.** C
223. B	**224.** D	**225.** B	**226.** A
227. C	**228.** B	**229.** B	**230.** A
231. C	**232.** C	**233.** B	**234.** C
235. C	**236.** A	**237.** A,C	**238.** B
239. D	**240.** B	**241.** A	**242.** E
243. A	**244.** A	**245.** A	**246.** A
247. C	**248.** C	**249.** B	**250.** C
251. A	**252.** C	**253.** D	**254.** D
255. 010011100011110101			**256.** B
257. B	**258.** C	**259.** D	**260.** C
261. C	**262.** A	**263.** D	
264. 160bits(refer Page 7.17)			
265. 777bits(refer Page 7.17)			
266. 1/501	**267.** A	**268.** B	
269. A	**270.** B	**271.** A	**272.** C
273. C	**274.** A	**275.** A	**276.** B
277. B	**278.** A	**279.** B	**280.** D
281. C	**282.** D	**283.** C	**284.** C
285. A	**286.** D	**287.** D	**288.** A
289. B	**290.** D	**291.** C	**292.** B
293. E	**294.** C	**295.** A	**296.** C
297. A	**298.** B	**299.** A	**300.** B
301. E	**302.** C	**303.** D	**304.** C
305. A	**306.** A	**307.** A	**308.** B
309. B	**310.** B	**311.** C	**312.** C
313. B	**314.** D	**315.** D	**316.** C
317. C	**318.** C	**319.** C	**320.** C
321. C	**322.** E	**323.** C	**324.** B
325. C	**326.** C	**327.** B	**328.** A
329. A	**330.** B	**331.** C	**332.** D
333. C	**334.** D	**335.** E	**336.** D
337. A	**338.** D	**339.** C	**340.** B
341. C	**342.** C	**343.** C	**344.** A

345. B **346.** A **347.** C **348.** D
349. B **350.** C **351.** C **352.** C
353. C **354.** D **355.** B **356.** A
357. B **358.** A **359.** B **360.** C
361. A **362.** B **363.** C **364.** A
365. D **366.** C **367.** B **368.** B
369. D **370.** C **371.** A **372.** A
373. A **374.** C **375.** A **376.** C
377. C **378.** E **379.** C **380.** D
381. D **382.** D **383.** D **384.** D
385. B **386.** A **387.** N **388.** E
389. D **390.** D(Uses CRC) **391.** 38
392. A **393.** B **394.** D
395. Bootp Server, Bootp Client **396.** 472
397. B **398.** N **399.** A **400.** D
401. 5 **402.** Refer 7.3.4.4.2
403. Refer page 7.38 **404.** E **405.** A
406. Collision is required to be seen by the source
407. A **408.** Y **409.** Refer 7.3.4.1
410. 43dB **411.** N **412.** C **413.** B
414. C **415.** C **416.** A **417.** A
418. Refer Page 7.19 **419.** A **420.** D
421. C **422.** C **423.** C
424. 24x23000x2x12bits

7.8 Matching Examples

For Exercises 1-6, match the word or acronym with the definition or the appropriate blank.

A. LAN B. WAN
C. Gateway D. Bus topology
E. Ethernet F. Internet

1. The Internet is a _____
2. The industry standard for LANs
3. A node that handles communication between its LAN and other networks
4. A network that connects other networks.
5. Star technology is a _____ configuration.
6. Ethernet uses _____.

For Exercises 7 - 15, match the word or acronym with the definition or the appropriate blank.

A. DLS B. TCP/IP
C. UDP D. IP
E. TCP F. Boradband

7. _____ and voice communication can use the same phone line.
8. DLS and cable modems are_____ connections.
9. An Internet connection made using a digital signal on regular phone lines.
10. Network technologies that generally provide data transfer speeds greater than 128I bps.
11. The network protocol that breaks messages into packets, reassembles the packets at the destination, and takes care of errors.
12. The suite of protocols and programs that supports low-level network communication.
13. An alternative to TCP that achieves higher transmission speeds.
14. Software that deals with the routing of packets.
15. _____ has more reliability than UDP.

For Exercises 15 -20, match the protocol or standard with what it specifies or defines.

A. SMTP B. FTP
C. Telnet D. HTTP
E. MIME type

16. Transfer of electronic mail.
17. Log into a remote computer system.
18. Transfer files to and from another computer.
19. Format of email attachments.
20. Exchange of World Wide Web documents.

For Exercises 21-25, mark the answers true and false as follows:

A. True B. False

21. A port is a numeric designation that corresponds to a particular high-level protocol.
22. A firewall protects a local-area network from physical damage.
23. Each company can establish its own access control policy.
24. Some top-level domains are based on the country in which the registering organisation is based.
25. Two organisations cannot have the same name for a computer.

ANSWER KEY

1. B **2.** E **3.** C **4.** B
5. A **6.** D **7.** A **8.** F
9. A **10.** F **11.** E **12.** B
13. C **14.** D **15.** E **16.** A
17. C **18.** B **19.** E **20.** D
21. A **22.** B **23.** A **24.** A
25. B

7.9 True or False Questions

1. If stored video is streamed directly from a Web server to a media player, then TCP is used as the underlying transport protocol.
2. In an RTP/UDP streaming scenario, UDP segments are carried inside RTP packets.
3. When using RTP, it is possible for a sender to change encoding in the middle of a session.
4. RTP does not provide mechanisms to ensure timely delivery of data.
5. For a given digital modulation scheme, the higher the Signal-to-Noise Ratio (SNR), the higher the Bit Error Rate (BER).
6. The IEEE 802.11 wireless LAN uses CSMA/CD as the random access protocol.
7. An 802.11 frame header contains 4 address fields, each of which can hold a 6-byte MAC address.
8. Consider a mobile node in the Internet. In case of direct routing, a correspondent node (sender) needs to first query the home agent of the mobile node to learn the mobile node's care-of-address (COA).
9. Assume Alice and Bob want to communicate securely. Alice can use her private key to sign a message.
10. Alice can use her public key to encrypt a message such that only Bob can decrypt it.
11. An application that runs on a given host needs to use unique source port numbers for each of its TCP connections.
12. Congestion can be overcome if we have an unlimited amount of storage (for buffering packets) in routers.
13. When using RTP in conjuction with RTCP, it is possible for a sender to change encoding in the middle of a session.
14. The IP header checksum is recomputed at every hop.
15. It is possible that a router implements several types of link layers.
16. A router decides on a route for an IP packet based on its source and destination address.
17. The TTL (time to live) field, which is decremented at every hop in the network to avoid packet forwarding loops, is part of the TCP header.
18. The IEEE 802.11 wireless LAN uses CSMA/CD (Collusion Detection) as the random access protocol.
19. Alice can use her public key to encrypt a message such that only Bob can decrypt it.
20. In public key encryption, the public keys of Alice and Bob must be unknown to Trudy the intruder.
21. A network layer packet is encapsulated within a link layer packet.
22. In IPV6, when an IP layer packet is larger than what the link is designed to carry, the packet is fragmented. These fragments are reassembled at the destination host IP layer.
23. Both hubs and switches use store-and-forward transmission.
24. A client program is a program running on one end system that requests a service from a server program running on another end system.
25. All the routers in an autonomous system use the same forwarding table.
26. UDP is the preferred transport layer protocol for delivering e-mail.
27. A Web cache is both a client and a server.
28. Assume that a file of size F is to be distributed to N clients in client-server architecture. If the upload rate of the server's access link is us, then the time to distribute the file to N clients is equal to $NF = us$.
29. TCP socket is identified by a four-tuple: source IP, destination IP, source port number and destination port number.
30. If two UDP packets have different source IP addresses and source port numbers, then the two packets will be directed to different destination processes.
31. TCP does not compute a SampleRTT for segments that have been retransmitted.
32. In TCP, 3 duplicate ACKs are interpreted as a sign of network congestion.
33. In Network Address Translation (NAT), the source IP address field of a datagram leaving the local network is equal to the IP address of the local host that originally created it.
34. Real-Time Transport Protocol (RTP) packets are interpreted by routers for improved multimedia Quality of Service (QoS).
35. Real Time Streaming Protocol (RTSP) defines how audio/video is encapsulated for streaming over network.
36. Reliable transport services may be needed from the transport layer even if the data link transport is reliable along the end-to-end path.
37. In mobility via direct routing, communication from the correspondent host to the mobile host goes through the home agent of the mobile.
38. In mobility via direct routing, assume that the mobile initially visits a foreign network with foreign agent $FA0$ and then keeps moving across other foreign

networks. Wherever the mobile goes, the data from the correspondent is always forwarded to the mobile through FA0.

39. The IEEE 802.11 wireless LAN uses CSMA/CD as the random access protocol.

40. A laptop computer may implement several types of link layers.

41. CRC can both correct and detect errors.

42. 2-d parity can detect all 2-bit errors and correct all 1-bit errors.

43. 4B/5B encoding solves the problem of long sequences of zeroes but still has issues with long sequences of 1s.

Answer: 4B/5B uses NRZI to get over the issue of a long series of 1s.

44. When bandwidth is plenty and errors are rare, error checking may actually add unnecessary overhead.

45. The transmission rate of a link is measured in bits/second.

46. End systems, packet switches and other pieces of the Internet run protocols that control the sending and receiving of information within the Internet.

47. Tier-1 ISPs are also known as Internet backbone networks.

48. The total nodal delay for a router is a sum of processing delay, queuing delay, transmission delay, and propagation delay.

49. A transport-layer packet is called a segment. A network-layer packet is called a datagram. A link-layer packet is called a frame.

50. The application architecture dictates how the application is structured over various end systems. There are two predominant architectural styles used in practice: client-server architecture and P2P architecture.

52. A process sends/receives messages to/from the network through a software interface called a socket.

53. A web page consists of objects, which are addressable by URLs. These are simply files such as an HTML file, a JPEG image, a Java applet or a video clip.

54. Because an HTTP server maintains no information about the clients, an HTTP server is said to be stateless.

55. Because FTP uses a separate control connection, FTP is said to send its control information out-of-band.

56. A reliable data transfer protocol may send multiple packets without waiting for acknowledgements, rather than operating in a stop-and-wait manner. This technique is called pipelining.

57. In P2P file sharing (e.g. Gnutella), a TCP connection between two peers X and Y is visualised as an edge between X and Y in the overlay network graph.

58. The three sources of signal impairments on a physical link are i)attenuation, ii)distortion, and iii)noise.

59. For a given network throughput, real-time networked multimedia applications are delay and jitter sensitive. They are, however, loss tolerant.

60. For networked multimedia applications, client-side buffering and playout delay can compensate for network delay and delay-jitter.

61. A host can dynamically get an IP address from a server (IP pool) using the DHCP protocol.

62. When a host sends a packet with destination IP address 255.255.255.255, the message is delivered to all hosts in the same subnet.

63. A switch/hub/bridge does not implement Internet protocol stack layers above the link layer.

64. The IEEE 802.11 wireless LAN uses CSMA/CA as the random access protocol.

ANSWER KEY

1. T	**2.** F	**3.** T	**4.** T
5. F	**6.** F	**7.** T	**8.** T
9. T	**10.** F	**11.** F	**12.** F
13. T	**14.** T	**15.** T	**16.** F
17. F	**18.** F	**19.** F	**20.** F
21. T	**22.** F	**23.** F	**24.** T
25. F	**26.** F	**27.** T	**28.** F
29. T	**30.** F	**31.** T	**32.** F
33. F	**34.** F	**35.** F	**36.** T
37. F	**38.** T	**39.** F	**40.** T
41. F	**42.** T	**43.** F	**44.** T
45. T	**46.** T	**47.** Y	**48.** Y
49. T	**50.** T	**51.** T	**52.** T
53. T	**54.** T	**55.** T	**56.** T
57. T	**58.** T	**59.** T	**60.** T
61. T	**62.** T	**63.** T	**64.** T

Previous Years' GATE Questions

1. The transport layer protocols that are used for real time multimedia, file transfer, DNS and email are

(GATE 2013)

A. TCP, UDP, UDP, and TCP
B. UDP,TCP, TCP and UDP
C. UDP, TCP, UDP and TCP
D. TCP, UDP, TCP and UDP

2. Using public key encryption, X adds a digital signature σ to a message M, encrypts <M, σ>, and sends it to Y, where it is decrypted. Which one the following sequence of keys are used for the operations? **(GATE 2013)**

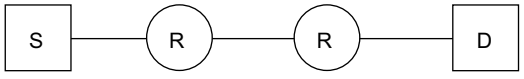
A. Encryption: X's private key followed by Y's private key; Decryption: X's public key followed Y's public key

B. Encryption: X's private key followed by Y's public key; Decryption: X's public key followed Y's private key

C. Encryption: X's public key followed by Y's private key; Decryption: Y's public key followed X's private key

D. Encryption: X's private key followed by Y's public key; Decryption: Y's private key followed Y's public key

3. Assume that source S and destination D are connected through two intermediate routers labeled R. Determine how many times each packet has to visit the network layer and data link layer during a transmission from S to D. **(GATE 2013)**



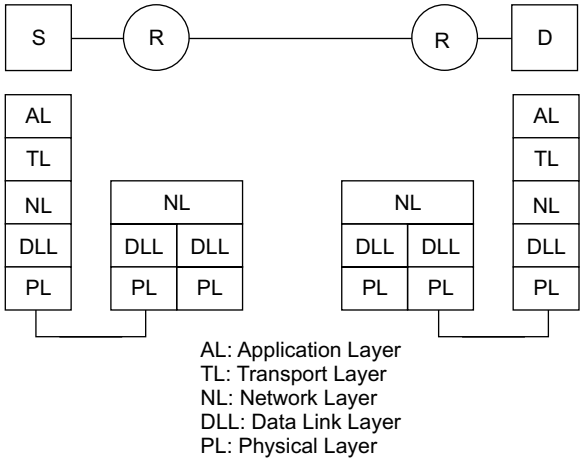
A. Network Layer-4 times and Data link layer-4 times

B. Network Layer-4 times and Data link layer-3 times

C. Network Layer-4 times and Data link layer-6 times

D. Network Layer-2 times and Data link layer-6 times

Explanation: See the following diagram which explains the data flow in the given network.



4. Determine the maximum length of the cable (in Km) for transmitting data at a rate of 500Mbps in an Ethernet LAN with frames of size 10,000bits. Assume the signal speed in the cable to be 2,00,000Km/s. **(GATE 2013)**

A. 1 B. 2

C. 2.5 D. 5

Explanation: In Ethernet, minimum frame time is two times of propagation delay. Consider x be the length of the cable. Frame time=10000/500x106. Therefore,

$10000/500x10^6=2*x/200000$

$1/50000=x/100000$

Therefore, x= 100000/50000=2KM

5. In an PIv4 datagram, the M bit is 0, the value of HLEN is 10, the value of total length is 400 and the fragment offset value is 300. The position of the datagram, the sequence numbers of the first and the last bytes of the payload, respectively are **(GATE 2013)**

A. Last segment, 2400, and 2789

B. First segment, 2400 and 2759

C. Last fragment, 2400 and 2759

D. Middle fragment, 300 and 689

6. The protocol data unit (PDU) for the application layer in the Internet stack is **(GATE 2012)**

A. Segment B. Datagram

C. Message D. Frame

7. Which of the following transport layer protocols is used to support electronic mail? **(GATE 2012)**

A. SMTP B. IP

C. TCP D. UDP

8. In the IPv4 addressing format, the number of networks allowed under Class C addresses is **(GATE 2012)**

A. 2^{14} B. 2^7

C. 2^{21} D. 2^{24}

9. An Internet Service Provider (ISP) has the following chunk of CIDR-based IP addresses available with it: 245.248.128.0/20. The ISP wants to give half of this chunk of addresses to Organisation A, and a quarter to Organisation B, while retaining the remaining with itself. Which of the following is a valid allocation of addresses to A and B? **(GATE 2012)**

A. 245.248.136.0/21 and 245.248.128.0/22

B. 245.248.128.0/21 and 245.248.128.0/22

C. 245.248.132.0/22 and 245.248.132.0/21

D. 245.248.136.0/24 and 245.248.132.0/21

Explanation: Given IP address, we can understand 20 bits for network ID and 12 for host ID. Which indicates the network can have 2^{12} hosts, half of it means 2^{11} for A and 2^{10} for B.

From given IP address, probable address ranges for A and B

Option 1:

Network A

245.248.1000 0000 0000 0000	245.248.128.0
245.248.1000 0111 1111 1111	245.248.135.255

Network B

245.248.1000 1000 0000 0000	245.248.136.0
245.248.1000 1011 1111 1111	245.248.139.255

Option 2:

Network B

245.248.1000 0000 0000 0000	245.248.120.0
245.248.1000 0011 1111 1111	245.248.131.255

Network A

245.248.1000 1000 0000 0000	245.248.136.0
245.248.1000 1111 1111 1111	245.248.143.255

So, option A is valid.

10. Consider a source computer (S) transmitting a file of size 10^6 bits to a destination computer (D) over a network of two routers (R1 and R2) and three links (L1, L2, and L3). L1 connects S to R1; L2 connects R1 to R2; and L3 connects R2 to D. Let each link be of length 100 km. Assume signals travel over each link at a speed of 10^8 meters per second. Assume that the link bandwidth on each link is 1Mbps. Let the file be broken down into 1000 packets each of size 1000 bits. Find the total sum of transmission and propagation delays in transmitting the file from S to D?

(GATE 2012)

A. 1005 ms B. 1010 ms

C. 3000 ms D. 3003 ms

Explanation: Propagation Delay=100Km/10^8=1ms

Packet Transmission time=1000bits/10^6=1ms

No more assumptions are given. Thus, we assume it works like pipelining. Source S goes on sends packets and intermediate routers will behave store and forward style. Thus,

First packet first bit starts 0^{th} time which reaches R1 after 1ms

First packet last bit ends 1^{st} ms which reaches R1 after 2ms

Like this it reaches D after 6^{th} ms.

In the mean time, S will be continuing to pump packets. After first packet, for every 1ms one packet will be emerging at D. Thus, total time=999+6=1005ms

11. Consider an instance of TCP's Additive Increase Multiplicative Decrease (AIMD) algorithm where the window size at the start of the slow start phase is 2 MSS and the threshold at the start of the first transmission is 8 MSS. Assume that a timeout occurs during the fifth transmission. Find the congestion window size at the end of the tenth transmission.

(GATE 2012)

A. 8 MSS B. 14 MSS

C. 7 MSS D. 12 MSS

Answer: 8 or 7 is most optimal solutions.

Explanation for AIMD examples, see the notes.

12. A layer-4 firewall (a device that can look at all protocol headers up to the transport layer) cannot

(GATE 2011)

A. Block entire HTTP traffic during 9:00PM and 5:00AM

B. Block all ICMP traffic

C. Stop incoming traffic from a specific IP address but allow outgoing traffic to the same IP address

D. Block TCP traffic from a specific user on a multi-user system during 9:00PM and 5:00AM

Explanation: Since it is a layer 4 firewall it cannot block application layer protocol like HTTP.

13. Consider different activities related to email.

(GATE 2011)

m1: Send an email from a mail client to a mail server

m2: Download an email from mailbox server to a mail client

m3: Checking email in a web browser

Which is the application level protocol used in each activity?

A. m1:HTTP m2:SMTP m3:POP

B. m1:SMTP m2:FTP m3:HTTP

C. m1: SMTP m2: POP m3: HTTP

D. m1: POP m2: SMTP m3:IMAP

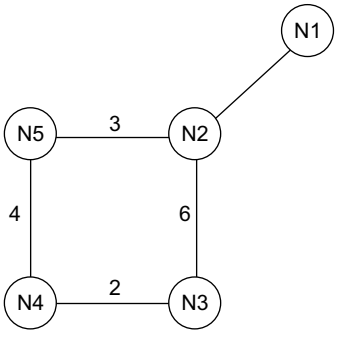
Explanation: Sending an email will be done through user agent and message transfer agent by SMTP, downloading an email from mail box is done through POP, checking email in a web browser is done through HTTP

14. HTML (Hyper Text Markup Language) has language elements which permit certain actions other than describing the structure of the web document. Which one of the following actions is NOT supported by pure HTML (without any server or client side scripting) pages? **(GATE 2011)**

A. Embed web objects from different sites into the same page

B. Refresh the page automatically after a specified interval
C. Automatically redirect to another page upon download
D. Display the client time as part of the page

15. Consider a network with five nodes, N1 to N5, as shown below **(GATE 2011)**



The net work uses a Distance Vector Routing protocol. Once the routes have stabilised, the distance vectors at different nodes are as following.

N1: (0,1,7,8,4) N2 : (1,0,6,7,3)

N3: (7,6,0,2,6) N4 : (8,7,2,0,4)

N5 : (4,3,6, 4,0)

Each distance vector is the distance of the best known path at that instance to nodes, N1 to N5, where the distance to itself is 0. Also, all links are symmetric and the cost is identical in both directions. In each round, all nodes exchange their distance vectors with their respective neighbours. Then all nodes update their distance vectors. In between two rounds, any change in cost of a link will cause the two incident nodes to change only that entry in their distance vectors.

The cost of link N2-N3 reduces to 2 in (both directions). After the next round of updates, what will be the new distance vector at node, N3? **(GATE 2011)**

A. (3. 2, 0, 2, 5) B. (3, 2, 0, 2, 6)

C. (7, 2, 0, 2, 5) D. (7, 2, 0, 2, 6)

Explanation: N3 tries to update its vector from the vectors of its neighbours N2 and N4 and its own vector.

Vector of N2	Vector of N3	Vector of N4	Minimum	Via station of out-going line	Now add current delays of via stations
1	7	8	1	N2	1+2=**3**
0	6	7	0	N2	0+2=**2**
6	0	2	0	N3	**0**
7	2	0	0	N4	0+2=**2**
3	6	4	3	N2	3+2=**5**

16. After the update in the previous question, the link N1-N2 goes down. N2 will reflect this change immediately in its distance vector as cost, ∞. After the NEXT ROUND of update, what will be the cost to N1 in the distance vector of N3? **(GATE 2011)**

A. 3 B. 9

C. 10 D. ∞

Explanation:

N3 has neighbors N2 and N4

N2 has made entry ∞

N4 has the distance of 8 to N1

N3 has the distance of 2 to N4

So 2 + 8 = 10

17. One of the header fields in an IP datagram is the Time to Live (TTL) field. Which of the following statements best explains the need for this field? **(GATE 2010)**

A. It can be used to prioritize packets
B. It can be used to reduce delays
C. It can be used to optimize throughput
D. It can be used to prevent packet looping

18. Which one of the following is not a client server application? **(GATE 2010)**

A. Internet chat B. Web browsing

C. E-mail D. Ping

19. Suppose computers A and B have IP addresses 10.105.1.113 and 10.105.1.91 respectively and they both use the same net mask N. Which of the values of N given below should not be used if A and B should belong to the same network? **(GATE 2010)**

A. 255.255.255.0 B. 255.255.255.128

C 255.255.255.192 D 255.255.255.224

Explanation: A: 10.105.1.113= 10.105.1. 0111 0001

B: 10.105.1.91 = 10.105.1. 0101 1011

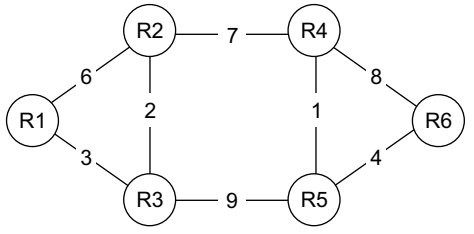
If we apply bitwise AND with netmask, we will be getting netID. Except for option D, for all remaining we are getting same netID for both A and B. The following table shows last byte for the sake of brevity.

	Bitwise AND with A	BitWise AND with B
255.255.255.0	10.105.1.0000 0000	10.105.1.0000 0000
255.255.255.128	10.105.1.0000 0000	10.105.1.0000 0000
255.255.255.192	10.105.1.0100 0000	10.105.1.0100 0000
255.255.255.224	10.105.1.**0110 0000**	10.105.1.**0100 0000**

20. Consider a network with 6 routers R1 to R6 connected with links having weights as shown in the following diagram **(GATE 2010)**

All the routers use the distance vector based routing algorithm to update their routing tables. Each router

starts with its routing table initialized to contain an entry for each neighbour with the weight of the respective connecting link. After all the routing tables stabilize, how many links in the network will never be used for carrying any data?



A. 4 B. 3
C. 2 D. 1

21. Suppose the weights of all unused links in the previous question are changed to 2 and the distance vector algorithm is used again until all routing tables stabilize. How many links will now remain unused? **(GATE 2010)**

A. 0 B. 1
C. 2 D. 3

22. While opening a TCP connection, the initial sequence number is to be derived using a time-of-day (ToD) clock that keeps running even when the host is down. The low order 32 bits of the counter of the ToD clock is to be used for the initial sequence numbers. The clock counter increments once per millisecond. The maximum packet lifetime is given to be 64s. Which one of the choices given below is closest to the minimum permissible rate at which sequence numbers used for packets of a connection can increase? **(GATE 2009)**

A. 0.015/s B. 0.064/s
C. 0.135/s D. 0.327/s

23. Let $G(x)$ be the generator polynomial used for CRC checking. What is the condition that should be satisfied by $G(x)$ to detect odd number of bits in error? **(GATE 2009)**

A. $G(x)$ contains more than two terms
B. $G(x)$ does not divide $1 + x^k$, for any knot exceeding the frame length
C. $1 + x$ is a factor of $G(x)$
D. $G(x)$ has an odd number of terms

Frames of 1000 bits are sent over a 10^6 bps duplex link between two hosts. The propagation time is 25ms. Frames are to be transmitted into this link to maximally pack them in transit (within the link).

24. What is the minimum number of bits I that will be required to represent the sequence numbers distinctly? Assume that no time gap need to be given between transmission of two frames.

A. l=2 B. l=3
C. l=4 D. l=5

Explanation:

Transmission delay for 1 frame = 1000/(10^6) = 1 ms

Propogation time = 25 ms

The sender can atmost transfer 25 frames before the first frame reaches the destination.

The number of bits needed for representing 25 different frames = 5

25. Suppose that the sliding window protocol is used with the sender window size of 2^1, where 1 is the number of bits identified in the earlier part and acknowledgements are always piggy backed. After sending 2^1 frames, what is the minimum time, the sender will have to wait before starting transmission of the next frame? (Identify the closest choice ignoring the frame processing time.) **(GATE 2009)**

A. 16ms B. 18ms
C. 20ms D. 22ms

Explanation:

Size of sliding window = 2^5 = 32

Transmission time for a frame = 1ms

Total time taken for 32 frames = 32ms

The sender cannot receive acknowledgement before round trip time which is 50ms.

After sending 32 frames, the minimum time the sender will have to wait before starting transmission of the next frame = 50 – 32 = 18

26. What is the maximum size of data that the application layer can pass on to the TCP layer below? **(GATE 2005)**

A. Any size
B. 2^16 bytes-size of TCP header
C. 2^16 bytes
D. 1500 bytes

27. Which of the following system calls results in the sending of SYN packets? **(GATE 2005)**

A. Socket B. Bind
C. Listen D. Connect

28. In the slow start phase of the TCP congestion control algorithm, the size of the congestion window **(GATE 2005)**

A. Does not increase
B. Increases linearly
C. Increases quadratically
D. Increases exponentially

Explanation: Although the name is slow start, during the slow start phase, window size is increased by the number of segments acknowledged, which means window size grows exponentially. This happens until either an acknowledgment is not received for some segment or a predetermined threshold value is reached. See this for more details.

29. If a class B network on the Internet has a subnet mask of 255.255.248.0, what is the maximum number of hosts per subnet? **(GATE 2005)**

A. 1022 B. 1023
C. 2046 D. 2047

Explanation: The binary representation of subnet mask is 11111111.11111111.11111000.00000000. There are 21 bits set in subnet. So 11 (32-21) bits are left for host ids. Total possible values of host ids is 2^11 = 2048. Out of these 2048 values, 2 addresses are reserved. The address with all bits as 1 is reserved as broadcast address and address with all host id bits as 0 is used as network address of subnet.

In general, the number of addresses usable for addressing specific hosts in each network is always 2^N – 2 where N is the number of bits for host id.

30. An organisation has a class B network and wishes to form subnets for 64 departments. The subnet mask would be **(GATE 2005)**

A. 255.255.0.0 B. 255.255.64.0
C. 255.255.128.0 D. 255.255.252.0

Explanation: The size of network ID is 16 bit in class B networks. So bits after 16th bit must be used to create 64 departments. Total 6 bits are needed to identify 64 different departments. Therefore, subnet mask will be 255.255.252.0.

31. Suppose the round trip propagation delay for a 10 Mbps Ethernet having 48-bit jamming signal is 46.4 ms. The minimum frame size is **(GATE 2008)**

A. 94 B. 416
C. 464 D. 512

Explanation:

Transmission Speed = 10Mbps.

Round trip propagation delay = 46.4 ms

The minimum frame size = (Round Trip Propagation Delay) * (Transmission Speed) = 10*(10^6)*46.4*(10^–3) = 464 * 10^3 = 464 Kbit

32. Packets of the same session may be routed through different paths in **(GATE 2006)**

A. TCP, but not UDP B. TCP and UDP
C. UDP, but not TCP D. Neither TCP nor UDP

33. The maximum window size for data transmission using the selective reject protocol with n-bit frame sequence numbers is **(GATE 2006)**

A. 2^n B. 2^(n–1)
C. 2^n – 1 D. 2^(n–2)

34. The address resolution protocol (ARP) is used for **(GATE 2006)**

A. Finding the IP address from the DNS
B. Finding the IP address of the default gateway
C. Finding the IP address that corresponds to an MAC address
D. Finding the MAC address that corresponds to an IP address

Explanation: Address Resolution Protocol (ARP) is a request and reply protocol used to find MAC address from IP address.

35. In a network of LANs connected by bridges, packets are sent from one LAN to another through intermediate bridges. Since more than one path may exist between two LANs, packets may have to be routed through multiple bridges. Why is the spanning tree algorithm used for bridge-routing? **(GATE 2006)**

A. For shortest path routing between LANs
B. For avoiding loops in the routing paths
C. For fault tolerance
D. For minimising collisions

ANSWER KEY

1. C	**2.** D	**3.** C	**4.** B
5. C	**6.** C	**7.** C & D	**8.** C
9. A	**10.** A	**11.** Incomplete question	
12. A	**13.** C	**14.** D	**15.** A
16. C	**17.** D	**18.** D	**19.** D
20. D	**21.** A	**22.** A	**23.** C
24. D	**25.** B	**26.** A	**27.** D
28. D	**29.** C	**30.** D	**31.** C
32. B	**33.** B	**34.** D	**35.** B

CHAPTER EIGHT

Introduction to HTML, XML and Client Server Programming

8.1 Introduction

Internet and browsing the World Wide Web (WWW) have become common in everyone's daily life. Internet is a collection of computers, connected through various inter-connecting devices such as bridges, routers, gateways; whereas WWW is a collection of web pages or hyper documents. Hyper documents are ones which may contain text in different size, colours, audio, video, pictures and hyper links. We can create hyper documents using many languages such as HTML, DHTML, XML, JSP, ASP, Java, Perl, PHP, etc. Out of them, HTML based pages are very simple to create.

8.1.1 What is HTML?

- HTML is a language for describing web pages.
- HTML stands for **H**yper **T**ext **M**arkup **L**anguage.
- HTML is not a programming language, it is a **markup language.**
- A markup language is a collection of **markup tags.**
- HTML uses **markup tags** to describe web page content.

8.1.2 HTML Tags

HTML markup tags are usually called HTML tags and are surrounded by **angle brackets** like <html>. HTML tags normally **come in pairs** like and . The first tag in a pair is the **start tag**, the second tag is the **end tag**. Start and end tags are also called **opening tags** and **closing tags**. HTML tags are not case sensitive: <P> means the same as <p>. Plenty of web sites use uppercase HTML tags in their pages. W3Schools use lowercase tags because the World Wide Web Consortium (W3C) **recommends** lowercase in HTML 4, and **demands** lowercase tags in future versions of ((X)HTML). A web browser (like Internet Explorer, Netscape Navigator, Chrome, or Firefox) is used to read HTML documents and display them as web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

8.1.3 My First Web Page

Suppose that we want to create a simple html page. Write the following statements in a file with the name "first.htm" or "first.html" using any editor such as notepad. Now, open Internet Explorer (or any other browser) and open this "first.htm" file through **OPEN** (and browse) option under **File** menu item.

```
<html>
<body>
Welcome for Learning to succeed GATE CSE/IT.
</body>
</html>
```

Here,

- The text between <html> and </html> describes the web page.
- The text between <body> and </body> is the visible page content.

The following is the output in Internet Explorer when it is opened (Figure 8.1).

Figure 8.1 A simple HTML page

Unlike program files (such as C, C++, Java), HTML pages are not translated. Rather they will be interpreted by the browsers. Browsers take the responsibility of displaying (playing) text/others according to the given tags. Browsers will not display any errors if the web page contains erroneous tags or erroneous combination of tags or wrongly spelt tags. They simply display with some default behaviour. One may be wondering if we say that we can even open a simple text file (without any HTML tags) also using a Web browser!

■ **Exercise** Try opening the above "first.htm" file after removing HTML tags.

8.1.4 <body> Tag

This is where we will begin writing our document and placing our HTML codes. We can use many options with body. The following can be used to specify background colour or background image for our page by changing body tag as:

<body bgcolor="red"> To assign red as the **background colour.**
<body background="image.jpg"> To make image file **image.jpg** as the **background image.**

8.1.5 <head> Tag

This tag contains information about the page such as the TITLE, META tags for proper Search Engine indexing, STYLE tags, which determine the page layout, and JavaScript coding for special effects.

8.1.5.1 Meta Tags

Meta tags are inserted in the <head> region of our web page and the same is not viewable in the browser. Meta tags in general "work behind the scenes" and a human visitor will not care too much about them. They are usually used to help increase search engine rankings by including relevant information about the page. Of course, if we want to be ranked #1 in a search engine like Google, meta tags alone will not help us as search engines use a different algorithm in site ranking. Good content will make more people link to our page, thus driving up the site ranking. Here is an example of a with web page with some meta tags.

```
<head>
</title> Indian student world </title>
```

```
<meta name = "description" content = "All of you  who want to know about Indian students, who thrive
for marks rather than knowledge.">
<meta name = "keywords" content = "Indian students, marks, knowledge">
<meta name = "robots" content = "noindex">
</head>
```

In the above page, we have described three things using meta tags.

Description– Brief statement describing the page. This will be displayed in response to a search. That is, if a search engine selects this page as having the keywords entered by the user, it displays this description as the results of the search.

Keywords– What a user would type into a search engine to find this page.

Robots – Should web spiders be allowed to index this page? (we can control whether our page should be indexed by the search engines or not With this tag).

Evidently, meta tags **do not** have an end tag. Another **useful** meta tag is REFRESH. For instance, use the following little snippet of code for redirecting to another site.

```
<meta http-equiv="refresh"
content="5;url=http://www.site.com/">
```

The number 5 inside the content indicates the number of seconds before the browser will attempt to redirect you before jumping to another page.

Also, the following Content-type tells the browser what sort of character set (iso-8869-1) is to be used.

```
<meta http-equiv="Content-Type" content="text/html;  charset=iso-8869-1" >
```

In addition, for adding Favorites Icon, we can put the following code inside the <head> tag to link it to the required icon **favicon.ico**:

```
<link REL="SHORTCUTAICON"
HREF= http://www.site.com/favicon.ico >
```

8.1.6 How to title a Page?

We can have the required title to appear in the title bar with the help of <title> and </title> options. For example, take the following simple html file (title.htm) and view through IE.

```
<html>
<title>
This is My Second Web Page
</title>
<body>
I wish I will succeed in my GATE. I pray all mighty for the same.
</body>
</html>
```

When we open the html file with the above content through IE, we get a screen as shown below (Figure 8.2) . Observe the circled portion which is called as the title portion.

Figure 8.2 Web page with title

8.1.7 How to make Lines (horizontal rules)?

Making a horizontal rule is probably the easiest thing to write in HTML. (This is the command (or tag) for having a horizontal rule in our web page. Do note that we don't need any end tag for this tag.) **<hr>**
We will see a horizontal line in our page which stretches across whole page as shown below.

8.1.8 Empty HTML Elements (Line Breaks)

HTML elements without content are called empty elements. Empty elements can be built-in the start tag.
**
** is an empty element without a closing tag (it defines a line break).
In XHTML, XML, and future versions of HTML, all elements must be closed. Adding a slash to the start tag, like <br />, is the proper way of closing empty elements, accepted by HTML, XHTML and XML. Even if <br> works in all browsers, writing <br /> instead is more future proof.

8.1.9 How to make text in different Colours?

Changing the colour of the text is also very simple. Here is the tag <font>.

```
<font face="" size = 9 color="">
```

(Here is where you write your text.)
You can change the font size to make it bigger or smaller. In between the ""should you, put the code of whichever colour you, wish to use. The following are the colour codes of various colours.

Basic Colors

White rgb=#FFFFFF
Black rgb=#000000
Red rgb=#FF0000
Green rgb=#00FF00
Blue rgb=#0000FF
Magenta rgb=#FF00FF
Cyan rgb=#00FFFF
Yellow rgb=#FFFF00
Aquamarine rgb=#70DB93
Baker's Chocolate rgb=#5C3317
Blue Violet rgb=#9F5F9F
Brass rgb=#B5A642
Bright Gold rgb=#D9D919
Dark Purple rgb=#871F78
Dark Slate Blue rgb=#6B238E
Dark Slate Grey rgb=#2F4F4F
Dark Tan rgb=#97694F
Dark Turquoise rgb=#7093DB
Dark Wood rgb=#855E42
Dim Grey rgb=#545454
Dusty Rose rgb=#856363
Feldspar rgb=#D19275

Brown rgb=#A62A2A
Bronze rgb=#8C7853
Bronze II rgb=#A67D3D
Cadet Blue rgb=#5F9F9F
Cool Copper rgb=#D98719
Copper rgb=#B87333
Coral rgb=#FF7F00
Corn Flower rbg=#42426F
Dark Brown rgb=#5C4033
Dark Green rgb=#2F4F2F
Dark Green Copper rgb=#4A766E
Dark Olive Green rgb=#4F4F2F
Dark Orchid rgb=#9932CD
Midnight Blue rgb=#2F2F4F
Navy Blue rgb=#23238E
Neon Blue rgb=#4D4DFF
Neon Pink rgb=#FF6EC7
New Midnight Blue rgb=#00009C
New Tan rgb=#EBC79E
Old Gold rgb=#CFB53B
Orange rgb=#FF7F00
Orange Red rgb=#FF2400

```
Firebrick rgb=#8E2323
Forest Green rgb=#238E23
Gold rgb=#CD7F32
Goldenrod rgb=#DBDB70
Grey rgb=#C0C0C0
Green Copper rgb=#527F76
Green Yellow rgb=#93DB70
Hunter Green rgb=#215E21
Indian Red rgb=#4E2F2F
Khaki rgb=#9F9F5F
Light Blue rgb=#C0D9D9
Light Grey rgb=#A8A8A8
Light Steel Blue rgb=#8F8FBD
Light Wood rgb=#E9C2A6
Lime Green rgb=#32CD32
Mandarian Orange rgb=#E47833
Maroon rgb=#8E236B
Medium Aquamarine rgb=#32CD99
Medium Blue rgb=#3232CD
Medium Forest Green rgb=#6B8E23
Medium Goldenrod rgb=#EAEAAE
Medium Orchid rgb=#9370DB
Medium Sea Green rgb=#426F42
Medium Slate Blue rgb=#7F00FF
Medium Spring Green rgb=#7FFF00
Medium Turquoise rgb=#70DBDB
Medium Violet Red rgb=#DB7093
Medium Wood rgb=#A68064
```

```
Orchid rgb=#DB70DB
Pale Green rgb=#8FBC8F
Pink rgb=#BC8F8F
Plum rgb=#EAADEA
Quartz rgb=#D9D9F3
Rich Blue rgb=#5959AB
Salmon rgb=#6F4242
Scarlet rgb=#8C1717
Sea Green rgb=#238E68
Semi-Sweet Chocolate rgb=#6B4226
Sienna rgb=#8E6B23
Silver rgb=#E6E8FA
Sky Blue rgb=#3299CC
Slate Blue rgb=#007FFF
Spicy Pink rgb=#FF1CAE
Spring Green rgb=#00FF7F
Steel Blue rgb=#236B8E
Summer Sky rgb=#38B0DE
Tan rgb=#DB9370
Thistle rgb=#D8BFD8
Turquoise rgb=#ADEAEA
Very Dark Brown rgb=#5C4033
Very Light Grey rgb=#CDCDCD
Violet rgb=#4F2F4F
Violet Red rgb=#CC3299
Wheat rgb=#D8D8BF
Yellow Green rgb=#99CC32
```

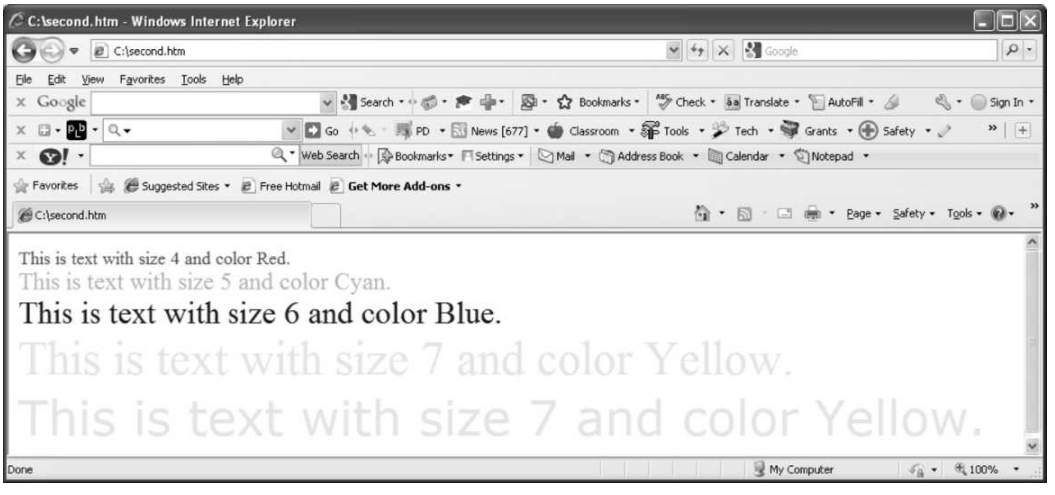


Figure 8.3 Web page Having text with colours

The following html file ("second. htm") with font tags will show the web page as shown above (Figure 8.3). As usual, we run IE and open second.htm file.

```
<html>
<body>
<font size="4" color="red">
This is text with size 4 and color Red.<br></font>
<font size="5" color="cyan">
This is text with size 5 and color Cyan.<br>
</font>
<font size="6" color="blue">
This is text with size 6 and color Blue.<br>
</font>
<font size="7" color="yellow">
This is text with size 7 and color Yellow.<br>
</font>
<font face="verdana" size="8" color="yellow">
This is text with size 7 and color Yellow.<br>
</font>
</body>
</html>
```

8.1.10 How to make text in different Sizes

In HTML, there exist two ways to display text in different sizes. The first way is using <font> tag which is explained above in section 8.1.9. How to make text in (different Colours. This is slighthy complicated. The approach which is shown below is very simple and we can display the text in 7 different sizes.

We put text between the <h#> tags to change the size; with <h1> being the biggest and <h7> being the smallest size. Do remember to end the tags with </h#>.

See the following file "third.htm" with the above tags. When we open this html file using the IE, the web page looks like the following Figure (Figure 8.4).

```
<html>
<body>
<h1>This is text with size 1.</h1>
<h2>This is text with size 2.</h2>
<h3>This is text with size 3.</h3>
<h4>This is text with size 4.</h4>
<h5>This is text with size 5.</h5>
<h6>This is text with size 6.</h6>
<h7>This is text with size 7.</h7>
</body>
</html>
```

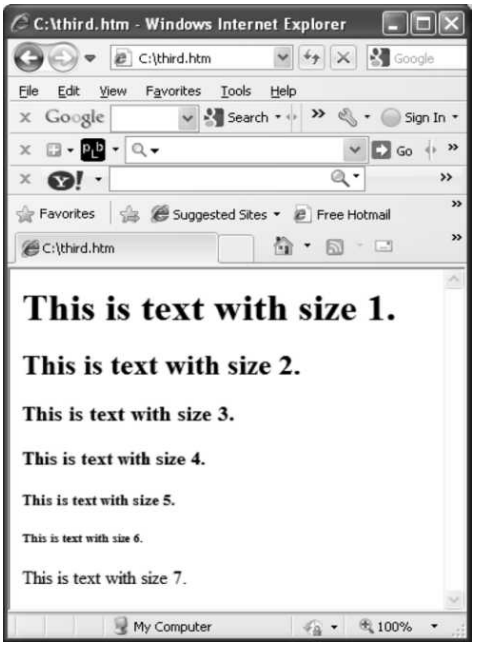


Figure 8.4 A web page Having text with different sizes

8.1.11 Let us enjoy the fonts in a variety of Styles

We can also use the following tags for displaying a variety of font styles.

```
<b>          For bold text.
<u>          For Underlined text
<i>          For Italic text
<strike>     To struck some text.
<em>         For Emphasized text
<tt>         For teletype type text.
<big>        For big sized text.
<small>      For Small text.
<blink>      For blinking text.
```

We can combine the above tags with text size tags such as <h1>, <h2>, etc. For example, the following html file (fourth.htm) shows the use of the same. The effect of these tags is shown in the following Figure (Figure 8.5).

```
<html>
<body>
<i>The Tata McGrawHill Co.<br> </i>
<u> N.B. Venkateswarlu<br></u>
<b>
This is an example bold text.<br></b>
<strike>This is an example striked text.<br></strike>
<h1><i><b><u><strike>
Best of Luck for your GATE examination<br>
</h1></i></b></u></strike>
</body>
</html>
```

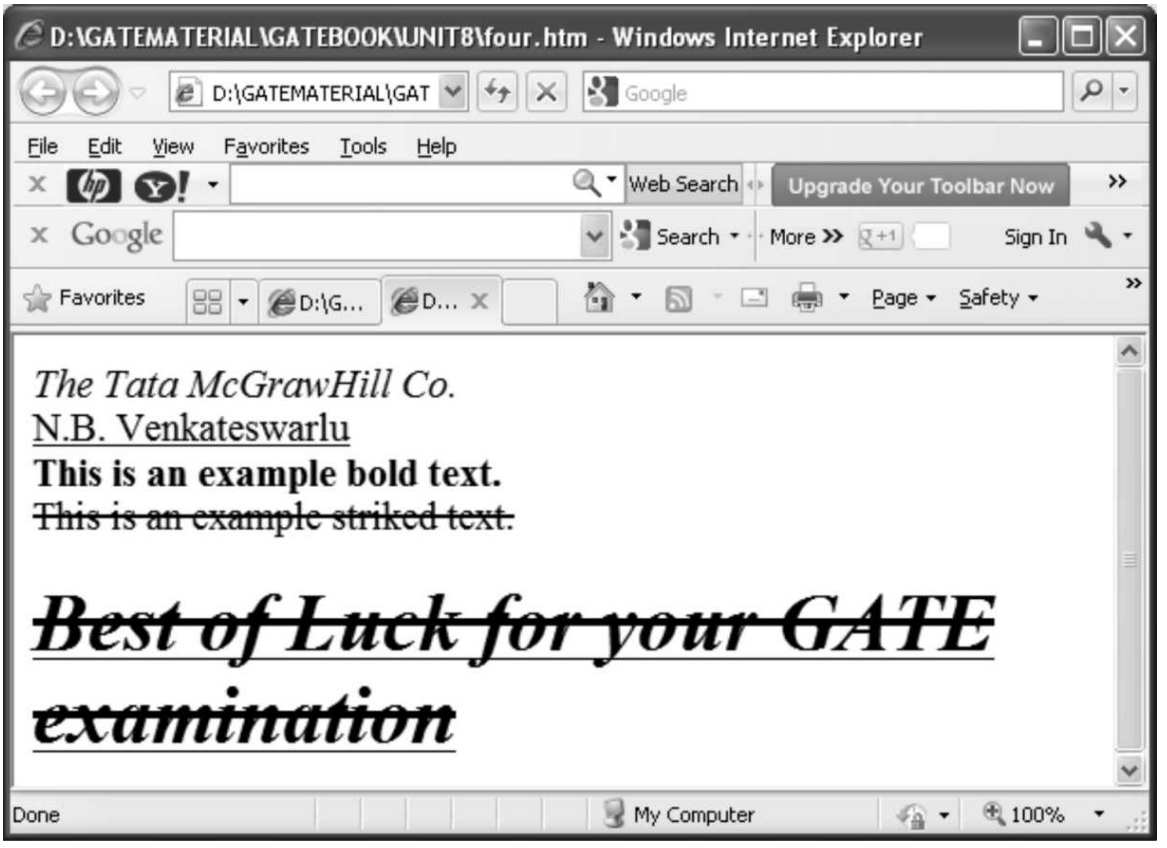


Figure 8.5 A web page Having text in variety of styles

Also, we can use , , <dfn>, <code>, <samp>, <kbd>, <var>, and <cite> tags for formatting the text. Their use is illustrated in the following table.

Tag	Description
	Renders as emphasized text
	Renders as strong emphasized text
<dfn>	Defines a definition term
<code>	Defines computer code text
<samp>	Defines sample computer code
<kbd>	Defines keyboard text
<var>	Defines a variable part of a text
<cite>	Defines a citation

8.1.12 How to make scrolling Text?

We can display a message to be scrolled in our HTML page with <MARQUEE> tag. The following code will scroll the message "Hello How are you my dear?". We can change the SCROLLDELAY and other parameters.

```
<MARQUEE behavior=alternate SCROLLDELAY="10" SCROLLAMOUNT="1"> Hello How are you my dear?
</MARQUEE>
<MARQUEE behavior=alternate SCROLLDELAY="10"  SCROLLAMOUNT="1">
<img  src="imagefilename with extension such as bmp" >
</MARQUEE>
```

We can also make an image to scroll using the following tag.

8.1.13 How to make "freestyle" Text?

Making "freestyle" text (in other words, text "as is") is as easy as making it bold or italic. We use <pre> tag for this purpose. This feature is very useful to display programs, etc., in the html page. Consider the following html file (fifth.htm) which contains <pre> tags. Output looks like the following figure, when opened through IE (Figure 8.6). We may observe the empty lines, spaces, etc., which are shown as it is (verbatim).

```
<html><body>
  <pre>

  int main()
{

        printf("Hello Welcome to HTML\n");

}
</pre>
</body></html>
```

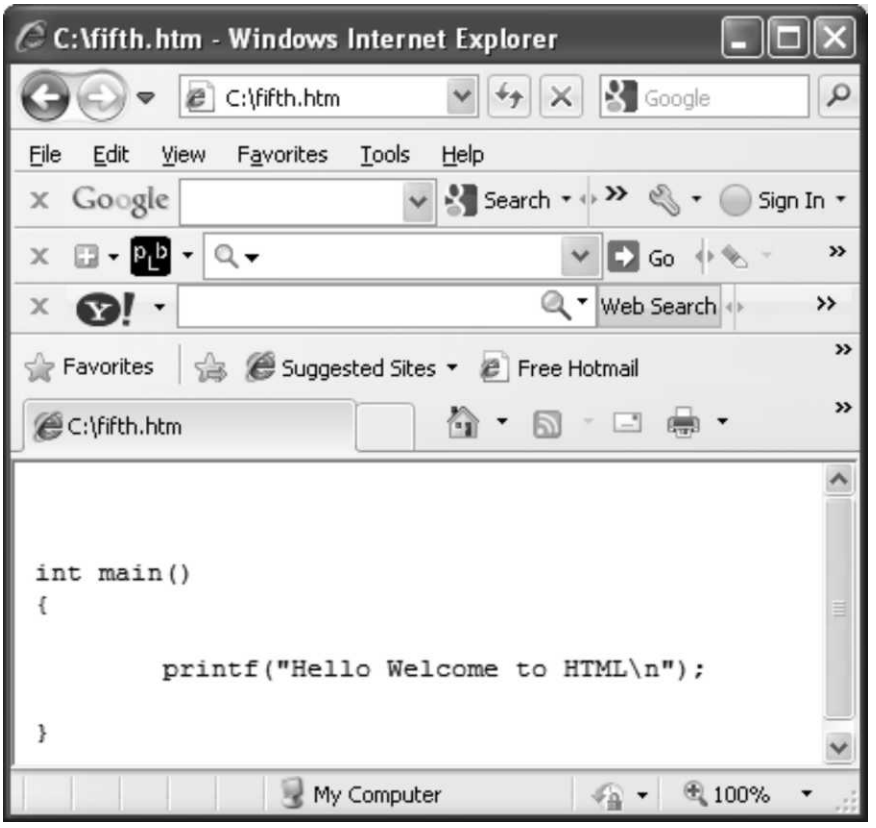


Figure 8.6 A Web page that displays verbatim output

8.1.14 To link to your e-mail ID

We can create a link to our email ID from our page by using the following tag. Here, we are using a URL (uniform resource locator) that uses mailto protocol.

```
<a href="mailto:venkatritch@gmail.com">Contact me!</a>
```

8.1.15 HTML attributes

With HTML tags, we can use different attributes.

Attribute	Value	Description
class	*class_rule or style_rule*	The class of the element
id	*id_name*	A unique id for the element
style	*style_definition*	An inline style definition
title	*tooltip_text*	A text to display in a tool tip

The attributes listed below are standard, and are supported by all HTML and XHTML tags, with a few exceptions.

8.1.15.1 Core Attributes

Not valid in **base**, **head, html, meta, param, script, style,** and **title** elements.

Attribute	Value	Description
class	*classname*	Specifies a classname for an element
id	*id*	Specifies a unique id for an element
style	*style_definition*	Specifies an inline style for an element
title	*text*	Specifies extra information about an element

8.1.15.2 Language Attributes

Not valid in **base, br, frame, frameset, hr, iframe, param,** and **script** elements.

Attribute	Value	Description
dir	ltr rtl	Specifies the text direction for the content in an element
lang	*language_code*	Specifies a language code for the content in an element. Language code reference
xml:lang	*language_code*	Specifies a language code for the content in an element, in XHTML documents. Language code reference

8.1.15.3 Keyboard Attributes

Attribute	Value	Description
accesskey	*character*	Specifies a keyboard shortcut to access an element
tabindex	*number*	Specifies the tab order of an element

Comments are written like this: <!-- This is a comment -->

8.1.15.4 Some more HTML Text Formatting Tags

Tag	Description
<b>	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<s>	Deprecated. Use <del> instead
<strike>	Deprecated. Use <del> instead
<u>	Deprecated. Use styles instead

8.1.15.5 "Computer Output" Tags

Tag	Description
<code>	Defines computer code text
<kbd>	Defines keyboard text
<samp>	Defines sample computer code
<tt>	Defines teletype text

Tag	Description
<var>	Defines a variable
<pre>	Defines preformatted text
<listing>	Deprecated. Use <pre> instead
<plaintext>	Deprecated. Use <pre> instead
<xmp>	Deprecated. Use <pre> instead

8.1.15.6 Citations, Quotations and Definition Tags

Tag	Description
<abbr>	Defines an abbreviation
<acronym>	Defines an acronym
<address>	Defines an address element
<bdo>	Defines the text direction
<blockquote>	Defines a long quotation
<q>	Defines a short quotation
<cite>	Defines a citation
<dfn>	Defines a definition term

8.1.15.7 HTML Styles

We can specify HTML Style such as:

```
style="background-color:yellow"
style="font-size:10px"
style="font-family:Times"
style="text-align:center"
```

8.1.15.8 Font Family, Color and Size

```
<p style="font-family:courier new; color:red; font-size:20px">
```

8.1.15.9 Text Alignment

```
<h1 style="text-align:center">
```

8.1.16 How to make an Image as Background?

We have some control on the behavior of the total page. This is achieved through body tag. For example, body tag can be used for changing the colours of the background, the links, etc., as shown below:

```
<body bgcolor="" text="" alink ="" link="" vlink="">
```

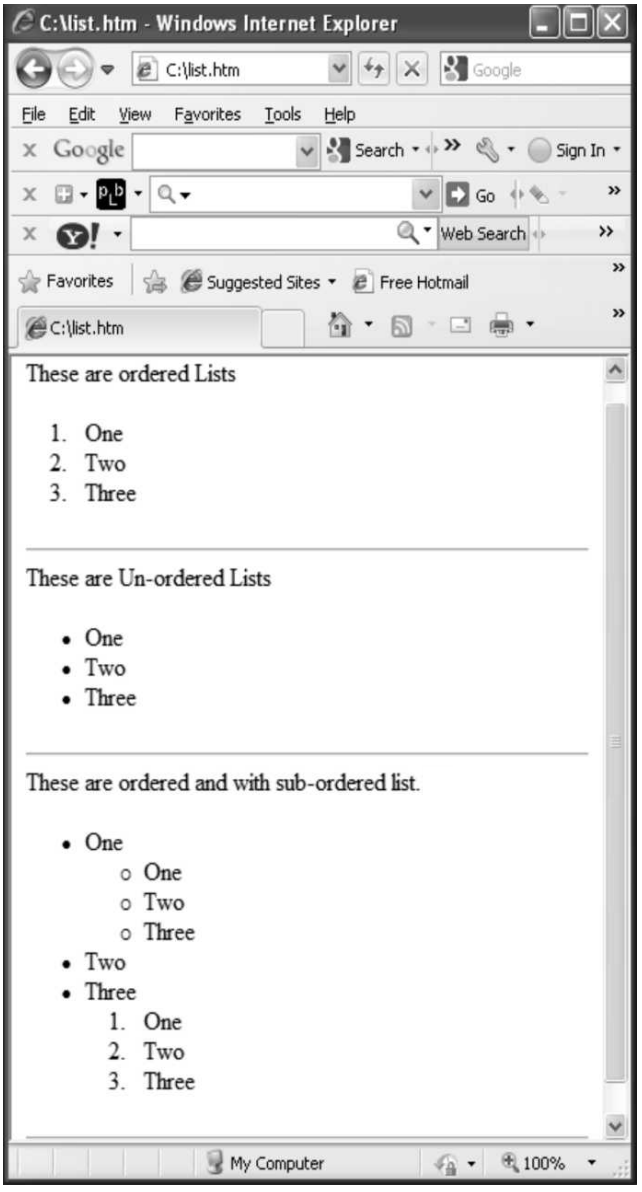
In between all the ""s, we have to add the required colour codes. Here, alink indicates active links, link indicates links, vlink indicate visited links. We can specify the required colours for these links. To make our background into a pattern or a picture, we should first upload a pattern or picture into our directory as a (JPG or gif) image or know the URL of a good background image. For example:

```
<body background="http://www.what.com/what.gif" text="#000000" link="#000000" vlink="#000000">
```

This would make our background into what.gif image. It would make the writing, the links, and the visited links all black. Also, text attribute is used to specify the colour of the text in the document. In this case, we are specifying the text also as black.

8.1.17 How to make Lists (Ordered and Un-Ordered) lists

We can create lists (also referred as bullets) in HTML. We can have ordered and un-ordered lists. Ordered lists can be created using <ol> and tags; while un-ordered lists can be created through <ul> and . In both the lists, items are specified with <li> and tags. See the following demonstration html file (list.htm), whose output in the IE will be as shown in figure 8.7. In the following html page we have shown sub-lists also. However, look and feel of these lists depends on the browser. We do not have much control on them. In addition, we can maintain anything such as text, images, text boxes, etc., as lists.




```
<html><body>
These are ordered Lists <br>
<ol>
<li> One </li><br>
<li> Two </li><br>
<li> Three </li><br>
</ol>
<hr>
These are Un-ordered Lists <br>
<ul>
<li> One </li><br>
<li> Two </li><br>
<li> Three </li><br>
</ul>
<hr>
These are ordered and with
sub-ordered list. Lists <br>
<ul>
<li> One </li><br>
<ul>
<li> One </li><br>
```

Figure 8.7 A Web page with Lists

8.1.17.1 Definition Lists

A definition list is not a list of single items. It is a list of items (terms), with a description of each item (term). A definition list starts with a <dl> tag (**d**efinition **l**ist). Each term starts with a <dt> tag (**d**efinition **t**erm). Each description starts with a <dd> tag (**d**efinition **d**escription).

```
<dl>
<dt>Coffee</dt>
<dd>Black hot drink</dd>
<dt>Milk</dt>
<dd>White cold drink</dd>
</dl>
```

Here is how it looks in a browser:

Coffee

Black hot drink

Milk

White cold drink

Inside <dd> tag, we can put paragraphs, line breaks, images, links, other lists.

List Tags

Tag	Description
	Defines an ordered list
	Defines an un-ordered list
	Defines a list item
<dl>	Defines a definition list
<dt>	Defines a term (an item) in a definition list
<dd>	Defines a description of a term in a definition list
<dir>	Deprecated. Use instead
<menu>	Deprecated. Use instead

8.1.18 How to make a Link to another Page (Hyper Links)?

We know that WWW is collection of hyper documents in which a group of words in a document are linked to other documents situated elsewhere in the Internet. In this section, we shall learn how to create links (hyper links or Uniform Resource Locators (URLs)) in our web page. Making links is VERY easy. Here is the format of link creation tag or anchor tag <a>:

```
<a href= "the URL of the link" >
```

For example, to make a link on our page to here it would look like this:

```
<a href="http://www.geocities.com/sunsetstrip/alley/5615/>
```

This will make a link but we won't be able to see it or use it. To make it work, we have to put some text in between the <a> and. We have to include the tag at the end of every link or the rest of our page will be the link. See the following html file (link.htm) which creates three hyper links with the help of anchor tags <a> and . If we view this file in IE, we will get output as shown in the figure 8.8. When we place mouse on any of the links, we will see that the cursor (pointer) changes to a hand-like icon. If we click on this link, we will get the corresponding web page. Thus, these words of current html document are logically connected to a file on another machine.

```
<html><body>
Select the Email Providers You Like<br>
<ol>
<li> <a href="http://mail.yahoo.com">Yahoo</a> </li><br>
<li> <a href="http://www.gmail.com">Gmail</a> </li><br>
<li> <a href="http://www.hotmail.com">Hotmail</a></li><br>
</ol>
</body> </html>
```

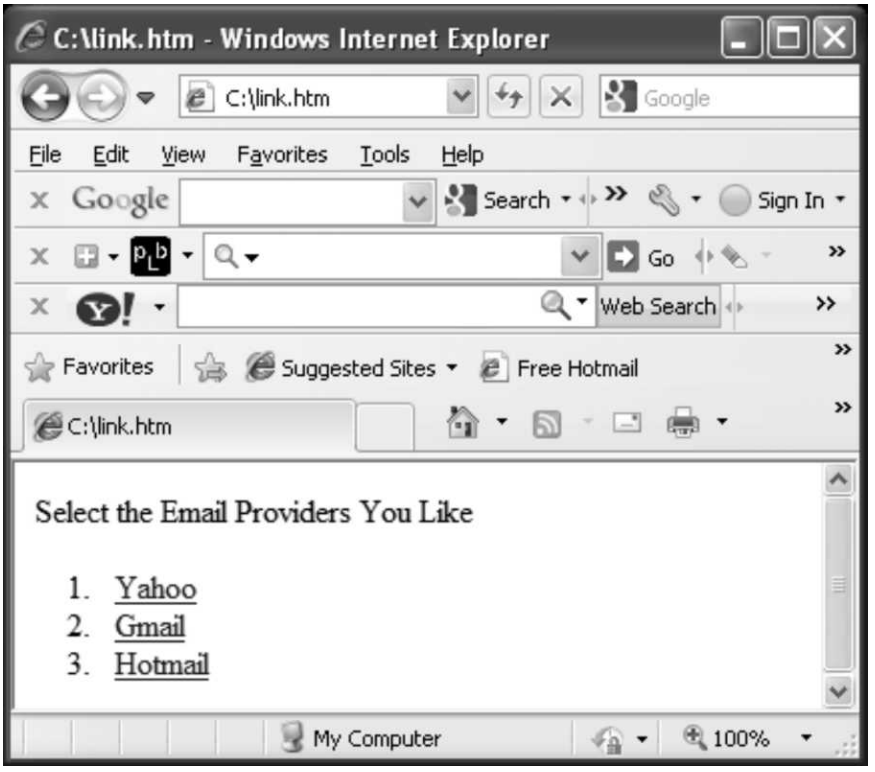


Figure 8.8 A web page with Hyper links

We can use target attributes with anchor tags to indicates **where** the linked document has to be opened. For example, the code below will open the document in a new browser window:

```
<a href="http://www.w3schools.com/" target= "_blank">Visit W3Schools!</a>
```

We also have named anchors which are explained in detail in the following section. Named anchors are not displayed in any special way. They are invisible to the reader.

Named anchor syntax: Any content

The link syntax to a named anchor: Any content

The # in the href attribute defines a link to a named anchor.

8.1.19 How to make a Link to Another Page in the Same Directory?

Making a link to a page in the same directory as your intitial page is VERY VERY easy. Here is the script to do it. **<a href= " address of page">**

For Example, here is how we made a link from this page to the page that shows you colours: **<a href="color.htm">**

How To Make Links in the Same Page. Sometimes we often require to move from one location to another location in a web page. This is achieved by defining local anchor tags. The following html file (intrapagelink.htm) shows how to use the same.

```
<html> <body>
<a href="#sec1">Section 1</a><br>
<a href="#sec2">Section 2</a><br>
<a href="#sec3">Section 3</a><br>
<a href="#sec4">Section 4</a><br>
<a name="sec1"><h1>Section1</h1></a><br>
This section is the first one.<br>
<a name="sec2"><h1>Section2</h2></a><br>
```

```
This section is the Second one.<br>
<a name="sec3"><h1>Section3</h1></a><br>
This section is the Third one.<br>
<a name="sec4"><h1>Section4</h1></a><br>
This section is the Fourth one.<br>
</body></html>
```

In the above html page, we have defined 4 local anchor tags with the name sec1, sec2, etc. When the above page is opened through IE, we will get display as shown in Figure 8.9 (a) If we click, section3, we get the display as shown in Figure 8.9 (b). That is, we have moved to section3 of the same page. Thus, we can move from one area to another area of the same web page using local anchor tags.

Figure 8.9 (a) A web page with local anchors

Figure 8.9 (b) A web page after clicking local anchor

8.1.20 How to display or insert an Image in our Web Page?

This is also an extremely simple command. It's basically a link but a little different. Here is the script:

```
<img src= " URL of the image ">
```

We can specify the local images also. If the images are in the same directory of the web page, then we only have to write their names.

8.1.21 How to make Images as Hyper Links?

Making images into links is easy as it involves combining both the anchor and img tags as shown below:

```
<a href="URL of the link"> <img src="whatever.gif"> </a>
```

We must make sure that the actual link reference comes before the image. Also the link closing tag must come last as shown above.

8.1.22 How to control the size of our Images?

To do this all we have to do is to add this into our image tags:
width=whatever height=whatever
For example,

```
<img src="whatever.gif" width=50 height=50>
```

8.1.23 How to make Text appear beside an Image?

Have you ever noticed that when you type beside an image the text only starts at the bottom of the picture and we end up with a big empty space? Well, this little addition to the image command will solve that problem:

```
align=right or align=left
```

For example,

```
<img src="whatever.gif" height=50 width=50 align=right>
```

8.1.24 How to make a portion of an Image Clickable?

With the help of <map> and </map> tags we can make an image clickable. Rather, we can define some portion of the image as clickable. That is, when we click in the defined portion of the image, some page will be loaded. To achieve this, first we have to load an image with <img> tag with usemap attribute as shown below. Also, we have to define the clickable areas as shown below. To define the clickable areas in the image, we will be using <area> and </area> tags.

```
<html> <body>
<p>Click on the sun or on one of the planets to watch it closer:</p>
<img src="planets.gif" width="145" height="126" alt="Planets" usemap="#planetmap"/>
<map name="planetmap">
   <area shape="rect"  coords="0,0,82,126" alt="Sun" href="sun.htm" />
  <area shape="circle" coords="90,58,3" alt="Mercury" href="mercur.htm" />
  <area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm" />
</map>
</body> </html>
```

Based on the location of our click in the image that is displayed in the current web page, the correspondings web pages will load according to the areas defined above (Figure 8.10).

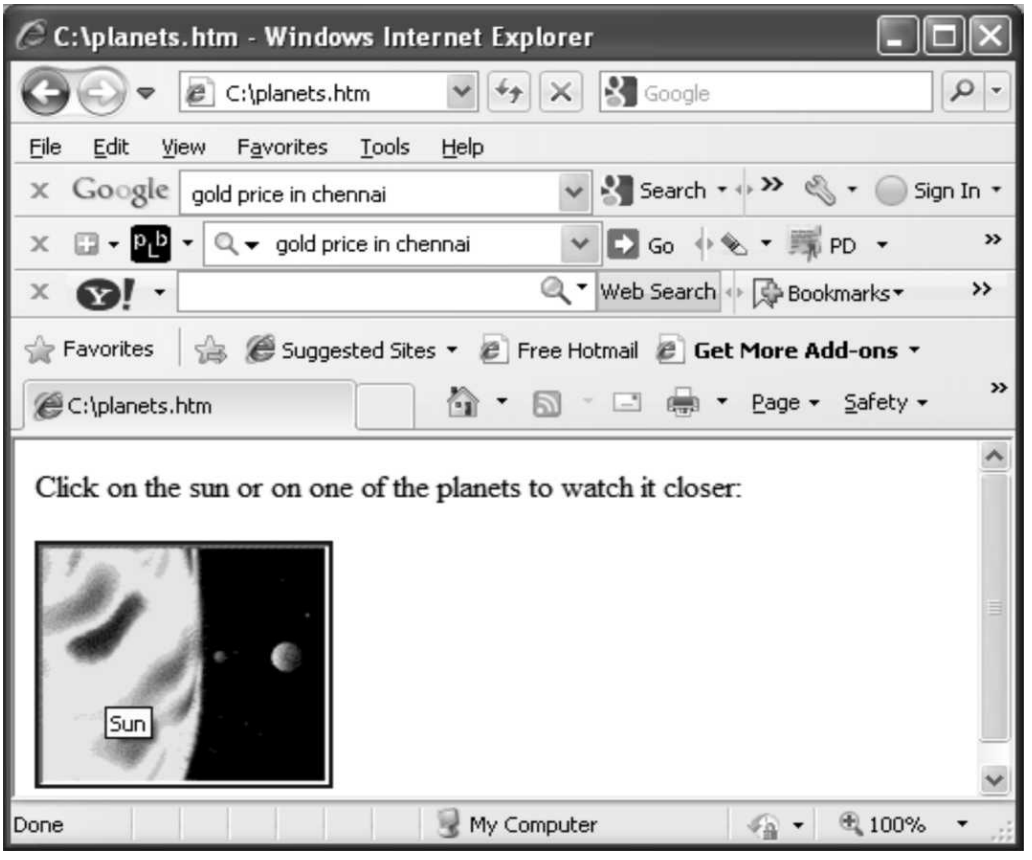


Figure 8.10 A web page with images as links

The following table explains the use of tags used in the above page.

Tag	Description
<img>	Defines an image
<map>	Defines an image map
<area>	Defines a clickable area inside an image map

8.1.24.1 The Alt Attribute with Images

The **alt** attribute is used to define an "alternate text" for an image. That is, when we place mouse on the image, this text will be displayed. Sometimes when the image does not load properly, we get an error icon (X mark). To know the name of the image that did not load, scroll the mouse pointer on the image. Of course, the value of the alt attribute is an author-defined text.

```
<img src="boat.gif" alt="Big Boat" />
```

8.1.25 How to make a Link that opens a New Browser Window?

This is as easy as adding a command at the end of the link command. Here is the command : target=body
For example,

```
<a href=" URL of place " target=body>
```

This will open a new window when we click a link, especially useful when making links during chatting, etc. Possible values for target attributes are given in the following table.

Value	Description
_blank	Open the linked document in a new window
_self	Open the linked document in the same frame as it was clicked (this is the default value)
_parent	Open the linked document in the parent frameset
_top	Open the linked document in the full body of the window
framename	Open the linked document in a named frame

The <base> tag is used to specify a default URL and a default target for all links on a page. The <base> tag specifies the base URL/target for all relative URLs in a document.

```
<head>
<base href="http://www.w3schools
com/images/" target="_blank" />
</head>
```

The <base> tag must go inside the <head> element. In HTML, the <base> tag has no end tag. Whereas in XHTML the <base> tag must be properly closed. It also contains href and target attributes.

8.1.26 How to make a Table?

Tables are a more complicated command than most of the ones here. To create a table we have to use <table> and </table> tags. We have to define each row with <tr> and </tr> tags. In each row, we can maintain fields with <td> and </td> fields. In the following html file ("table.htm"), we are creating a simple table with border. Also, last table uses <th> and </th> tags to specify table headings. If we open this html file from IE, we will see the display as shown in Figure 8.11.

```
<html><body>
A Simple Table with borders<br>
<table border=4>
<tr>
```

```
<td>S.No</td>
<td>Name</td>
</tr>
<tr><td>1</td><td>
Prof NB Venkateswarlu</td></tr>
<tr><td>2</td><td>
Dr. G.V. Saradamba</td></tr>
</table>
<br>
<br>
A Simple Table without borders.<br>
<table>
<tr>
<td>S.No</td>
<td>Name</td>
</tr>
<tr><td>1</td><td>
Prof NB Venkateswarlu</td></tr>
<tr><td>2</td><td>
Dr. G.V. Saradamba</td></tr>
</table>
<br>
<br>
A Simple Table with header
specification.<br>
<table border=4>
<tr>
<th>S.No</th>
<th>Name</th>
</tr>
<tr><td>1</td><td>
Prof NB Venkateswarlu</td></tr>
<tr><td>2</td><td>
Dr. G.V. Saradamba</td></tr>
</table>
</body>
</html>
```

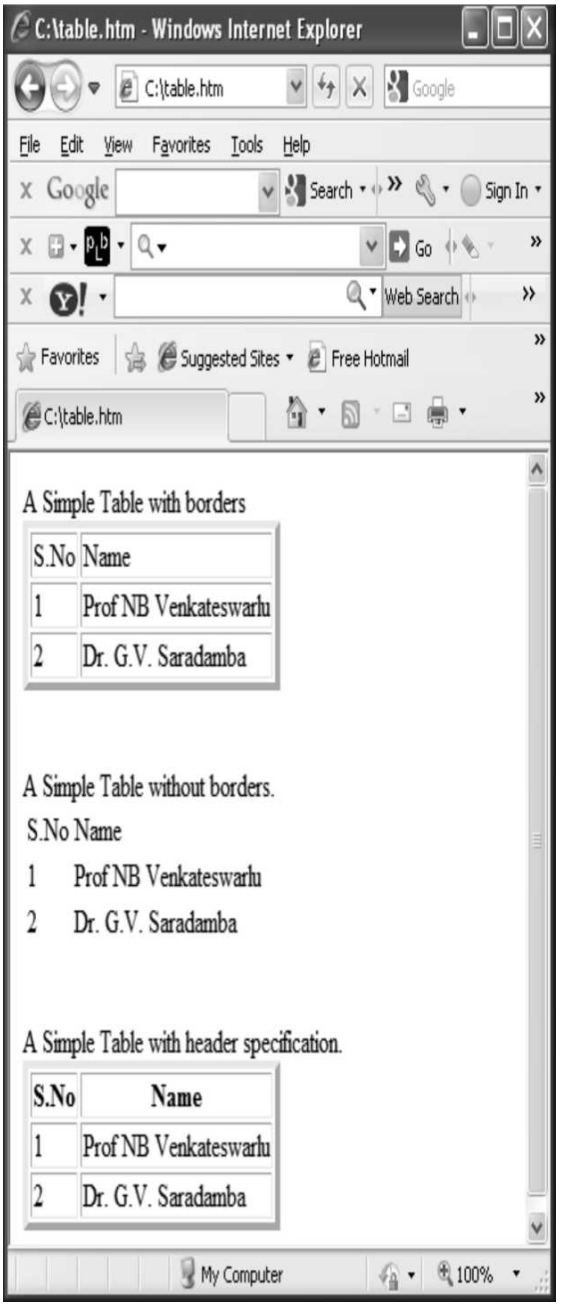


Figure 8.11 A web page tables

8.1.26.1 Table Tags

The following table lists all the possible tags that can be used in creating tables.

Tag	Description
<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines groups of table columns

Tag	Description
<col>	Defines the attribute values for one or more columns in a table
<thead>	Defines a table head
<tbody>	Defines a table body
<tfoot>	Defines a table footer

8.1.27 How to make Forms?

We are often required to create pages which viewers are required to fill. For this purpose forms are used. A form is an area that can contain form elements or fields. We have fields such as text field, text area, password field, submit/reset button etc., in addition to combo box. A typical form is defined with the <form> tag.

```
<form>
.
input elements
.
</form>
```

8.1.27.1 Input

The most used form tag is the <input> tag. The type of input is specified with the type attribute. The most commonly used input types are explained below.

8.1.27.2 Text Fields

Text fields are used when we want the user to type letters, numbers, etc. in a form.

```
<form>
First name:
<input type="text" name="firstname" />
<br />
Last name:
<input type="text" name="lastname" />
</form>
```

The above code generates a page in a browser that looks like:

First name : [ ]

Last name : [ ]

Note that the form itself is not visible. Also note that in most browsers, the width of the text field is 20 characters by default. We can change the size by size attribute of input tag.

8.1.27.3 Radio Buttons

Radio Buttons are used when we want the user to select one of a limited number of choices.

```
<form>
<input type="radio" name="sex" value="male" /> Male
<br />
<input type="radio" name="sex" value="female" /> Female
</form>
```

How it looks in a browser:

- Male
- Female

Note that only one option can be chosen.

8.1.27.4 Checkboxes

Checkboxes are used when we want the user to select one or more options of a limited number of choices.

```
<form>
I have a bike:
<input type="checkbox" name="vehicle" value="Bike" />
<br                                                                    />
I have a car:
<input type="checkbox" name="vehicle" value="Car" />
<br />
I have an airplane:
<input type="checkbox" name="vehicle" value="Airplane" />
</form>
```

8.1.27.5 The Form's Action Attribute and the Submit Button

When the user clicks on the "Submit" button, the content of the form is sent to the web server. The form's action attribute defines the name of the program to which we have to send the filled content. The file defined in the action attribute usually does something with the received input.

```
<form name="input" action="html_form_submit.asp" method="get">
Username:
<input type="text" name="user" />
<input type="submit" value="Submit" />
</form>
```

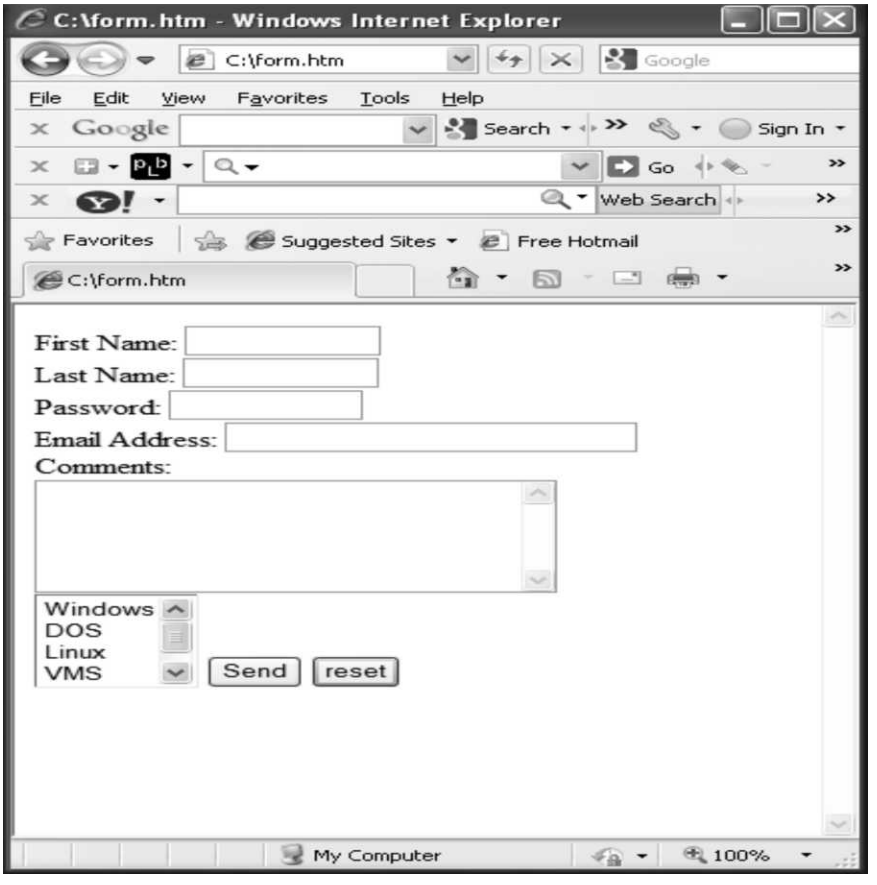


Figure 8.12 A web page with Form Fields

If we type some characters in the text field above, and click the "Submit" button, the browser will send our input to a page called "html_form_submit.asp". The page will show us the received input.
The following sample.html file with <form> and </form> tags will give a form as shown in the following Figure (Figire 8.12).

```
<html><body>
<form>
First Name: <input name="first" type="text" size=12><br>
Last Name: <input name="last" type="text" size=12><br>
Password: <input name="password" type="password" size=12><br>
Email Address: <input name="address"  type= "text"size=30><br>
Comments: <br>
<textarea name="whatever" rows=5 cols=30></textarea>
<br>
<select name="whatever" size=4> <option>Windows
<option>DOS <option>Linux <option>VMS <option>Unix
</select>
<input type="Submit" value="Send"> <input type="reset"  value="reset">
</form>
</body></html>
```

We can fill the fields as shown in Figure 8.13. If we press reset button, whatever we have typed will be cleared. If we press SEND button (submit button), the screen looks like Figure 8.14.

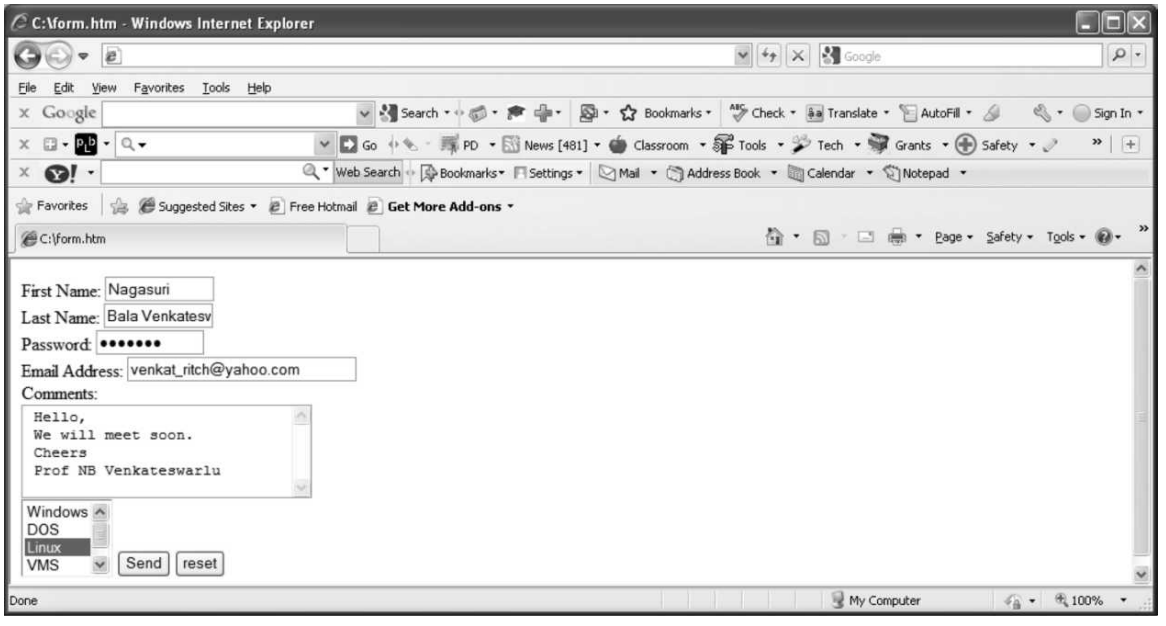


Figure 8.13 A web page with Filled Fields

The form data is usually used by either applets or server side program such as servlets, perl scripts, PHP scripts etc. The browser supplies it to the respective program like variable=value pairs separated by & See the encircled portion of the following Figure (Figure 8.14). Here, the variable is the one which is defined as name="" in form element fields. For example, in the above web page we have defined text field for First Name as "first", Last Name as "last", etc. Thus, once we press the submit button (Send button in this case), this variable and the value typed in the field are sent to the server side programs. More about these concepts can be found in any book on server side scripts.

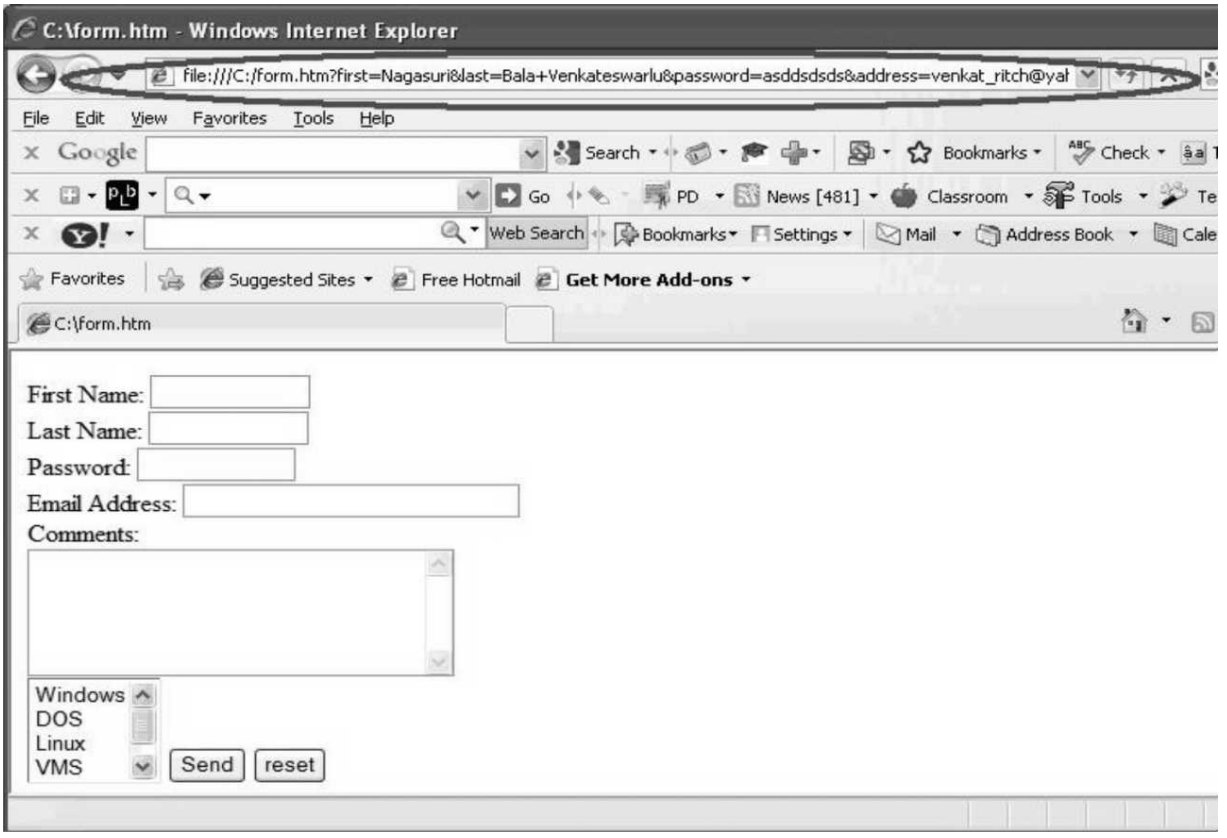


Figure 8.14 A web page that shows effect of submit button

8.1.28 How to make a TOC (Table of Contents)?

To have TOC, first we have to have sections all over our web page. To display TOC:

```
<a name="toc">the name of your table of contents goes here </a>
```

We can also specify the TOC location, at the top or the bottom. To make a link back to the TOC, we can write:

```
<a href="#toc">To the top</a>
```

The rest of the sections follow the same pattern:

```
<a href="#sec1">go to section one </a>
```

That is a link to a place called "sec1", which can be created as :

```
<a name="sec1">This is section one </a>
```

To sum it all, sections can be created by using the & tags. Links to them can be created by making & tags.

8.1.29 How to make Frames?

If we want to display more than one web page on the screen, we use frames. Making frames is slightly complicated but not quite as difficult as making forms. When we make a page with frames we actually don't have anything on that page. It's a kind of a reference page.

Here, we use FRAMESET tag which is a frame container for dividing a window into rectangular sub-spaces called frames. The FRAMESET element contains one or more FRAMESET or FRAME elements, along with an optional NOFRAMES element to provide alternate content for browsers that do not support frames or have frames disabled. A meaningful NOFRAMES element should always be provided and should at the very least contain links to the main frame or frames.

The ROWS and COLS attributes of frameset tag define the dimensions of each frame in the set. Each attribute takes a comma-separated list of lengths, specified in pixels, as a percentage, or as a relative length. A relative length is expressed as i* where i is an integer. For example, a frameset defined with ROWS="4*,*" (* is equivalent to 1*) will have its first row allotted four times the height of the second row. The values specified for the ROWS attribute give the height of each row, from top to bottom. The COLS attribute gives the width of each column from left to right. If ROWS or COLS is omitted, the implied value for the attribute is 100%. If both attributes are specified, a grid is defined and filled left-to-right then top-to-bottom.

The following example sets up a grid with two rows and three columns:
<FRAMESET ROWS="70%,30%" COLS="33%,33%,34%">
The next example features nested FRAMESET elements to define two frames in the first row and one frame in the second row. Second row is of 100 pixels width while the remaining is for the first row. The first row is divided into two columns such that 40% of the column is for the first frame and remaining for the second frame.

```
<FRAMESET ROWS="*,100">
<FRAMESET COLS="40%,*">
```

Here is an example:

```
<FRAMESET>
<FRAMESET COLS="50%,50%">
<frameset rows="50%,50%">
<FRAME SRC="http://www.ritchcenter.com/index.htm ">
<FRAME SRC="http://www.ritchcenter.com/index.htm ">
</frameset>
<frameset rows=50%,50%">
<frame src="http://www.ritchcenter.com/index.htm">
<frame src="http://www.ritchcenter.com/index.htm ">
</FRAMESET>
</FRAMESET>
```

The first half up to the first </frameset> tag defines what goes on the left side and everything after that defines what is on the right side. The <frame src=""> tags tell what is in each frame. In this example, there are 4 frames so there are four <frame src=""> tags. In the <frame src=""> tags, we would fill in whatever the URL is of what we want in that frame. The page created with these frames is actually four different documents of HTML. If we open the above html file, we will get a page which looks like Figure 8.15.

Do remember that HTML5.0 does not support framesets.

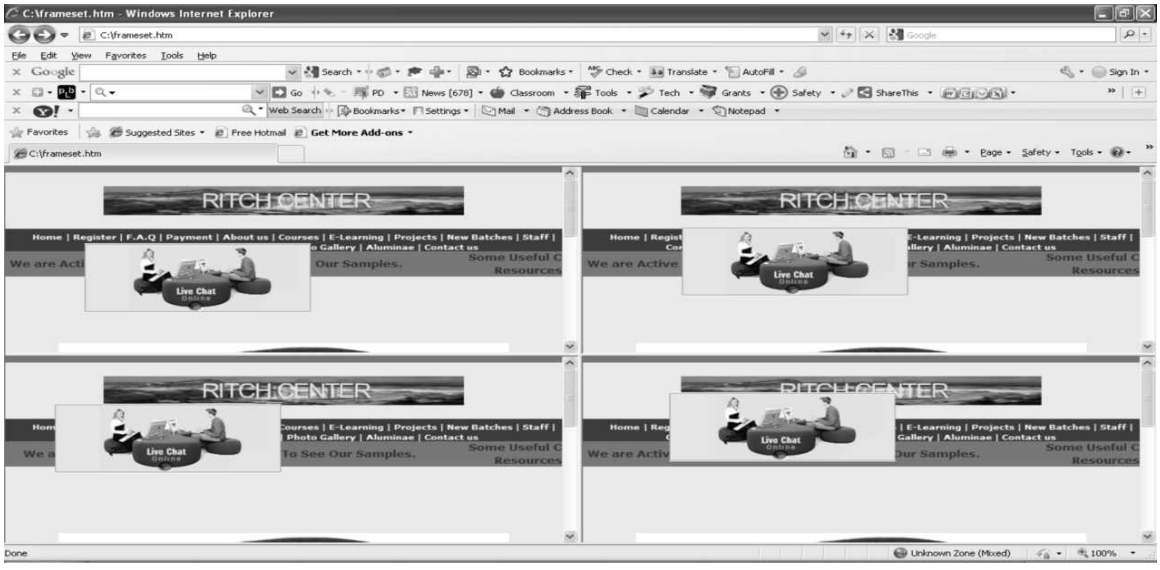


Figure 8.15 A web page with Four Frames

8.1.30 How to make Background Music to Play?

To add background music to our site, we can use:

```
<CENTER><DD><EMBED SRC="blah.mid" WIDTH=0 HEIGHT=0 AUTOSTART=TRUE>
<HR width="100%"></DD></CENTER>
```

The sound files can be midi type, au type or wav type. If they are available in the same directory of the html file, we can simply enter their names in place of blah.mid. If the sound file is on another machine, we can use URL of that sound file. We can also achieve the same using the following tag.

```
<bgsound src="path to your sound file" loop=infinite>
```

8.1.31 The APPLET HTML Tag

In many front-end applications, Java applets are included on web pages using the <APPLET> tag. The <APPLET> tag is most similar to the <IMG> tag. Like <IMG> tag, <APPLET> tag needs to reference a source file that is not part of the HTML page on which it is embedded. IMG's do this with the SRC= parameter. APPLET's do this with the CODE= parameter. The CODE parameter tells the browser where to look for the compiled Java.class file. It is relative to the location of the source document. Thus if we are browsing http://metalab.unc.edu/javafaq/index.html and that page references an applet with CODE=animation.class, then the animation.class file needs to be at http://metalab.unc.edu/javafaq/animation.class.

For reasons that remain a mystery to HTML authors everywhere, but possibly has something to do with packages and classpaths, if the applet resides somewhere other than the same directory as the page it lives on, then we don't have to give a URL to its location. Rather, we point at the directory from where the .class file is using the CODEBASE parameter. We still have to use CODE to give the name of the .class file. Also like <IMG> tag, <APPLET> tag has several parameters to define how it is positioned on the page. HEIGHT and WIDTH parameters work exactly as they do with <IMG>, specifying how big a rectangle the browser should leave for the applet. These numbers are specified in pixels. ALIGN attribute also works exactly as for images (in those browsers that support ALIGN) defining how the applet's rectangle is placed on the page relative to other elements. Possible values include LEFT, RIGHT, TOP, TEXTTOP, MIDDLE, ABSMIDDLE, BASELINE, BOTTOM and ABSBOTTOM. Finally as with IMG you can specify an HSPACE and a VSPACE in pixels to set the amount of blank space between an applet and the surrounding text.

QUESTIONS

1. Is the output of the following <pre> block and <code> block same?

```
<pre>
#include<stdio.h>
int main()
{
return 0;
}
</pre>
<code>
#include<stdio.h>
int main()
{
return 0;
}
</code>
```

Answer: No. The <code> block gives a single line output in the browser.

2. It stdio.h displayed in the browser?

```
<pre>
#include<stdio.h>
</pre>
```

Answer: No

3. Is the output of the following <samp> block and <code> block same?

```
<samp>
#include<stdio.h>
int main()
{
return 0;
}
</samp>
<code>
#include<stdio.h>
int main()
{
```

```
return 0;
}
</code>
```

Answer: Yes. Both give a single line output in the browser.

OBJECTIVE TYPE QUESTIONS

1. Is HTML language a compiled languages? (Y/N)
2. The browser displays elements according to HTML tags. (Y/N)
3. One can make a portion of an image to be ready for click using maptag. (Y/N)
4. The browser can open a plain text file also. (Y/N)
5. When we open a Java language source file using a Web browser, it runs. (Y/N)
6. When we open a Java Applet Class file using a HTML file (using APPLET tag), the program runs. (Y/N)
7. Browsers can send HTTP requests to Web servers. (Y/N)
8. The <pre> tag can be used to display verbatim. (Y/N)
9. Java applications (program with main method) have to be executed using browsers only. (Y/N)
10. We cannot change page look dynamically using static HTML tags. (Y/N)
11. __ attribute is used to specify text field width
    A. length B. size
    C. len D. None
12. Paragraph tag
    A. <para>
    B. <format align="justified">
    C. <p>
    D. None
13. Find the odd man out
    A. B.
    C. D. <du>
14. First tag in a typical HTML document
    A. <body> B. <title>
    C. <html> D. <head>
15. A tag unrelated to <td>
    A. <table> B. <du>
    C. <tr> D. <th>
16. Tag related to largest size heading
    A. <font size="huge"> B. <h1>
    C. <h7> D. <large-font>
17. To specify the background color of a page
    A. <background>"Yellow"</background>
    B. <bg>"Yellow</bg>
    C. <body background="yellow">
    D. <body bgcolor="yellow">
18. Find the odd man out
    A. http B. html C. ftp D. mailto
19. We cannot create an URL with
    A. http B. html C. ftp D. mailto
20. The tag which is used to display verbatim
    A. <pr> B. <p>
    C. <caption> D. <pre>
21. Is it possible to have multiple <body> tags?(Y/N)
22. The tag specifies the base URL/target for all relative URLs in a document.
    A. <rel> B. <link>
    C. <base> D. <relative-link>
23. The <base> tag goes inside the ___ tag.
    A. <title> B. <head>
    C. <body> D. <document>
24. The base URL should be an absolute URL. (Y/N)
25. The tag that makes a portion of an image ready for click is
    A. <click> B. <clickedimage>
    C. <img> D. <map>
26. The tags that are mostly associated with <form>
    A. <post> B. <get>
    C. <both> D. <action>
27. With which tag multiple is used
    A. <list> B. <menu>
    C. <select> D. <none>
28. ___ attribute is used to specify number of visible options
    A. size B. one
    C. two D. multiple
29. Drop down list related tag
    A. <list> B.
    C. <select> D.
30. It is not possible to have subscripts and superscripts in HTML. (Y/N)
31. Find the tag related to <tbody>, <tfoot>, <thead>
    A. <table> B. <column>
    C. <row> D. <body>
32. Find the odd man out of the following in relation to scrollable tables
    A. <tbody> B. <tfoot>
    C. <thead> D. <head>

33. Both <area> and <textarea> are related to font. (Y/N)

34. To create an e-mail in our contact details

A. <mail:nbv@xyz.com>

B. <a href=mailto:nbv@xyz.xom>

C. <mailto:nbv@xyz.com>

D. None

35. Find the odd man out

A. _self B. _blank

C. _parent D. None

36. To specify the relationship between the current document and the linked document in anchor tag

A. Rev B. Rel

C. Relative D. None

37. HTML enables us to manage web document's

A. Presentation

B. Content

C. Both presentation and content

D. Neither presentation nor content

38. The class attribute in HTML is used

A. to a apply some style to some elements

B. to embed some program into HTML

C. to specify the quality of a document

D. None

ANSWER KEY

1. N	**2.** Y	**3.** Y	**4.** Y
5. N	**6.** Y	**7.** Y	**8.** Y
9. N	**10.** Y	**11.** B	**12.** C
13. D	**14.** D	**15.** B	**16.** B
17. D	**18.** B	**19.** B	**20.** A
21. Y	**22.** C	**23.** B	**24.** N
25. D	**26.** D	**27.** C	**28.** A
29. C	**30.** N	**31.** A	**32.** D
33. Y	**34.** B	**35.** D	**36.** B
37. C	**38.** A		

8.2 Cascading Style Sheets (CSS): Introduction

HTML tags were originally designed to define the content of a document. They are basically designed to say "This is a header", "This is a paragraph", "This is a table", by using tags like <h1>, <p>, <table>, and so on. The layout of the document was supposed to be taken care by the browser, without using any formatting tags. Thus, the look and feel of pages developed using HTML used to be different in different browsers. It became more and more difficult to create web sites where the content of HTML documents was clearly separated from the document's presentation layout. To solve this problem, the World Wide Web Consortium (W3C, a non-profit, standard setting consortium, responsible for standardizing HTML) created STYLES (Cascading Style Sheets) in addition to HTML. As of now, all major browsers support Cascading Style Sheets.

Main attraction of CSS is that the Style Sheets can save a lot of our labor. Styles sheets define how HTML elements are to be displayed, just like the font tag and the color attribute in HTML. Styles sheets enable us to change the appearance and layout of all the pages in our site, just by editing one single CSS document.

8.2.1 Why do we use Style Sheets?

- Much better control over font face, font size, colors, backgrounds, and many other elements of pages.
- Ability to make changes in one location that can apply to many, possibly hundreds of pages in a site.
- Less code, smaller pages, faster downloads.

8.2.2 Why can't we do that in HTML?

- HTML is not designed to control appearance, but for the overall structure of a web page.
- Later versions of HTML added some formatting elements such as font tags, color, size, etc. However, people identified the need for separation of style from display.

8.2.3 Separation of Content and Style

- HTML documents contain the structure of the document.
- CSS contains all information related to how the document displays in browser.
- Separation is desirable for maintenance reasons, Search engines, and display of same content on different platforms.

8.2.4 What is a Style Sheet?

- Text file with style definitions.

8.2.5 How is Style applied?

- Web page is made of elements (paragraphs, headings, links, div's, images, etc.).
- Style is applied to elements.
- Individual elements, pagewide elements, or classes of elements.

8.2.6 Brief Introduction to CSS

Cascading Style Sheets (CSS) allow the web page designer to define HTML elements. This amount of control over the rendering of the page allows a great deal of freedom in determining what our page will look like. For example, take a tag <h1> and give it new attributes like highlighting text with red. Whenever we want text to be highlighted red, all we have to do is use the <h1> tag instead of using . Since CSS allows the designer to separate style from content, this saves a lot of time when editing pages and makes HTML code easier to read.

8.2.6.1 Syntax

CSS syntax is easy to remember: **selector, property** and **value**. The **selector** is the HTML tag which we want to modify, **property** is an attribute which we want to modify with a required **value**.

■ **Example**

```
p {font-family: "verdana, arial, helvetica"}
```

Here, p indicates that the <p> tag is our selector, "font-family" is the property that will be modified by the value "verdana, arial, helvetica" . The property and value are separated by a colon, and surrounded by curly braces. For example, the following makes the body black.

```
body {color: black}
```

Usually value is assigned directly. If the value is multiple words, put quotes around the value like "verdana, arial, helvetica". If we wish to specify more than one property, we must separate each property with a semicolon. The example below shows how to define a center aligned paragraph, with a red text color:

```
p {text-align:center;color:red}
```

We can also make style definitions easier to read by spacing them out and writing each property on a different line:

```
span
{
color:red
font-style: italic
}
```

When modifying fonts using CSS, there are units and values that should be kept in mind:

- em – height of a character
- px – pixels
- pt – point
- % - percentages

```
body { font-size:10px }
```

8.2.6.2 Grouping

We can group selectors. Separate each selector with a comma. In the example below we have grouped all the header elements. All header elements will be displayed in green text color:

```
h1,h2,h3,h4,h5,h6
{
color: green
}
```

8.2.6.3 Custom Selectors

Besides selecting HTML elements to apply styles to, we can also create our own custom element names to apply to any element. Custom styles take two forms, CLASS and ID.

8.2.6.4 When should I use ID or CLASS?

- **CLASS** styles can be attached to multiple elements
- **ID** styles can only be attached to one element.
- Use ID when there is only ONE instance. Use Class when there are multiple instances.

The syntax for both is as follows:

CLASS

```
.somename { color:green }
```

ID

```
#someothername { color:red }
```

ID must be preceeded by a hash "#".

```
p#ID1 { background-color:blue }
```

CLASS must be preceded by a period.

```
p.redpen {color:red}
p.greenpen {color:green}
```

Whenever we want to "highlight" text in a certain color:

```
<p class = "redpen">
This text is in red.
</p>
<p class = "greenpen">
The text is in green.
</p>
```

8.2.6.5 The Class Selector

With the class selector, we can define different styles for the same type of HTML element.
For example, if we would like to have two types of paragraphs in our document: one right-aligned paragraph, and one center-aligned paragraph. Here is how we can do it with styles:

```
p.right {text-align: right}
p.center {text-align: center}
```

We have to use the class attribute in our HTML document:

```
<p class="right">
This paragraph will be right-aligned.
</p>
<p class="center">
```

```
This paragraph will be center-aligned.
</p>
```

We can also omit the tag name in the selector to define a style that will be used by all HTML elements that have a certain class. In the example below, all HTML elements with class="center" will be center-aligned:

```
.center {text-align: center}
```

In the code below both the h1 element and the p element have class="center". This means that both elements will follow the rules in the ".center" selector:

```
<h1 class="center">
```

This heading will be center-aligned

```
</h1>
<p class="center">
```

This paragraph will also be center-aligned.

```
</p>
```

8.2.6.6 The id Selector

We can also define styles for HTML elements with the id selector. The id selector is defined as a #. The style rule below will match the element that has an id attribute with a value of "green":

```
#green {color: green}
```

The style rule below will match the p element that has an id with a value of "para1": p#para1

```
{
text-align: center;
color: red
}
```

8.2.6.7 CSS Comments

Comments are used to explain our code, and may help us when we edit the source code at a later date. A comment will be ignored by browsers. Like C language comment, a CSS comment begins with "/*", and ends with "*/", like this:

```
/* This is a comment */
p
{
text-align: center;
/* This is another comment */
color: black;
font-family: arial
}
```

8.2.7 Adding to a HTML Page

There are 3 ways to use CSS with our HTML pages. They are: external, internal, inline.

8.2.7.1 External

This is the most common implementation of styles. The CSS code is kept in a separate file with ".css" extension. A snippet is added to the <head> section of the HTML file specifying where the style sheet is. For example, if we have a style sheet called main.css in our styles folder under our web directory, our HTML pages <head> block is written as:

```
<head>
<link rel="stylesheet" type="text/css" href="./styles/main.css" />
</head>
```

For example, if we want to set the background color of our pages to yellow, the CSS file main.css should have the following line.

```
body {background-color: yellow}
```

Using external style sheets, it is easy to apply the style sheet to multiple pages. Any changes we make to the source style sheet, *cascades* and updates the styling of all our web pages.

8.2.7.2 Internal/Embedded

Let us assume that we plan to apply an external style sheet to our whole set of web pages, except one page with blue background. This can be done by including the page specific CSS code within the <head> section of that page which overrides an internal CSS instructions. While other styles of our external style sheet come through, the background color style of the external sheet will be overridden by the internal style sheet in the page. Evidently, the CSS code needs to be wrapped with special <STYLE> tags in that HTML page.

```
<head>
<style type="text/css">
<!-- body { background-color: blue;}-->
</style>
</head>
```

The use of comments within the style tag is to hide the code from someone viewing the page with a really old browser.

8.2.7.3 Inline

Inline uses of CSS is generally not recommended and is slowly being faded out. Inline CSS is where we stick the style directly inside a HTML tag. For example:

```
<p style="color:green">
The text in this paragraph would then be green.
</p>
```

The only time we should use Inline CSS is if we need one instance of CSS, say highlighting a sentence or something that would be difficult to do with other HTML methods. We can use more than one of these implementations. When they conflict, the order of precedence is:

1. **Inline styles**
2. **Internal styles**
3. **External styles**

8.2.8 Div and Span

The tag is used to divide portions of a web page and allows us to define a style section. to is used to indicate the beginning to the end of a paragraph. Remember that we cannot have a within a
 is used to tell the browser to apply formatting. The big difference between and is 's ability to create paragraph breaks (line break before and after.) elements only affect a small chunk of text in a line.
 can be nested within a .

```
<div class="blueback">
<span class="sitaround">Sit around in center! </span>
<span class="floatright">I am floating to the right! </span>
<span class="other">Other text</span>
</div>
```

The width of a can be set, and the importance of this will be discussed later.

```
.box {width = 10px;}
```

8.2.8.1 Links Color Change

A fun thing to do is to make our web links change to a different color when the mouse cursors hover over the link. In this example, "a" is the "a" in (anchor tag) and we can apply the following style changes:

```
a:link { text-decoration: none; color: #33CCFF; }
a:visited { text-decoration: none; color: #33CCFF; }
a:hover { text-decoration: underline; color: #FF0000; }
```

Now, the link should be displayed in light blue and once the user hovers over the link, it becomes underlined and in red font.

8.2.9 Building a Site without Tables

The ability to just "layout" a page without tables is one of CSS's strongest points. There is no need to keep track of millions of nested tables and tags.

8.2.9.1 Position

This property allows the coder to determine where a block of text will go in the page.

- Static – places the block wherever it is
- Absolute – places the block in the page defined by the coder
  .somewhere {position: absolute; top: 50px; right: 100px; }
  (This places the block of text 50 pixels from the top, and 100 pixels from the right)
- Relative – places the block where it would have been if there was nothing around.
  .shift {position: relative; top: 12px; right: 10px; }

8.2.9.2 Border

This is the standard property that allows us to draw lines around blocks of text. There are many border styles, ranging from solid to hidden.

```
.a1 {border-style: solid; } /* Your standard black border */
.a2 {border-style: double; } /* Double border*/
.a3 {border-style: hidden; } /* Hidden! */
.a4 {border-style: inset; } /* Creates an indented border */
.a5 {border-style: outset; } /* Creates a raised border */
.a6 {border-style: groove; } /* Creates a grooved border */
```

For the ubiquitous thin border, we can use this code:

```
border: 1px solid #000000;
```

Up to this point, a simple two-column page can be created using the following code:

```
#navbar { position: absolute;
          top: 10px;
          left: 2px;
          width: 200px;
           border: 1px solid #000000;
}
#maintext { position: absolute;
            top: 10px;
           left: 210px;
            border: 1px solid #000000;}
          }
```

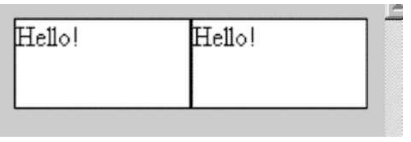
Adding a third column is very easy, just create a new ID with the position of the third area. Use the tags in the <body> region of the HTML code (remember that tags are like blocks of text) and you're done!

8.2.9.3 Float

This property "floats" a block of text or image in a direction (left or right or nowhere.)

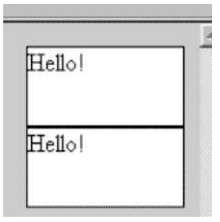
```
#flt_right {float: right;} /* self-explanatory */
```

Multiple blocks with the same float direction will appear alongside each other. To create the effect of blocks stacked on top of each other, but still floating towards a direction, use the CLEAR property.
Without clear:




```
#flt_rightclear { float: right;
clear:right; }
```

With clear:



■ **Exercise** Explain what happens if we have external and internal style sheets for the h2 selector as given below.
An external style sheet has these properties for the h2 selector:

```
h2
{
color: red;
text-align: left;
font-size: 8pt
}
```

and an internal style sheet has these properties for the h2 selector:

```
h2
{
text-align: right;
font-size: 20pt
}
```

■ **Answer:** If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet. The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.
Thus, the properties for h2 will be:

```
color: red;
text-align: right;
font-size: 20pt
```

QUESTIONS

1. **What will be the implication of the following CSS elements?**

```
TABLE.main { margin-left:-5px; margin-top:-10px;}
TH { font-size:16px; font-style:bold; font-weight:bold; font-family:
helvetica, sans-serif,serif; }
TD { font-size:16px; font-family: helvetica, sans-serif,serif; }
```

■ **Answer:** It defines table properties such as left and top margins; Specifies the table heading as 16 points bold Helvetica font while table data items as simple Helvetica font.

2. **What will be the effect of the CSS elements?**

```
A { text-decoration: none; }
A:link { color: green; }
A:visited { color: brown; }
A:hover { color: red; }
```

■ **Answer:** This CSS element uses anchor tag. It indicates the anchor tag (links) to be underlined and in green colour. When we place the mouse over the links, color to be changed to red. While the visited links to be shown in brown color.

3. **What will be the effect of the CSS element?**

```
ul {
        list-style: square;
        margin-top: 40px;
        margin-left: 40px;
}
```

■ **Answer:** This defines characteristics of unordered lists as square type, and top and left margins as 40 pixels.

OBJECTIVE TYPE QUESTIONS

1. External css can be referred in a HTML document by using
   A. <stylesheet>nbv.css</stylesheet>
   B. <style src="nbv.css">
   C. <link rel="stylesheet" type="text/css" href="nbv.css">
   D. None
2. In which section of a HTML document css file is specified
   A. head B. title
   C. body D. None
3. _____ tag is used to specify internal style sheet in a HTML document.
   A. link B. style
   C. css D. None
4. Correct way of specifying body color as black in CSS file is_____
   A. bodycolor=black
   B. bgcolor=black
   C. body{color:black}
   D. None
5. To add background color for all h1 type elements
   A. h1.all{background=#FFFAB}
   B. all.h1{background=#FFFFAB}
   C. h1{background=#FFFFAB}
   D. h1{background-color=#FFFFAB}
6. To change text color, ___ tag can be used in CSS.
   A. text-color B. text color
   C. color D. None
7. __ is used to control text size.
   A. text-size B. textsize
   C. font-size D. None
8. To make paragraphs as bold,
   A. <p style="text-color=bold">
   B. <p style="font-size=bold">
   C. <p style="font-size:bold">
   D. None
9. What is the use of CSS element

```
A:visited { color: brown; }?
```

   A. To change the paragraph color when mouse is positioned
   B. To change the visited links as brown
   C. To change hyper links to brown
   D. None

ANSWER KEY

1. C	**2.** A	**3.** B	**4.** C
5. C	**6.** C	**7.** C	**8.** C
9. B			

8.3 A Simple Introduction to XML

In the real world, computer systems and databases contain data in incompatible formats. One of the most time-consuming challenges for developers is how to exchange data between such systems over the Internet. Converting the data to XML (**Extensible Markup Language**) can greatly reduce this complexity and create data that can be read by many different types of applications.

XML, like **HTML,** is also a markup language that provides access to structured content. **XML** documents can be shared across applications, file systems, and operating systems. Thus, **XML is a cross-platform, software and hardware independent tool for transmitting information. HTML** contains a set of pre-defined tags of HTML that are aimed at giving display instructions to a browser. Instead, XML allows us to define our own elements to represent the data that is required to be shared across variety of platforms.

On what platforms XML can be used?

Some of them include PCs, PDAs, cell phones, WebTV, and many emerging technologies. Applications include spreadsheets, PHP, and, ASP, and PDF.

How does XML relate to HTML?

It augments HTML by focusing on what the data is. HTML is a markup language that dictates *how* the text looks in the browser (thus for formatting and layout). XML describes the data, and how it is organized. Like HTML, the XML source is hidden the computer knows what the data means:

```
HTML→   <b><i>Lord Balaj</i></b>
XML→    <mydata>
        <name>Lord Balaji</name>
        <city>Tirupathi</city>
        </mydata>
```

Evidently, HTML has a finite set of pre-defined tags; whereas in XML, we can create our own tags. Therefore, it describes the data like fields in a database.

XML Separates Data from HTML

If we need to display dynamic data in our HTML document, it demands lot of efforts to edit the HTML each time the data changes. With XML, data can be stored in separate XML files. This way one can concentrate on using HTML for layout and display, and be sure that changes in the underlying data will not require any changes to the HTML document. With a few lines of JavaScript, we can read an external XML file and update the data content of our HTML.

XML does not do Anything

Maybe it is a little hard to digest, but XML does not **do** anything. XML is created to structure, store, and transport information. The following example is a note to Venkat from Sarada, stored as XML:

```
<?xml version = "1.0" ?>
<note>
<to>Venkat</to>
<from>Sarada</from>
<heading>Reminder</heading>
<body>Don't forget our meeting next  weekend!</body>
</note>
```

The note above is quite self descriptive. It has sender and receiver information; it also has a heading and a message body. But still, this XML document does not DO anything. It is just information wrapped in tags. Someone must write a piece of software to send, receive or display it.

XML Simplifies Data Sharing

In the real world, computer systems and databases contain data in incompatible formats. XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data. This makes it much easier to create data that different applications can share.

XML Simplifies Data Transport

With XML, data can easily be exchanged between incompatible systems.

XML Tools

An **XML parser** is an application that can read and interpret both the data and processing instructions in an XML document.

- One can view XML files using the MSXML Parser (version 2.0) built into Internet Explorer.
- XML documents can be used by other types of applications such as Window Forms. These applications access the data in the XML document using the XML Document Object Model (DOM) and the Simple API for XML (SAX).

We can create the XML pages using:

- simple text editor such as Notepad, wordpad, etc.
- HTML/XML editor provided by Visual Studio.NET
- Microsoft XML NotePad called xmlpad.exe

In general, file extensions for XML related pages are summarized as:

- Generally saved with the file extension .**xml**.
- Some applications allow us to save a set of data from a database as an XML document with the file extension .**XSD**.

8.3.1 XML Coding Hints

An XML document must be **well-formed**. A well-formed document follows XML standards and can therefore be read by any XML parser.

- There should be only one root element under which all other elements have to be nested.
- Cannot mix nesting elements like the following
  - Welcome to Tara Store
- Enclose the values of properties within double quotation marks.
- XML is case sensitive. The case for the opening and closing tags of an element must match.

We can validate our XML code to ensure that it is well-formed. There are many online tools available to helf you. For instance, Microsoft has a free validation tool located at http://Msdn.Microsoft.com/downloads/samples/Internet/xml/xml_validator/sample.asp.

An XML document has two parts: **Prologue** and **Body**

8.3.1.1 The Prologue

This section contains global information such as the XML version, formatting information, and schema definitions. For instance, see the following simple prologue statement.

```
<?xml version="1.0" encoding="utf-8" ?>
```

The question mark indicates that this tag, XML, is a processing instruction and therefore does not contain data. Character encoding property describes any coding algorithms that are used within the page such as UTF-8 or UTF-16.

We can add a reference to a CSS style sheet in an XML document to an external Cascading Style Sheet (CSS) or an XSL file to format the XML document. We can also use an Extensible Style Language (XSL) style sheet to process & format XML documents. XSLT is a form of XSL that transforms the XML data using processing rules in the XSLT style sheet. The following is a sample prologue that indicates the use of style sheets.

```
<?xml:stylesheet  type="text/css" href="taragifts.css">
```

8.3.1.2 The Body

The XML document complies with the XML DOM standards. The XML DOM was influenced by the document object models that were used by browsers. The XML DOM states that XML documents must have a logical structure.

- The body of the XML file contains the elements, attributes, and data in the XML document.
- The root container element (node) must nest all the elements and data.
  - The root node can contain many other elements.
  - All tags must be nested within the root node (or root tag).
- A container element is an element that can nest other elements.
  - The root node is a container element because all elements must be nested within the root element in an XML document.

Comparison with an HTML document

- In an HTML page, the HTML tag is the root element.
- The HTML element is a container element because it contains child elements such as <head></head> and <body></body>.
- In this case, the HTML element is also the parent element for the head and body tags.
- In an HTML page, the <title> tag is nested within the <head> tag. The <head> tag is a container element, and the parent element for the <title> tag.
- The head and body tags are referred to as the child elements.

In the sample code below, productlist is the root node. In this sample, the productlist node contains two product nodes.

<productlist>

<product></product>

<product></product>

</productlist>

We can create elements that are containers for other elements.

- productlist element is the root element.
- product elements contained within the productlist root element.
- product element is a container element for code, name, price, category, image, and rating elements.

```
<productlist>
<product>
  <code>1221-2212-22</code>
  <name>Wipro</name>
  <price>99.00</price>
</product>
<product>
  <code>1221-22-122-21</code>
  <name>Birla </name>
  <price>59.00</price>
</product>
</productlist>
```

The following is a sample XML file with the name **A.xml** which when opened in IE browser, will display as shown in Figure 8.16.

```
<?xml version='1.0' ?>
```

```
<document>
<body>
   <warning>Beware of Compiling Errors.</warning>
However,
   <tip> See the error messages keenly for tips </tip>
   and <resource>use resources in Internet for help</resource>
</body>
</document>
```

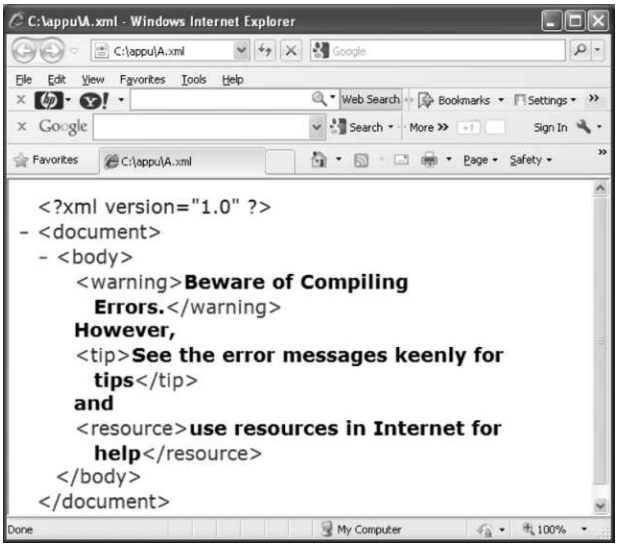


Figure 8.16 A simple XML output

8.3.1.3 Unicode

Like Java, XML uses Unicode to code for character data. There are a number of different versions of Unicode, but all have ASCII as the first 128 characters. The most common version used in the west, and the XML default, is UTF-8.

8.3.1.4 Declarations

XML documents do not require a declaration at the top, but it is always a good idea to put one there. The simplest declaration is the one used in A. XML:

<?xml version = "1.0" ?>

To this we can add two attributes. These are encoding and standalone. The result might be

<?xml version="1.0" encoding="UTF-8" standalone ="no"?>

The meaning of encoding was given above (refer to 8.3.1.3 Unicode). Standalone refers to whether or not the document has a DTD (Document Type Definition) included in-line.

There are many predefined declarations in use. The one that follows has been provided by the Apache Tomcat project to be used in configuring the *web application deployment descriptor*, web.xml. The encoding here is ISO-8859-1, known as Latin-1. ISO stands for International Standards Organization. The 8859 standard contains a number of encodings. Latin-1 is the only one that is the same as UTF-8 in the first 256 characters.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/dtd/web-app_2_3.dtd">
```

The DOCTYPE declaration is similar to the one used in xhtml. But this one refers to a DTD that has been created for Tomcat, version 4. This is the only content of the web.xml file, stored in the WEB-INF folder. A somewhat longer version of web.xml is included with Tomcat version 5. It uses an XML Schema. Both DTDs and Schema will be discussed later.

We can also specify other media such as images, sound, Braille, etc. with the @media directive. For example, a style sheet could have this coding:

```
@media aural{
note[type="tip"] {volume:medium; voice-family:FriendlyAdvice;}
```

8.3.1.5 All XML Elements must have a Closing Tag

In HTML some elements do not necesserily have a closing tag. The following code is legal in HTML:

```
<p>This is a paragraph
```

In XML all elements must have a closing tag like this:

```
<p>This is a paragraph</p>
```

8.3.1.6 XML Tags are Case Sensitive

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>. Opening and closing tags must therefore be written with the same case:

<Message>This is incorrect</message>

<Message>This is correct</Message>

8.3.1.7 All XML Elements must be properly Nested

In HTML some elements can be improperly nested within each other like this:

This text is bold and italic

In XML all elements must be properly nested within each other like this:

This text is bold and italic

8.3.1.8 All XML Documents must have a Root Tag

All XML documents must contain a single tag pair to define the root element. All other elements must be nested within the root element. All elements can have sub (children) elements. Sub elements must be in pairs and correctly nested within their parent element:

```
<root>
  <child>
    <subchild>
    </subchild>
  </child>
</root>
```

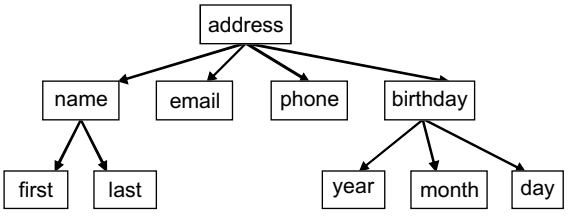
8.3.2 Tree Structure

An XML document exhibits a tree structure. The tree is a general ordered tree. There is a first child, a next sibling, etc. Nodes have parents and children. There are leaf nodes at the bottom of the tree. The declaration at the top is not part of the tree, but the rest of the document is. Consider the following XML document and the related tree.

```
<?xml version = "1.0" ?>
  <address>
        <name>
```

```
                <first>Rama Rao</first>
                <last>Gandhi</last>
    </name>
        <email>ntr@aol.com</email>
        <phone>011-29219292</phone>
        <birthday>
                <year>1903</year>
                <month>07</month>
                <day>15</day>
  </birthday>
    </address>
```

Notice that <name> has two children and <birthday> has three. Most processing on the tree is done with a pre-order traversal.



8.3.2.1 Entities

As in html, certain characters are not allowed in the data. The most obvious ones are the less than signs and quotation marks. Also, ampersands are used to start the escape string, so they too have a substitution. The following are some special characters with their substitutions used in XML data.

```
<       &lt;
>       &gt;
&       &amp;
“       &quot;
‘       &apos;
```

8.3.2.2 Attributes

As in html, tags can have attributes. These are name-value pairs such as width = “300”. We have seen these in applet and image tags. They can be used in XML and are required in some places.

An example from the preceding might be

<name first = “Alice” last = “Lee” />

However this is not very useful for data. It makes it harder to see the structure of the document.

There are places where attributes are necessary. One that we will be using in the future is for giving a reference to the location of a stylesheet.

<link rel=“stylesheet” type=“text/css” href=“address.css” />

8.3.2.3 Attribute Values must always be Quoted

In XML, the attribute value must always be quoted. Study the two XML documents below. The first one is incorrect, the second is correct:

```
<?xml version="1.0"?>
<note date=12/11/99>
<to>Venkat</to>
<from>Sarada</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

```
<?xml version="1.0"?>
<note date="12/11/99">
<to>Venkat</to>
<from>Sarada</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

8.3.2.4 Use of Elements vs. Attributes

Take a look at these examples:

```
Using an Attribute for sex:
<person sex="female">
  <firstname>Garimella</firstname>
  <lastname>Sarada</lastname>
</person>
Using an Element for sex:
<person>
  <sex>female</sex>
  <firstname>Garimella</firstname>
  <lastname>Sarada</lastname>
</person>
```

In the first example sex is an attribute. In the last example sex is an element. Both examples provide the same information to the reader. There are no fixed rules about when to use attributes to describe data, and when to use elements. In XML, it is recommended to avoid attributes as long as the same information can be expressed using elements.

8.3.3 Why one should avoid using Attributes in XML?

These are some of the problems in using attributes.

- Attributes cannot contain multiple values (elements can)
- Attributes are not expandable (for future changes)
- Attributes cannot describe structures (like child elements can)
- Attributes are more difficult to manipulate by program code
- Attribute values are not easy to test against a DTD

If we start using attributes as containers for XML data, we might end up with documents that are both difficult to maintain and to manipulate. We should use **elements** to describe our data. Use attributes only to provide information that is not relevant to the reader.

8.3.3.1 Some Exceptions to the Attribute rule

Rules always have exceptions. There are exceptions to using attributes; for example:
One can assign ID references to elements in XML documents. These ID references can be used to access XML element in much the same way as the NAME or ID attributes in HTML. This example demonstrates this:

```
<?xml version="1.0"?>
<messages>
  <note ID="501">
    <to>Venkat</to>
    <from>Sarada</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
  <note ID="502">
    <to>Sarada</to>
    <from>Venkat</from>
    <heading>Re: Reminder</heading>
    <body>I will not!. Go to Hell.</body>
  </note>
</messages>
```

The ID in these examples is just a counter, or a unique identifier, to identify the different notes in the XML file.

8.3.3.2 CDATA Sections

CDATA stands for character data. XML can have sections that contain characters of any kind that are not *parsed*. This means that they will be ignored by the XML parser that is used to put the document into a tree. These sections are similar to the *pre* sections in HTML. The browser simply displays them unchanged.

CDATA sections begin with <![CDATA[and end with]]>. An example might be an equation like the following:

```
<![CDATA[
        x + 2*y = 3
]]>
```

8.3.3.3 PCDATA

PCDATA means parsed character data. i.e. if we have a character data element declared as PCDATA then all characters or text or data inside the XML tags will be parsed by the XML parser. In this type of data, if we place a character like "<" or "&" inside an XML element, it will generate an error because the parser interprets it as the start of a new element. We cannot write something like this "if salary < 1000 then". It will fire an error. To avoid this, we have to replace the "<" character with an entity reference, like this, "if salary < 1000 then".

8.3.3.4 Styling XML

XML file just describes data in plain text. However, we need to have a way to present the data. Using a style sheet is a method of displaying XML data in a meaningful way. Cascading Style Sheet (CSS) is a rule-based language consisting of two sections:

- A pattern matching section, which expresses the association between an element and some action
- An action section, which specifies the action to be taken upon the specified section.

This means, we have to specify an element and then define how (a style) it has to be displayed. This can be done for each element we define. To add the link to the CSS, we need to add a reference to the CSS file within the XML file.

The following example explains how to create a simple XML file with a style sheet. We have replaced the prologue line with the following line in the XML file A.xml which is used earlier.

```
<?xml-stylesheet type="text/css" href="note1.css"?>
```

The style sheet in the file **note1.css** contains the following lines which indicates the display attributes of each element in the XML file.

```
tip {background-color:green;}

resource {background-color:yellow;}

warning {background-color:red;}
```

When we open the A.xml file along with the above style sheet information in IE browser, the page will look like the following figure (Figure 8.17).

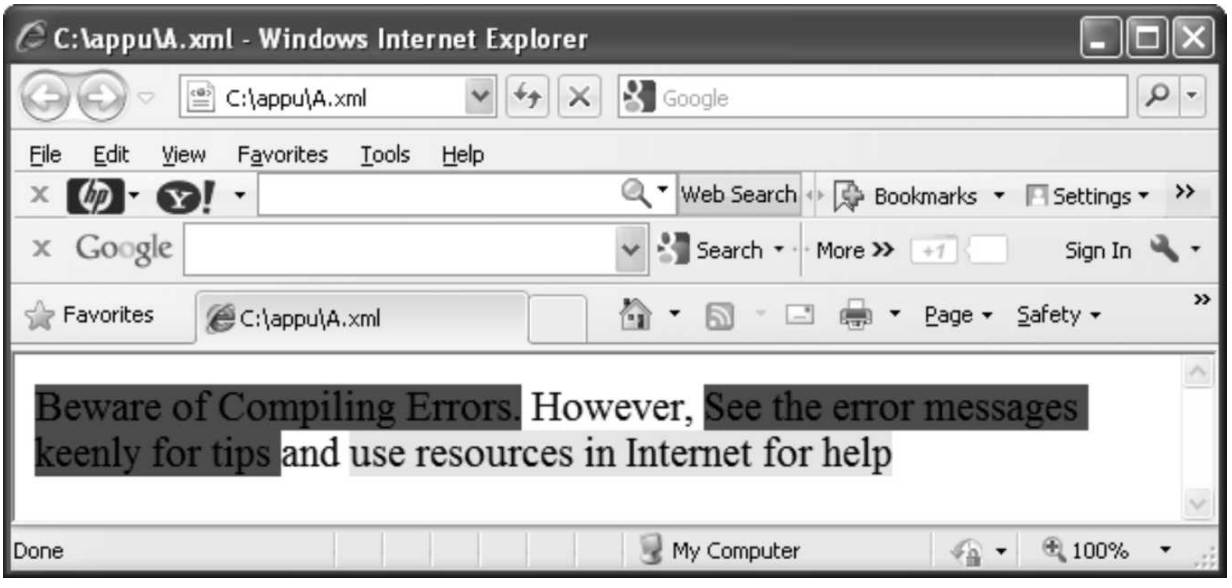


Figure 8.17 A web page illustrating effect of CSS

8.3.3.5 XML, DTD, and XML Schema

Extensible Markup Language (XML) is a markup language generally regarded as the universal format for structured documents and data on the Web. Like HTML, XML contains element tags and attributes that define data. Unlike HTML, XML element tags and attributes are not based on a predefined, static set of elements and attributes. Every XML file can have a different set of tags and attributes. Document Type Definition (DTD) files and XML schema files define the elements and attribute that can be used and the structure within which they fit in an XML file. DTD and XML schema files specify the structure and content of XML files in different ways. A DTD file defines **the names of elements, the number of times they occur, and how they fit together.** The XML schema file provides the same information plus the **data types** of the elements.

8.3.3.6 DTD

The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements. A DTD can be declared inline in your XML document, or as an external reference.

The DTD file contains only metadata. It contains the description of the structure and the definition of the elements and attributes that can be found in the associated XML file. It does not contain any data.

A sample DTD looks like this:

```
<!ELEMENT employees (companyname, employee ) >
<!ELEMENT companyname ( id, name) >
<!ELEMENT employee ( emp+ ) >
<!ELEMENT emp ( id, info ) >
<!ELEMENT info ( name, age, sex, job,  sal ) >
<!ELEMENT created-date ( format, timestamp ) >

<!ELEMENT id ( #PCDATA ) >
<!ELEMENT name ( #PCDATA ) >
<!ELEMENT format ( #PCDATA ) >
<!ELEMENT timestamp ( #PCDATA ) >

eg:
<employees>
      < companyname >
     <id>01</id>
      <name>RITCH CENTER</name>
   </ companyname >
 < employee >
    <emp>
        <id>91000</id>
              <info>
             <name>Venkat</name>
             <age>25</age>
             <sex>Male</sex>
             <job>Tech Lead</job>
             <sal>20000</sal>
                </info>
          </emp>
      </employee>
</employees>
```

8.3.4 XML Schema

The XML schema file, like the DTD file, contains only metadata. In addition to the definition and structure of elements and attributes, an XML schema contains a description of the type of elements and attributes found in the associated XML file.

A sample XML Schema file looks like this:

```
<xs:element name="NBV">
  <xs:complexType>
         <xs:sequence>
           <xs:element ref="NBV_object"/>
```

```
            <xs:element ref=" NBN_object " minOccurs="0" maxOccurs="n"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="NBV_object">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="number" type="xs:string"/>
            <xs:element name="summary" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="NBN_object">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="number" type="xs:string"/>
            <xs:element name="summary" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
E.g.,
<NBV>
<NBV_object>
        <number>00996</number>
        <summary>Testing</summary>
</NBV_object>
<NBN_object>
        <number>00896</number>
        <summary>Test</summary>
</NBN_object>
</NBV>
```

8.3.4.1 Cardinality in XML

Declaring only one occurrence of the same element (only once)

<!ELEMENT companyname (id, name) >(For DTD)

<xs:element name="number" type="xs:string"/>(For Schema file)

Declaring minimum one occurrence of the same element (one or more)

```
<!ELEMENT employee ( emp+ ) >(For DTD)
<xs:element name="number" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>(For Schema file)
        or
<xs:element name="number" type="xs:string" minOccurs="1" maxOccurs="n"/>(For Schema file)
```

Declaring zero or more occurrences of the same element (zero or more)

```
<!ELEMENT employee ( emp* ) >
<xs:element name="number" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>(For Schema file)
        or
```

```
<xs:element name="number" type="xs:string" minOccurs="0" maxOccurs="n"/>(For Schema file)
```

Declaring zero or one occurrences of the same element (zero or one)

```
<!ELEMENT employee ( emp? ) >
<xs:element name="number" type="xs:string" minOccurs="0" maxOccurs="1"/>(For Schema file)
```

8.3.4.2 Displaying XML with XSLT

XSLT is the recommended style sheet language of XML. XSLT (eXtensible Stylesheet Language Transformations) is far more sophisticated than CSS. One way to use XSLT is to transform XML into HTML before it is displayed by the browser.:

8.3.4.3 Loading XML with Microsoft's XML Parser

Microsoft's XML parser is built into Internet Explorer 5 and higher.

The following JavaScript fragment loads an XML document ("note.xml") into the parser:

```
var xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.load("note.xml");
```

- The first line of the script above creates an empty Microsoft XML document object.
- The second line turns off asynchronized loading, to make sure that the parser will not continue execution of the script before the document is fully loaded.
- The third line tells the parser to load an XML document called "note.xml".

The following JavaScript fragment loads a string called txt into the parser:

```
var xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.loadXML(txt);
```

Note: The **loadXML()** method is used for loading strings (text), **load()** is used for loading files.
In the following examples, we will see how an XML file can analyze/capture data at different levels. We will use a simple JavaScript to display the properties and values.

Create a file **F.xml** with the following lines.

```
<?xml version="1.0"?>
<root>
<demoelement demoattribute="things">the PCDATA (parsed character) is here</demoelement>
</root>
```

Also, create another file **F.html** with the following contents.

```
<html>
<head>
<script language="javascript">
var objDOM
objDOM=new ActiveXObject("MSXML.DOMDocument");
objDOM.async=false;
objDOM.load("f.xml");
var objMainNode;
objMainNode=objDOM.documentElement.firstChild;
```

```
alert(objMainNode.nodeName);
</script>
</head>
<body>
  This is a demo - example F
</body>
</html>
```

When we open F.html using the browser (IE) we will get an alert "demoelement" which is the name of the main node. MSIE has a parser called "msxml." The statement creates a new instance of the MS object and assigns it to the variable objDOM.

Now, open F.html in our editor and save it as F3.html. Edit the code as follows

```
objMainNode=objDOM.selectSingleNode ("/root/demoelement");
alert(objMainNode.firstChild.nodeName);
```

When we open F3.html in our browser, the alert will be "#text", which is the type of value.

Now, open F3.html in our editor and save it as F4.html. Edit the code as follows:

```
alert (objMainNode.firstChild.nodeValue);
```

When you open F4.html in your browser, the alert will be "the PCDATA(parsed character) is here".

The whole point of these examples is to show that data can be pulled into HTML files from XML files at various points.

As explained earlier, the MainNode and firstChild refer to the hierarchical, or tree-like structure of XML. It starts with the root and branches to child elements.

Do remember that some of the DOM objects include properties like documentElement, firstChild, lastChild, nextSibling, etc. The methods include removeChild, load, appendChild, etc. The events include onDataAvailable, and onTransformNode.

In the following example, we will continue analyzing XML Nodes with JavaScript. In a new file, type the following XML file, and save it as G2.xml

```
<?xml version="1.0" ?>
<book>
  <title>C and Data Structures: A snap shot oriented treatise with live examples from Science and Engineering</title>
<date>16.06.2001</date>
<author>
          <fname> Venkateswarlu</fname>
  <lname>Nagasuri</lname>
</author>
<abstract> C language is explained with table driven approach</abstract>
<teaser> Data Structures is more complicated, but don't let that put you off.
</teaser>
</book>
 Now type the following lines in a file, and save it as G2.html
<html>
<body>
<script  type= "text/javascript" language= "JavaScript">
var xmlDocument= new ActiveXObject("Microsoft.XMLDOM");
```

```
xmlDocument.load( "G2.xml");
var element = xmlDocument.documentElement;
document.write("<p>The name of the root node in the XML document is: " );
document.write("<strong>" + element.nodeName + "</strong>");
</script>
</body>
</html>
```

Now, add the following code before </script> tag.

```
document.write( "<br>The following are its child elements:");
document.write( "</p><ul>");

for (i = 0; i<element.childNodes.length; i++)
{
var getNode=element.childNodes.item(i);
document.write( "<li>" + getNode.nodeName+ "</li>");
}
document.write("</ul>");
var currentNode= element.firstChild;
document.write( "<p>The first child of the root node is:" + currentNode.nodeName);
document.write ("<br>whose next sibling is:" );
var nextSib = currentNode.nextSibling;
document.write( nextSib.nodeName);
document.write( "<br>Value of " + nextSib.nodeName + "element is:" );
var value= nextSib.firstChild;
document.write(value.nodeValue);
```

The documentElement corresponds to the document's root element. In the *for loop*, length is the number of child nodes. In this manner, we can traverse the elements of a XML tree or document.

8.3.5 Introduction to XPath

Three languages, XQuery, XPath and XSLT are structured XML query languages. XQuery is the most capable and powerful while XPath is simple and efficient; on the other hand XSLT is a full featured language. XSLT is a transformation language of XML, it stands for eXtensible Stylesheet Language Transformation; which means converting XML document into another type of document, including XML, XHTML, HTML and text. It is a functional programming language and uses the same syntax as XML.

XPath is a set of syntax rules for defining parts of an XML document. The idea of XPath is derived from the path expression of an object database, OQL (object query language). The data model in XPath views a document as a tree. In XPath, there are seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document nodes. XML documents are treated as trees of nodes. The topmost element of the tree is called the root element. Absolute and relative operators are used to traverse the tree; and will return a collection of nodes from the tree. Following is an XPath data model tree (Figure 8.18); it is an example of Products.xml from W3Schools.com.

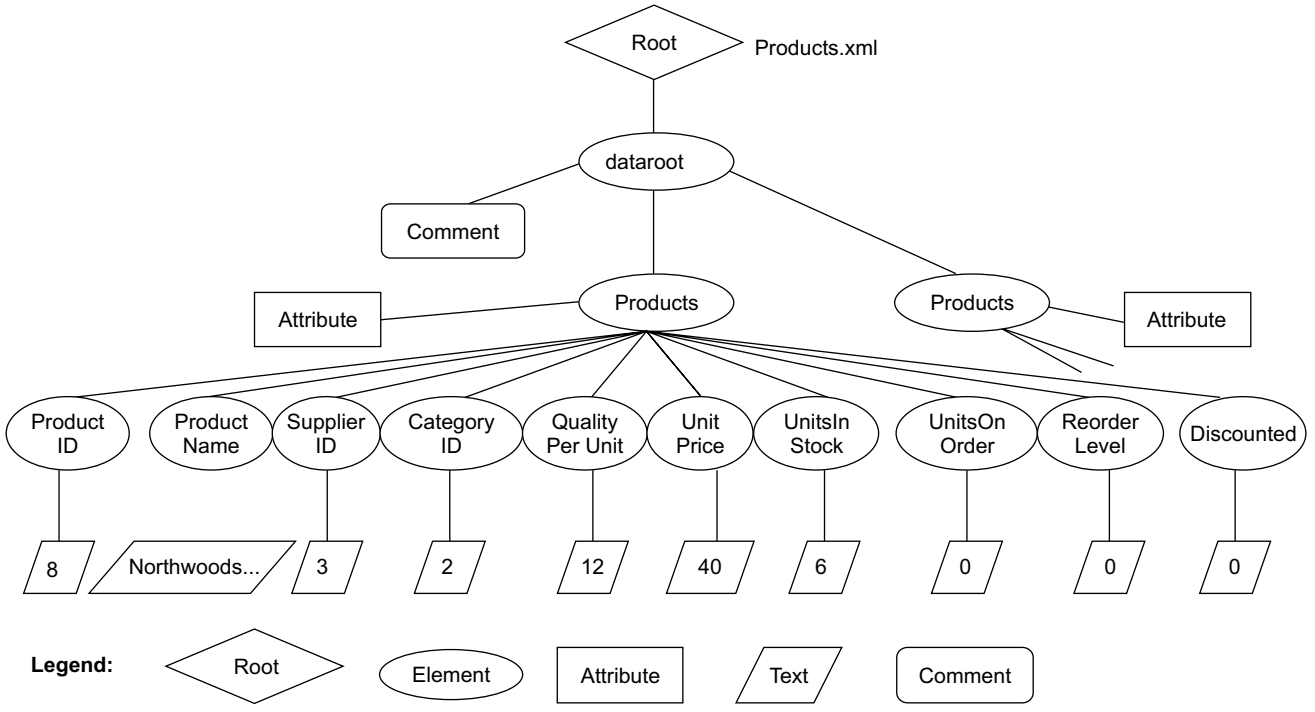


Figure 8.18 XPath Document Tree (*Courtesy* : *W3Schools*)

8.3.5.1 What is XPath?

- XPath is a syntax for defining parts of an XML document
- XPath uses paths to define XML elements
- XPath defines a library of standard functions
- XPath is a major element in XSLT
- XPath is not written in XML
- XPath is a W3C Standard

8.3.5.2 XPath is Used in XSLT

XPath is a major element of the XSLT standard. Without XPath knowledge you will not be able to create XSLT documents.

8.3.5.3 XPath is a W3C Standard

XPath was released as a W3C Recommendation 16. November 1999 as a language for addressing parts of an XML document.
XPath was designed to be used by XSLT, XPointer and other XML parsing software.
XPath uses path expressions to locate nodes within XML documents.

8.3.5.4 XPath is akin to Traditional File Paths

We will be using absolute and relative approaches in referring files in a file system. If we specify the path with respect to the top most directory (which is called as root directory), it is called as absolute path. If we refer with respect to current working directory, it is called as relative path of the file. For example, C:\windows\systems32\notepad.exe is the absolute path. If the current directory is C:\windows then system32\notepad.exe is the relative path. Xpath also uses the same style. XPath uses path expressions to identify nodes in an XML document.

Let us Learn XPath through an example. We assume file name is "cinimacatalog.xml" with the following content.

```
<?xml version="1.0" encoding="utf-8"?>
<cinimacatalog>
 <cd country="India">
  <title>Dukudu</title>
  <artist>Mahesh</artist>
  <price>900</price>
 </cd>
 <cd country="India">
  <title>Pokiri</title>
  <artist>Mahesh</artist>
  <price>800</price>
 </cd>
</cinimacatalog>
```

Here, nodes can be:

```
<cinimacatalog> Root Node
<title>Dukudu</title>
```

As explained earlier, child, grandchild, parent relationships are assumed to be available among the nodes. Essentially, XPath illustrates the selection of nodes based on some criterion. Detailed description is beyond the scope of this book. Thus, I will explain the use of XPath with some examples.

The XPath expression below selects the ROOT element cinimacatalog:

```
/cinimacatalog
```

The XPath expression below selects all the cd elements of the cinimacatalog element:

```
/cinimacatalog/cd
```

The XPath expression below selects all the price elements of all the cd elements of the cinimacatalog element:

```
/cinimacatalog/cd/price
```

If the path starts with a slash (/) it represents an absolute path to an element. Otherwise, it is relative.

The following table illustrates how elements of the catalogue in Figure 1 can be represented in XPath.

XPath Expression Example	Mean
/data/Products/Unit Price	Returns the collection of references to the nodes that correspond to the UnitPrice elements
CategoryID or ./CategoryID	If current position is Products, it will return the node that correspond to the same CategoryID elements
/dataroot/Products/Product Name/text()	The Collection of content of ProductName element
/dataroot/Products/@ProductID	The collection of values of ProductID attribute
/dataroot/Products [1]/CategoryID[1]	Values of first Products node and first Category node will be returned
//CategoryID	All CategoryID elements in the tree (descendant-or-self)
Products/*	All element children of the Products children of the current node
/dataroot/Products [search_expression]	All dataroot nodes which match the expressions

8.3.5.5 Use of XPath Library of Standard Functions

XPath defines a library of standard functions for working with strings, numbers and Boolean expressions. For instance, the XPath expression below selects all the cd elements that have a price element with a value larger than 800:

```
/cinimacatalog/cd[price>800]
```

If the path starts with two slashes (//) then all elements in the document that fulfill the criteria will be selected (even if they are at different levels in the XML tree). The following XPath expression selects all the cd elements in the document:

```
//cd
```

8.3.5.6 Selecting Unknown Elements

Wildcards (*) can be used to select unknown XML elements. The following XPath expression selects all the child elements of all the cd elements of the cinimacatalog element:

```
/cinimacatalog/cd/*
```

The following XPath expression selects all the price elements that are grandchild elements of the cinimacatalog element:

```
/cinimacatalog/*/price
```

The following XPath expression selects all price elements which have 2 ancestors:

```
/*/*/price
```

The following XPath expression selects all elements in the document:

```
//*
```

8.3.5.7 Selecting Branches

By using square brackets in an XPath expression, we can specify an element further.
The following XPath expression selects the first cd child element of the cinimacatalog element:

```
/cinimacatalog/cd[1]
```

The following XPath expression selects the last cd child element of the cinimacatalog element (Note: There is no function named first()):

```
/cinimacatalog/cd[last()]
```

The following XPath expression selects all the cd elements of the cinimacatalog element that have a price element:

```
/cinimacatalog/cd[price]
```

The following XPath expression selects all the cd elements of the cinimacatalog element that have a price element with a value of 900:

```
/cinimacatalog/cd[price=900]
```

The following XPath expression selects all the price elements of all the cd elements of the cinimacatalog element that have a price element with a value of 900:

```
/cinimacatalog/cd[price=900]/price
```

8.3.5.8 Selecting Several Paths

By using the | operator in an XPath expression, we can select several paths. The following XPath expression selects all the title and artist elements of the cd element of the cinimacatalog element:

```
/cinimacatalog/cd/title|/cinimacatalog/cd/artist
```

The following XPath expression selects all the title and artist elements in the document:

```
//title|//artist
```

The following XPath expression selects all the title, artist and price elements in the document:

```
//title|//artist|//price
```

The following XPath expression selects all the title elements of the cd element of the cinimacatalog element, and all the artist elements in the document:

```
/cinimacatalog/cd/title|//artist
```

8.3.5.9 Selecting Attributes

In XPath all attributes are specified by the @ prefix. This XPath expression selects all attributes named country:

```
//@country
```

This XPath expression selects all cd elements which have an attribute named country:

```
//cd[@country]
```

This XPath expression selects all cd elements which have any attribute:

```
//cd[@*]
```

This XPath expression selects all cd elements which have an attribute named country with a value of 'India':

```
//cd[@country='India']
```

8.3.5.10 XPath Location Paths

A location path expression results in a node-set.

8.3.5.11 Location Path Expression

A location path can be absolute or relative. An absolute location path starts with a slash (/) and a relative location path does not. In both cases the location path consists of one or more location steps, each separated by a slash:

An absolute location path:

```
/step/step/...
```

A relative location path:

```
step/step/...
```

The location steps are evaluated in order, one at a time, from left to right. Each step is evaluated against the nodes in the current node-set. If the location path is absolute, the current node-set consists of the root node. If the location path is relative, the current node-set consists of the node where the expression is being used. Location steps consist of :

- an axis (specifies the tree relationship between the nodes selected by the location step and the current node)
- a node test (specifies the node type and expanded-name of the nodes selected by the location step)
- zero or more predicates (uses expressions to further refine the set of nodes selected by the location step)

The syntax for a location step is:

```
axisname::nodetest[predicate]
```

Example:

```
child::price[price=9.90]
```

8.3.5.12 Axes and Node Tests

An axis defines a node-set relative to the current node. A node test is used to identify a node within an axis. We can perform a node test by name or by type.

AxisName	Description
ancestor	Contains all ancestors (parent, grandparent, etc.) of the current node **Note:** This axis will always include the root node, unless the current node is the root node
ancestor-or-self	Contains the current node plus all its ancestors (parent, grandparent, etc.)
attribute	Contains all attributes of the current node
child	Contains all children of the current node

AxisName	Description
descendant	Contains all descendants (children, grandchildren, etc.) of the current node **Note:** This axis never contains attribute or namespace nodes
descendant-or-self	Contains the current node plus all its descendants (children, grandchildren, etc.)
following	Contains everything in the document after the closing tag of the current node
following-sibling	Contains all siblings after the current node **Note:** If the current node is an attribute node or namespace node, this axis will be empty
namespace	Contains all namespace nodes of the current node
parent	Contains the parent of the current node
preceding	Contains everything in the document that is before the starting tag of the current node
preceding-sibling	Contains all siblings before the current node **Note:** If the current node is an attribute node or namespace node, this axis will be empty
self	Contains the current node

8.3.5.13 Examples

Example	Result
child::cd	Selects all cd elements that are children of the current node (if the current node has no cd children, it will select an empty node-set)
attribute::src	Selects the src attribute of the current node (if the current node has no src attribute, it will select an empty node-set)
child::*	Selects all child elements of the current node
attribute::*	Selects all attributes of the current node
child::text()	Selects the text node children of the current node
child::node()	Selects all the children of the current node
descendant::cd	Selects all the cd element descendants of the current node
ancestor::cd	Selects all cd ancestors of the current node
ancestor-or-self::cd	Selects all cd ancestors of the current node and, if the current node is a cd element, the current node as well
child::*/child::price	Selects all price grandchildren of the current node
/	Selects the document root

8.3.5.14 Predicates

A predicate filters a node-set into a new node-set. A predicate is placed inside square brackets ([ ]).

Examples

Example	Result
child::price [price=9.90]	Selects all price elements that are children of the current node with a price element that equals 9.90
child::cd[position()=1]	Selects the first cd child of the current node
child::cd[position()=last()]	Selects the last cd child of the current node
child::cd[position()=last()-1]	Selects the last but one cd child of the current node
child::cd[position()<6]	Selects the first five cd children of the current node
/descendant::cd[position()=7]	Selects the seventh cd element in the document
child::cd[attribute::type="classic"]	Selects all cd children of the current node that have a type attribute with the value "classic".

8.3.5.15 Location Path Abbreviated Syntax

Abbreviations can be used when describing a location path. The most important abbreviation is that child:: can be omitted from a location step.

Abbr	Meaning	Example
none	child::	cd is short for child::cd
@	attribute::	cd[@type="classic"] is short for child::cd[attribute::type="classic"]
.	self::node()	.//cd is short for self::node()/descendant-or-self::node()/child::cd
..	parent::node()	../cd is short for parent::node()/child::cd
//	/descendant-or-self::node()/	//cd is short for /descendant-or-self::node()/child::cd

8.3.5.16 Examples

Example	Result
cd	Selects all the cd elements that are children of the current node
*	Selects all child elements of the current node
text()	Selects all text node children of the current node
@src	Selects the src attribute of the current node
@*	Selects all the attributes of the current node
cd[1]	Selects the first cd child of the current node
cd[last()]	Selects the last cd child of the current node
*/cd	Selects all cd grandchildren of the current node
/book/chapter [3] /para[1]	Selects the first para of the third chapter of the book
//cd	Selects all the cd descendants of the document root and thus selects all cd elements in the same document as the current node
.	Selects the current node
.//cd	Selects the cd element descendants of the current node
..	Selects the parent of the current node
../@src	Selects the src attribute of the parent of the current node
cd[@type ="classic"]	Selects all cd children of the current node that have a type attribute with value "classic"
cd[@type ="classic"][5]	Selects the fifth cd child of the current node that has a type attribute with value "classic"
cd[5][@type ="classic"]	Selects the fifth cd child of the current node if that child has a type attribute with value "classic"
cd[@type and @country]	Selects all the cd children of the current node that have both a type attribute and a country attribute

8.3.5.17 XPath Expressions

XPath supports numerical, equality, relational, and Boolean expressions.

8.3.5.18 Numerical Expressions

Numerical expressions are used to perform arithmetic operations on numbers.

Operator	Description	Example	Result
+	Addition	6 + 4	10
-	Subtraction	6 - 4	2
*	Multiplication	6 * 4	24
div	Division	8 div 4	2
mod	Modulus (division remainder)	5 mod 2	1

XPath always converts each operand to a number before performing an arithmetic expression.

8.3.5.19 Equality Expressions

Equality expressions are used to test the equality between two values.

Operator	Description	Example	Result
=	Like (equal)	price=9.70	true (if price is 9.70)
!=	Not like (not equal)	price!=9.70	false

8.3.5.20 Testing against a Node-Set

If the test value is tested for equality against a node-set, the result is true if the node-set contains any node with a value that matches the test value.

If the test value is tested for not equal against a node-set, the result is true if the node-set contains any node with a value that is different from the test value.

The result is that the node-set can be equal and not equal at the same time !

8.3.5.21 Relational Expressions

Relational expressions are used to compare two values.

Operator	Description	Example	Result
<	Less than	price<9.70	false (if price is 9.70)
<=	Less or equal	price<=9.70	true
>	Greater than	price>9.70	false
>=	Greater or equal	price>=9.70	true

XPath always converts each operand to a number before performing the evaluation.

8.3.5.22 Boolean Expressions

Boolean expressions are used to compare two values.

Operator	Description	Example	Result
or	or	price=9.70 or price=9.60	true (if price is 9.70)
and	and	price<=9.70 and price=9.60	false (if price is 9.70)

XPath Functions

XPath contains a function library for converting data.

XPath Examples
We will use the CD cinimacatalog to demonstrate some more XPath examples.

Selecting Nodes
We can select nodes from the XML document by using the selectNodes function in Internet Explorer. This function takes a location path expression as an argument:

```
xmlobject.selectNodes(XPath expression)
```

The following example selects all the cd nodes from the CD cinimacatalog:

```
xmlDoc.selectNodes("/cinimacatalog/cd")
```

Selecting the First cd Node
The following example selects only the first cd node from the CD cinimacatalog:

```
xmlDoc.selectNodes("/cinimacatalog/cd[0]")
```

Selecting price Nodes
The following example selects all the price nodes from the CD cinimacatalog:

```
xmlDoc.selectNodes("/cinimacatalog/cd/price")
```

Selecting price Text Nodes
The following example selects only the text from the price nodes:

```
xmlDoc.selectNodes("/cinimacatalog/cd/price/text()")
```

Selecting cd Nodes with Price>10.80
The following example selects all the cd nodes with a price>10.80:

```
xmlDoc.selectNodes("/cinimacatalog/cd[price>10.80]")
```

Selecting price Nodes with Price>10.80
The following example selects all the price nodes with a price>10.80:

```
xmlDoc.selectNodes("/cinimacatalog/cd[price>10.80]/price")
```

8.3.6 Introduction to XQuery

XQuery is designed to query XML data in XML files (also anything that can appear as XML, including XML databases). We can say that XQuery is for XML which is akin to SQL for databases.

8.3.6.1 What is XQuery?

- XQuery is ***the*** language for querying XML data
- XQuery for XML is like SQL for databases
- XQuery is built on XPath expressions
- XQuery is supported by all the major database engines (IBM, Oracle, Microsoft, etc.)

XQuery 1.0 and XPath 2.0 share the same data model and support the same functions and operators. If we have already studied XPath we will have no problems in understanding XQuery.

8.3.6.2 How to select Nodes from "cinimacatalog.xml" ?

Functions
XQuery uses an extensive set of functions to extract data from XML documents. For instance, the doc() function is used to open the "cinimacatalog.xml" file:

```
doc("cinimacatalog.xml")
```

8.3.6.3 Path Expressions

XQuery uses an path expressions to navigate through elements in an XML document. The following path expression is used to select all the title elements in the "cinimacatalog.xml" file:

```
doc("cinimacatalog.xml")/cinimacatalog/cd/title
```

The XQuery above will extract the following:

```
<title>Dukudu</title>
<title>Pokiri</title>
```

8.3.6.4 Predicates

XQuery uses predicates to limit the extracted data from XML documents. The following predicate is used to select all the book elements under the cd element that have a price element with a value that is more than 800:

```
doc("cinimacatalog.xml")/cinimacatalog/cd[price>800]
```

8.3.6.5 XQuery FLWOR Expressions

FLWOR is an acronym for "For, Let, Where, Order by, Return". The expression below will select all the title elements under the cd elements that are under the cinimacatalog element that have a price element with a value that is higher than 300.

```
doc("cinimacatalog.xml")/cinimacatalog/cd[price>300]/title
```

The following FLWOR expression will select exactly the same as the path expression above:

```
for $Y in doc("cinimacatalog.xml")/cinimacatalog/cd
where $Y/price>300
return $Y/title
```

With the following FLWOR expression, we can sort the result:

```
for $Y in doc("cinimacatalog.xml")/cinimacatalog/cd
where $Y/price>30
order by $Y/title
return $Y/title
```

Here, the **for** clause selects all cd elements under the cinimacatlog element into a variable called $Y. The **where** clause selects only the cd elements with a price element with a value greater than 300. The **order by** clause defines the sort-order (Here it is sorted by the title element). The **return** clause specifies what should be returned. Here it returns the title elements.

8.3.6.6 XQuery FLWOR + HTML

Now, if we want to list all the titles as an HTML list. We can add <ul> and <li> tags to the FLWOR expression:

```
<ul>
{
for $x in doc("cinimacatalog.xml")/cinimacatalog/cd/title
order by $x
return <li>{$x}</li>
}
</ul>
```

Now if we want to eliminate the title element, and show only the data (titles names) inside each of the the title elements, then the following can be used.

```
<ul>
{
for $x in doc("cinimacatalog.xml")/cinimacatalog/cd/title
order by $x
return <li>{data($x)}</li>
}
</ul>
```

8.3.6.7 XQuery Conditional Expressions

"If-Then-Else" expressions are allowed in XQuery. Assume that we want to display those cd's in which "Mahesh" is the artist.

```
for $x in doc("cinimacatalog.xml")/cinimacatalog/cd
return    if ($x/@actor="Mahesh")
  then <li>{data($x/title)}</li>
              else().
```

8.3.6.8 XQuery Comparisons

In XQuery there are two ways of comparing values.

1. General comparisons: =, !=, <, <=, >, >=
2. Value comparisons: eq, ne, lt, le, gt, ge

The difference between the two comparison methods are shown below.
Look at the following XQuery expressions:

```
$cinimacatalog//cd/@q > 10
```

The expression above is true if any q attributes have values greater than 10.

```
$cinimacatalog//cd/@q gt 10
```

The expression above is true if there is only one q attribute returned by the expression, and its value is greater than 10. If more than one q is returned, an error occurs.

8.3.6.9 The For Clause

The FOR clause binds a variable to each item returned by the (in expression). The FOR clause results in iteration. There can be multiple For clauses in the same FLWOR expression.

To loop a specific number of times in a FOR clause, you may use the **to** keyword:

```
for $x in (1 to 5)
return <test>{$x}</test>
```

Result :

```
<test>1</test>
<test>2</test>
<test>3</test>
<test>4</test>
<test>5</test>
```

The **at** keyword can be used to count the iteration:

for $x at $i in doc("cinimacatalog.xml")/cinimacatalog/cd/title

return <cd>{$i}. {data($x)}</cd>

Result:

<cd>1. Dukudu</cd>

<cd>2. Pokiri</cd>

It is also allowed with more than one **in expression** in the for clause. Use comma to separate each **in expression**:

```
for $x in (10,20), $y in (100,200)
return <test>x={$x} and y={$y}</test>
```

Result:

```
<test>x=10 and y=100</test>
<test>x=10 and y=200</test>
```

```
<test>x=20 and y=100</test>
<test>x=20 and y=200</test>
```

8.3.6.10 The let Clause

The let clause allows variable assignments and it avoids repeating the same expression many times. The let clause does not result in iteration.

let $x := (1 to 9)

return <test>{$x}</test>

Result:

<test>1 2 3 4 5 6 7 8 9</test>

QUESTIONS

1. **What is complement to CDATA in XML?**

   Answer: PCDATA, which is required to be parsed by the XML parser.

2. **What is the output of the following XML document?**

```
<?xml version="1.0"?>
<staff>
      <employee id="45">
                <name>PN Rao</name>
                <salary>Rs.100,000</salary>
      </employee>
</staff>
<xsl:stylesheet version="1.0" xmlns:xsl="http://
www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
       <xsl:if test="employee/name = 'PN Rao'">
                true
        </xsl:if>
    </xsl:template>
</xsl:stylesheet>
```

   Answer: Nothing

3. **What is the output of the following XSL statement?**

```
<xsl:text disable-output-escaping="yes">
<!-- hello -->
</xsl:text>
```

   Answer: Nothing

4. **Which XPath query is needed to select bids whose item number is 1? Consider the following XML document.**

```
<auctions>
<items>
<item itemno="1" description="Sitara" reserve-
price="1000"></item>
<item itemno="2" description="Guitar" reserve-
price="20000"></item>
<item itemno="3" description="Drums" reserve-
price="8000"></item>
</items>
<bids>
<bid username="PN Rao" bidprice="200" item-
no="1"></bid>
<bid username="S.N. Rahu" bidprice="800" item-
no="1"></bid>
<bid username="PN Rao" bidprice="6500" item-
no="3"></bid>
</bids>
</auctions>
```

   Answer: /auction/bids/bid[@itemno = "A"]

5. **What is the XPath query needed to display all claims in the following XML document?**

```
<policies>
<policy type="electronics goods">
<policy-holder>PN Rao </policy-holder>
<claims>
<claim>
<year>2002</year>
<description>TV</description>
</claim>
</claims>
</policy>
<policy type="fire">
<policy-holder> PN Rao </policy-holder>
<claims>
<claim>
```

```
<year>2007</year>
<description>Front Room</description>
</claim>
</claims>
</policy>
</policies>
```

Answer:

```
//claim
/policies/policy/claims/*
/policies/policy/claims/claim
```

6. Consider the following XML file. Answer the questions following the XML file description.

```
<addressBook>
<address>
<firstName>Reddy</firstName>
<surname>PP</surname>
<email>ppr@world.org</email>
<tel type="work">9822332334</tel>
</address>
<address>
<firstName>Rao</firstName>
<surname>PN </surname>
<email>PNrao@yahoo.com</email>
<tel type="home">9823334444</tel>
</address>
<address>
<firstName>Smurthi</firstName>
<surname>Kakarla</surname>
<email>sk@rock.com</email>
</address>
</addressBook>
```

What is the expression to display the first names of all the people?

/addressBook/address/firstName/text()

What is the expression to display email IDs of all the people?

/addressBook/address/email/text()

What is the expression to display work telephones of all the people?

/addressBook/address/tel[@type="work"]

What is the expression to display work telephones of all the people?

/addressBook/address/tel[@type="work"]/text()

The above displays only work telephone numbers. Not all the elements.

OBJECTIVE TYPE QUESTIONS

1. How is XML data described?
- A. XML uses a description node to describe data.
- B. XML uses a DTD to describe data.
- C. XML uses XSL to describe data.

2. Correct syntax of the declaration that defines the XML version is
- A. <?xml version="1.0" />
- B. <xml version="1.0" />
- C. <?xml version="1.0"?>

3. The valid statement about XML
- A. All XML elements must be properly closed
- B. All XML elements must be in lower case
- C. All XML documents must have a DTD
- D. All the statements are true

4. Correct name for an XML element
- A. <h1>
- B. <Note>
- C. <1dollar>
- D. All the above names are incorrect

5. Correct way of referring to a stylesheet called "mystyle.xsl"
- A. <stylesheet type="text/xsl" href="mystyle.xsl" />
- B. <?xml-stylesheet type="text/xsl" href="mystyle.xsl" ?>
- C. <link type="text/xsl" href="mystyle.xsl" />

6. An XML document exhibits _____ structure
- A. Style
- B. Linear
- C. Tree
- D. Linked

7. ___ character is used to, start an escape string.
- A. \
- B. .
- C. /
- D. &

8. XML supported character data does not use ___
- A. 12 bytes
- B. 8 bits
- C. 16 bits
- D. 4 bytes

9. In the following XML document, the root element is _____

```
<?xml version = "1.0" ?>
<address>
        <name>NB Venkat</name>
        <email>nbv@yahoo.com</email>
        <phone>08913098705</phone>
        <birthday>15-08-1963</birthday>
  </address>
```

A. name B. email
C. address D. None

10. Processing XML tree is normally carried out in __ fashion.
A. Level by level B. Level order
C. In order D. Pre-order

11. Find the correct XML statement.
A. <![CDATA[x + 2*y = 3]]>
B. <[CDATA[x + 2*y = 3]]>
C. <[[CDATA[x + 2*y = 3]]>
D. <![CDATA[x + 2*y = 3]!>

12. The tag element equivalent to <pre> of HTML
A. <pre> B. <No Format>
C. CDATA D. <No Parsing>

13. Structure and content of an XML file is specified through
A. an XML file B. a DTD file
C. an XML schema file D. an HTML file

14. Not a valid XPath
A. ancestor B. attribute
C. following D. followed

15. To find a specific node ___ are used.
A. ancestor B. predicate
C. attribute D. //$

16. In valid character with respect to XPath
A. / B. @
C. // D. $

17. ____ node is also called as "root".
A. comment B. element
C. document D. namespace

18. XPath is a
A. Compiler B. Parser
C. Debugger D. Assembler

19. Which is valid?
A. <xsl:variable name="$hisName" select="'Syd'"/>
B. <xsl:variable name="hisName" select="Syd"/>
C. <xsl:variable name="hisName" select="'Syd'"/>
D. <xsl:variable name="$hisName" select="Syd"/>

20. ___ XPath function gives you back the largest number of a sequence that we pass to it.
A. max() B. largest()
C. highest() D. top()

21. Imagine you have a list of names, and many of them occur several times in the list. What function would you use to get a reduced list, with no duplicates in it?
A. distinct-values() B. starts-with()
C. string-join() D. remove-duplicates()

22. Invalid Xpath function
A. string() B. string-length()
C. string-concat() D. None

23. In XPath, the following axis contains:
A. all descendants of the node
B. all the children of the node
C. all the nodes that appear after the current node is opened
D. all the nodes that appear after the current node is closed

24. Consider the following XML document, what will be the output of count(//*)?

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
      <employee id="45">
                <name>Rao PN</name>
                <salary>Rs.100,000</salary>
      </employee>
</root>
```

A. 2 B. 3
C. 4 D. 6

25. In XPath, the descendant-or-self axis contains the context node and the descendants of the context node. (y/n)

26. _____ XPath query returns all items with a quantity of 10.
A. //item
B. //item/quantity=5
C. //item[quantity="5"]
D. //item[@quantitiy="5"]

27. xsl:for-each is used
A. To select the elements whole document of the
B. To select a single node
C. To iterate over the nodes of a XML document
D. None

28. Xquery uses ___ syntax to address parts of the XML document.
A. XSL B. XSLT
C. XPath D. XML

29. XQuery language extracts elements and attributes from ____
A. Text files B. XML documents
C. Databases D. Database tables

30. Find the wrong statement.
A. XQuery can be used for generating summary resports.
B. XQuery can be used for transforming XML data to XHTML.

C. XQuery can be used to search web documents for required information.

D. None

31. __ is used in XQuery language to open a XML document.

A. open() B. fopen()

C. fileopen() D. doc()

32. Invalid XQuery node

A. root B. attribute

C. number D. namespace

33. Nodes that have the same parent are

A. Children B. Siblings

C. Successors D. None

ANSWER KEY

1. B	**2.** C	**3.** C	**4.** D
5. B	**6.** C	**7.** A	**8.** A
9. C	**10.** D	**11.** A	**12.** C
13. B,C	**14.** D	**15.** B	**16.** D
17. C	**18.** B	**19.** C	**20.** A
21. A	**22.** C	**23.** D	**24.** C
25. N	**26.** D	**27.** C	**28.** C
29. B	**30.** D	**31.** D	**32.** C
33. B			

8.4 Client/Server Computing

Client/server computing is the result of years of computing research which is one of the central ideas of network computing. Undoubtedly, all of today's business applications use the client/server model. For example, to check our bank account, a client program in our computer forwards our request to a server program at the bank. That program may in turn send results to the client software in our personal computer, which displays the information for us. The following is a brief recapitulation of the chronological developments in computing that lead to the development of client/server computing.

1970s Mainframes with users accessing mainframe via dumb terminals

1980s PC with data residing in mainframes (manual extraction)

1980s PC with data residence on mainframes (intelligent terminal extraction)

(Proliferation of Snapshots of mainframe database)

1990s Networks of heterogeneous computers. Data access without regard to the data location, data model, or communication characteristics of the other computers in the network (software and hardware independence).

Today Modern end users use intelligent computers, GUIs, user friendly systems, and data analysis tools to effectively increase their productivity. In addition, cloud computing is expected to inundate the IT services.

8.4.1 By the way what is Client/Server?

Client/server is a computational architecture that involves client processes requesting service from server processes. In the client/server model, these processes are generally maintained on different pieces of hardware. **But it is important to remember that the client and server are both SOFTWARE processes!** Although many generally call hardware boxes 'servers' or 'clients', the clients or servers are actually the software processes run by these computers. Evidently, even though the client and server processes are separated, a Client/server application gives the impression of a single application to the user. Client-Server computing assumes that separation of a huge program into its constituent parts ("modules") can create the possibility for further modification, easier development and better maintainability. In Client-Server Computing, all large modules need not be executed within the same **memory space**. With this architecture, the calling module becomes the client (requesting service) and the called module becomes the server (providing service). Clients and Servers are run separately on appropriate hardware and software platforms for their functions. For example, database management system servers run on platforms specially designed and configured to perform queries, or file servers run on platforms with special elements for managing files.

8.4.1.1 The CLIENT

The client is a process (program) that sends a message to a server process (program), requesting the server to perform a task (service). Client programs usually manage the user-interface portion of the application, validate data entered by the user, dispatch requests to server programs, and sometimes execute business logic. The client process contains solution-

specific logic and provides the interface between the user and the rest of the application system. The client process also manages the local resources that the user interacts with such as the monitor, keyboard, workstation CPU and peripherals. Evidently, graphical user interface (GUI) is one of the prominent features of modern client processes.

8.4.1.2 The SERVER

The server is also a process (program) that is developed to fulfill the client requests by performing the tasks requested. For instance, a Servlet based Server program may receive requests from a client program (may be java applet), executes a SQL query to retrieve a record from the database and dispatches the same as a response to the client requests. Very often, a server process acts as a software engine that manages shared resources such as databases, printers, communication links, or high powered-processors. Rather, we can say that the server process performs the back-end tasks. Thus, server related software development is also called as **back-end** development while client related software development as **front-end** development.

8.4.1.3 Internet: A good example of Client/Server Computing Model

The Internet services including e-mail, newsgroups, file transfer, remote login, and the Web are organised according to a client/server architecture. Client programs, such as Web browsers and file transfer programs, create connections to servers, such as Web and FTP servers. The clients make requests of the server, and the server responds to the requests by providing the service requested by the client. Here, Web browsers are the clients and Web servers are the servers. Browsers request HTML files from Web servers on our behalf by establishing a connection with a Web server and submitting file requests to the server. The server receives the file requests, retrieves the files, and sends them to the browser over the established connection. The browser receives the files and displays them on our browser window (see Figure 8.19).

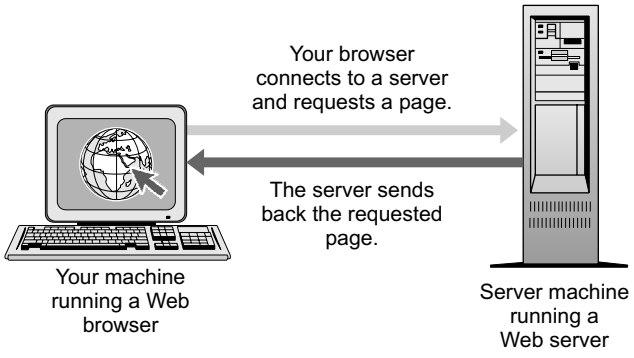


Figure 8.19 A simple web server

8.4.2 Sockets and Client/Server Communication

In practice, clients and servers establish connections and communicate via **sockets**, which are communication links that are created over the Internet using TCP (or UDP). Sockets are the endpoints (Figure 8.20) of Internet communication stream or channel (logical). Clients create client sockets and connect them to server sockets. Sockets are associated with a host address and a port address. The host address is the IP address of the host where the client or server program is located. The port address is the communication port used by the client or server program. Server programs use the well-known port number associated with their application protocol. A client communicates with a server by establishing a connection to the socket of the server. The client and server then exchange data over the connection.

8.4.2.1 Sockets

The socket is the software abstraction used to represent the "terminals" of a connection between two machines. For a given connection, there's a socket on each machine, and we can imagine a hypothetical "cable" running between the two machines with each end of the "cable" plugged into a socket. Of course, the physical hardware and cabling between machines is completely unknown. The whole point of the abstraction is that we don't have to know more than is necessary.

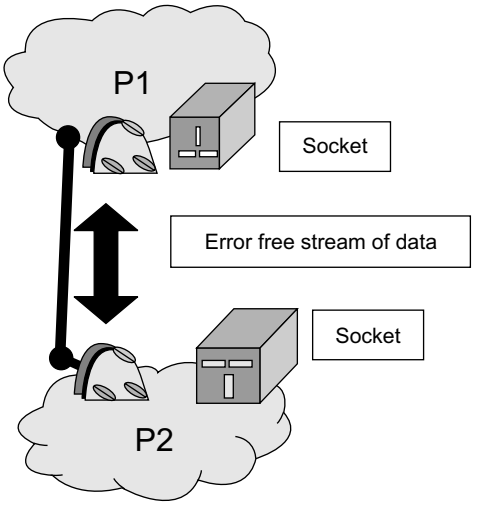


Figure 8.20 Sockets are means of communication

If two processes wish to communicate they must each create a socket. Once created, the socket must then be bound to a specific network address. One process must initiate the connection (This process is called the client). The other process must wait for a connection request (This process is the server). This delegation of responsibilities is highlighted in Figure 8.20 and 8.21. The client starts by trying to establish a connection with the listening socket at a server. If the server accepts the communication request then the connection is established and communication can start. If a connection cannot be established then the client process must recover from this situation. It can do this either by trying to re-connect to the same server or by choosing to connect to a different server that offers the same services. Once the communication has ended the sockets must be closed.

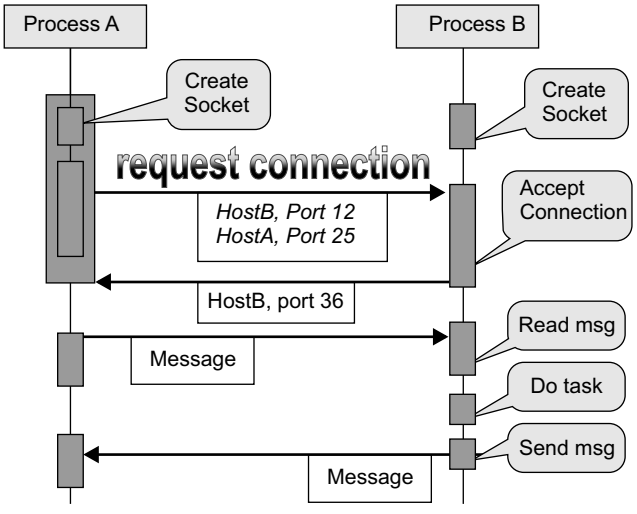


Figure 8.21 Communication oriented client-server communication

The client process needs to know the Internet address and the port number of the server process in order to initiate the connection. If the client simply knows the servers Internet address as a string such as "www.ritchcenter.com" then it must first use a lookup service to obtain the true 32 bit internet address of the server. When the client knows the servers real address, it sends a connection request packet containing its own Internet address and port number to the server process where required service will be available. If the server process accepts the connection request from the client, it creates a new socket bound to a new port number for use by the client and sends this information back to the client as an acknowledgement. Once the client receives a positive acknowledgment from the server it re-binds its socket to the new port number supplied by the server and at this point a connection has been established. Both processes now know the Internet address and port number of the process at the other end of the connection and therefore bi-directional communication can begin (Figure 8.21).

8.4.3 Characteristics of Client-Server Architecture

Basic characteristics of Client-Server Architecture can be summarised as:

- Client portion interacts with the user while a server or back-end portion interacts with the shared resource.
- The front-end task and back-end task have fundamentally different requirements for computing resources such as processor speeds, memory, disk speeds and capacities, and input/output devices.
- The environment can be heterogeneous and multivendor. That is, hardware platform and operating systems of client and server systems are not necessarily same. Client and server processes communicate through a well-defined set of standard application program interfaces (API's) and RPC's (remote procedure calls).
- Scalability, both horizontally and vertically, is an important characteristic of client-server systems. Horizontal scaling means adding or removing client workstations will have only a slight performance impact on the total system. Vertical scaling means migrating to a larger and faster server machine or multiple servers.

8.4.4 Some Goals of Client/Server Systems

Main goals include development of systems that are independent of hardware or software (cross boundaries) to optimise processing resources. Evidently, Client/server development is based on a very complex technology that generates its own set of management problems but goals focus on

- applications development and implementation costs
- advantages of scalability and portability (modular and flexible)
- system operations costs
- change of function from development to end user support (user productivity)

8.4.5 Types of Servers

Disk and File Servers

The simplest forms of servers are disk servers and file servers. In a file server, the client passes requests for files or file records over a network to the file server.

Database, Transaction and Application Servers

The more advanced forms of servers are Database servers, Transaction servers and Application servers.

In **database servers**, client process pass SQL (Structured Query Language) requests as messages to the server while the results of the query are returned over the network by the server.

In **transaction servers**, clients invoke remote procedures that reside in the servers with a database engine. There are procedural statements on the server to execute a group of SQL statements (transactions) which either all succeed or fail as a unit.

Application servers are not necessarily database centered but are used to serve user needs, such as regulating an electronic mail process, etc. Basing resources on a server, users are allowed to share data.

8.4.6 Reasons for employing Client-Server Technology in Business

Client/server computing has arisen because of a change in business needs. Businesses today need integrated, flexible, responsive and comprehensive applications to support the complete range of business processes. Problems with yesterday's systems include:

- Applications were developed to model vertical applications
- Applications were built in isolation
- Applications were implemented as monolithic systems
- Applications were complex
- The supporting technology was based on a centralised control model

The development and implementation of client/server computing is more complex, more difficult and more expensive

than traditional, single process applications. The only answer to the question "why build client/server applications?" is because the business demands the increased benefits.

8.4.7 Business Benefits From Client-Server Computing

Tremendous benefits for implementing Client/Server Technology in Business. Below are just a few of it.

- Vendor independence as compared to the traditional mainframe computing. This includes application development methodologies, programming paradigms, products and architectures.
- Organisation might have changed from steep hierarchies to flattened hierarchies. Decision making is carried out by many lower ranked managers across the organisation rather than performed only by CEOs in the past.
- Network management is replacing vertical management.
- Faster response and flexibility to changing environment of business world outside.
- The customer has a single point of contact for all business with the organisation.
- The time required to complete the work will be minimised.
- Better sharing of critical database resources and other application software's among clients through the network.
- Companies can cut costs on maintenance in employing Client/Server Computing since servers are usually cheaper than mainframe (1/10 of mainframe) and the performance of servers and mainframe are nearly identical.
- Networked webs of small, powerful machines. If one machine goes down, the organisation can still function properly.
- Systems grow easily. It is easy to update and modernise systems both hardware and software as the companies evolve and have new requirements.
- We can mix and match computer platforms to suit the needs of individual departments and users (Individual client).

8.4.8 Business Drivers for Client / Server Computing

- Downsising from 'big iron' (mainframe) environments
- Movement from vertical 'stovepipe' applications to enterprise-wide systems
- Organisations have changed from steep hierarchies to flattened hierarchies
- Network management is replacing vertical management
- There is a change to team based management
- The user will perform as much processing as possible during customer contact time
- Multi-skilled and multi-function teams need access to multiple applications

8.4.9 Client/Server Architecture

Client/Server architecture is based on a set of principles. These are:

- Hardware and Software Independence
- Open Access to Services
- Process Distribution
- Standards

Components of a client/server system

- Client
- Server
- Communications channel

Some systems include middleware as a separate component since middleware (**software that is used to manage client/server interactions**) is critical to management of the communications channel. Middleware provides services to insulate the client from the details of network protocols and server processes. The client/server network infrastructure includes network cabling, network topology, network type, communication devices, and network protocols. But, network proto-

cols constitute the core of the network infrastructure. Network protocols determine how messages between computers are sent, interpreted and processed.

8.4.10 Categories of Component Processes in the Client/Server Model

Three categories of processes are:

- The *User Interface* (**UI**): How the application is presented to the user
- The *Business Logic* (**BL**): The processing rules that are used in the C/S application; for example, a payroll application will have business rules different from that of a trading floor application.
- *Data Management* (**DM**): The storage of the data used by the application which implements the information model

These three components can be divided among the client and server processes in several ways as shown in the table below.

8.4.10.1 Types of Client/Server Computing

The Gartner Group came out with five ways of describing the different client/server styles based on how they split the three components of any application: user interface, business or application logic, data management. The five styles are distributed presentation, remote presentation, distributed function, remote data management, and distributed data management. (Note: This is an arbitrary classification and others may do it differently).

Types of Client/Server Computing					
C/S Type	**Distributed Presentation**	**Remote Presentation**	**Distributed Logic**	**Remote Data Management**	**Distributed Database**
Functions on Server Side	Data Mgt. Business Logic User Interface	Data Mgt. Business Logic	Data Mgt. Business Logic	Data Mgt.	Data Mgt.
Network over which the processes communicate					
Functions on Client Side	User Interface	User Interface	Business Logic User Interface	Business Logic User Interface	Data Mgt. Business Logic User Interface
	'Thinnest' Client ---------- 'Fattest' Client				
Courtesy of Datamation, 4/1/95					

Remote Data Management: In remote data management, the entire application resides on the client and the data management is located on a remote server/host. Remote Data Management is relatively easy to program as there is just one application program.

Distributed Logic (Distributed Application Logic): Here, the split occurs in the application functionality, one part goes to the client and the other to the server. Here, two separately compiled application programs must be developed. Developers first analyse where each function should reside and what type of dialog must occur between the two programs. The underlying communications facilities may implement either a message-based or remote procedure call (RPC) mechanism for transfer of dialog and data.

Distributed presentation: Evidently, for people whose roots are in the IBM mainframe world, client-server is essentially distributed or remote presentation. This style maps a workstation Graphical User Interface (GUI) front end onto an existing application's text-based screen. In Distributed Presentation, we are generally looking at a system which uses a dumb terminal at the client side. This is the extreme of the 'thin' client model (discussed below). Distributed presentation is the first step in migration of legacy applications to a GUI. In a nutshell, a distributed presentation model characterises:
Presentation management function shared between client and server, everything else remains on the server.

Remote Presentation: Here, presentation manager entirely lies in client while presentation logic, data logic and data manager lies in the server. Usually, X window system (X client or X terminals) is used as windows manager which takes care of GUI aspects.

Distributed Database: Here, data management and application functions occur at both the client and server. In this instance, data management at the client would include referential, read-only data. Data frequently updated or accessed by many users would reside on the server. This is the extreme of the 'fat' client model. The following figure 8.22 illustrates where the network activity takes place in various models described above.

Presentation layer is the one through which the user can submit operations and obtain a result.
Application logic: This establishes what operations can be performed and how they take place. It enforces business rules and establishes business processes.
Resource manager: This deals with storage, indexing, and retrieval of data necessary to support the application logic.

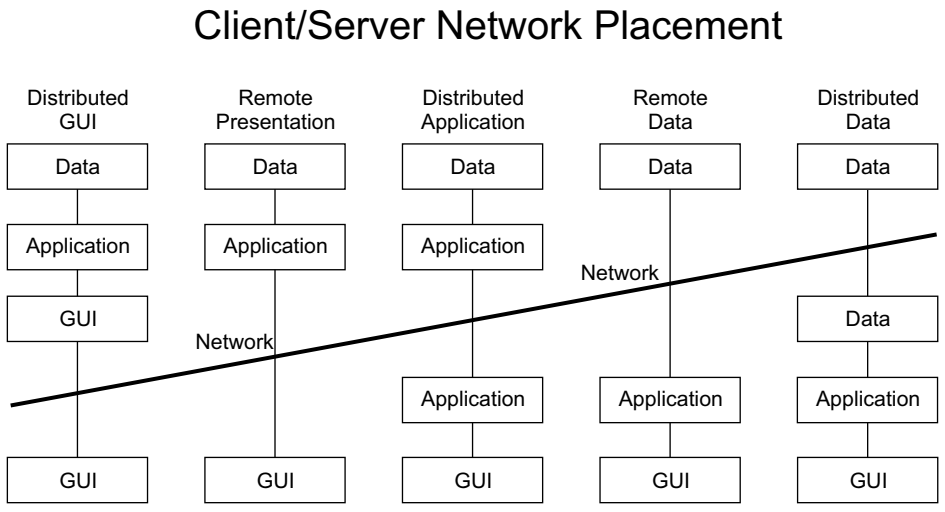


Figure 8.22 Client server Models and network activity

8.4.10.2 Fat Clients vs. Thin Clients

Fat Clients: These are fully loaded PC's or laptops equipped with a full suite of PC operating system, Windows, PC applications and network connectivity software. These systems are ready to run all types of processes such as user interface, business logic, and data management. Thus, they are costly and are complex to manage. These are not very scalable and hence larger systems cannot be developed easily. If application becomes complex, the client becomes fatter. Often, the client hardware thus must become increasingly powerful in order to be able to support it. As a result, the cost of adequate client technology can become rather prohibitive. It may in fact defeat the affordability of the application. What is more, the footprint of the network utilising the fat clients is incredibly large, (think Bigfoot here), so there is an inevitable reduction of the network's bandwidth as well as the number of users who can use the network in an effective manner.

Another approach often invoked in Two-tier architecture is the thin client and fat server configuration. In this configuration, the user will invoke procedures that are stored at the database server. The fat server model gains performance in a more effective fashion, as the network footprint, while heavy, is still a lot lighter than the fat client method.
Thin Clients: These are machines that will download what they need to run from a network server. They attach to a server and provide a graphical interface on a terminal that is optimised for network-centric computing. Thin clients cost a fraction of the price of a fat client. Thin clients can readily access the internet, network based applications and other host based systems. These devices are designed to take the complexity out of managing them. They are managed like dumb terminals; there are no local data storage and no disk drives to worry about, yet users are able to select applications from various operating systems connected to the network simultaneously.

Fat Client:
Advantages:

- More flexibility for the user.
- As the client does most of the processing, the server need not to be very powerful.

Disadvantages:

- Because the server has to deliver more data for the client to process, there are large data sets going through the network, leading to greater network congestion;
- Machines capable of running fat client processes are more expensive, because they require more processing power, more RAM, and more secondary storage;

- Fat clients increase managerial costs; changes to business logic have to be distributed to all of the individual clients; there is much more user training involved; fat clients have a variety of software that must be supported;
- The annual cost of ownership of each fat client is relatively more which includes all the real costs of hardware and software, plus all the hidden costs of software upgrades, managing the machines, and training the users for all the different client processes they are using.

Thin Client:

Advantages:

- The server only delivers the **results** of client queries, not entire data sets. Because of this the server sends much less data through the network. Thus there is generally less congestion on the network;
- Machines that run only thin client processes do not need as much processing power, RAM, or secondary storage. Because of this, these thin client machines are less expensive;
- Thin clients minimise some managerial costs; most changes can be made at the server level, eliminating the need to distribute changes to each client machine.

Disadvantages:

- Less flexibility for the user;
- As the server handles most of the processing, the server machine has to be very powerful;

8.4.11 Two-tier and Three-tier Architectures

8.4.11.1 Two-tier Architecture

The three components of an application (user interface, business logic, and data management) are split into only two tiers, that is as the client and the server. In a two-tier architecture, a client talks directly to a server, with no intervening server. It is typically used in small environments (less than 50 users). A common error in client/server development is prototyping an application in a small, two-tier environment, and then scaling up by simply adding more users to the server. This approach will usually result in an ineffective system, as the server becomes over-loaded. To properly scale to hundreds or thousands of users, it is usually necessary to move to three-tier architecture.

8.4.11.2 Three-tier Architecture

Three-tier architecture splits three parts of an application (user interface, business logic, and data management) into three tiers, and introduces a server (or an "agent" or a middle man) between the client and the server. This agent can be made to function in a variety of ways. It can provide translation services (as in adapting a legacy application on a mainframe to a client/server environment), or intelligent agent services (as in mapping a request to a number of different servers, collating the results, and returning a single response to the client), or metering services (as in acting as a transaction monitor to limit the number of simultaneous requests to a given server). This three-tier solution allows the client process to emphasise on the user interface, and the server side to emphasise on data management.

Types of Middleware

Database middleware provides the link between client and server when the client application that accesses data in the server's database is designed to use only one database type.

Remote procedure calls (RPC) middleware is a more general-purpose solution to client/server computing than database middleware which is used to access a wide variety of data resources for use in a single application.

Messaging middleware is an extension to RPC to address failures in the client/server systems. It provides synchronous or asynchronous connectivity between client and server, so that messages can be either delivered instantly or stored and forwarded as needed.

Object middleware is based on object-oriented technology in the name of object request brokers (ORB) model. ORBs package and manage distributed objects through which middleware activities are extended.

Transaction-processing (TP) monitors have evolved into a middleware technology that can provide a single API for writing distributed applications. TP monitors generally come with a robust set of management tools that add mainframe like controls to open distributed environments.

8.4.11.3 An Example: Two-tier Client/Server Database Systems

In two-tier architecture the application logic is with the User Interface in the client or within the database in the server or both. User interface is usually located in the user's desktop environment while the database management services can be in a server which is a more powerful machine that can serve many clients. That is, two-tier Client/server systems are constructed so that the database can reside on a central computer, known as a server, and is shared among the several users. Users access the server through a client or server application. If the client application runs both business logic and the code to display output to the user, it is called as a thick client. This model of having data stored and managed in a central location offers several advantages:

- Each data item is stored in a central location where all users can work with it. As separate copies of the item are not stored on each client, every user sees the same thing.
- Business and security rules can be defined and enforced equally among all users. This is carried out through the use of a database constraints, stored procedures, and triggers.
- A relational database server optimises network traffic by returning only the data an application needs.

  For example, if an application working with a file server needs to display a list of the names of Heads of departments of university, it must retrieve the entire employee file. If the application is working with a relational database server, it sends this command:

```
SELECT first_name, last_name
FROM employees
WHERE emp_title = 'HOD' ;
```

  The relational database sends back only the names of heads of departments, not all the information about all employees of the University.
- Hardware costs can be minimised. As the data is not stored in each client, clients do not have to dedicate disk space for storing data. The clients are not required to spare their power for managing data while server is not required to spare its efforts for displaying data. Evidently, the server can be configured to optimise the disk I/O capacities needed to retrieve data, and clients can be configured to optimise the formatting and display of data retrieved from the server.
- Maintenance tasks such as backup and restoring data are simplified because they can focus on the central server.

8.4.11.4 Three-tier Architecture

In three-tier architecture the application logic or process lives in the middle-tier, it is separated from the data and the user interface (see Figure 8.23). Three-tier systems are more scalable, robust and flexible. In addition, they can integrate data from multiple sources. In three-tier architecture, a middle tier was added between the user system interface client environment and the database management server environment. There are a variety of ways of implementing this middle tier, such as transaction processing monitors, message servers, or application servers. The middle tier can perform queuing, application execution, and database staging. For example, if the middle tier provides queuing, the client can deliver its request to the middle layer and disengage because the middle tier will access the data and return the answer to the client. The most basic type of three-tier architecture has a middle layer consisting of Transaction Processing (TP) Monitor Technology. The TP Monitor Technology is a type of message queuing, transaction scheduling, and prioritisation service where the client connects to the TP monitor (middle tier) instead of the database server. The transaction is accepted by the monitor, which queues it and then takes responsibility for managing it to completion, thus freeing up the client.

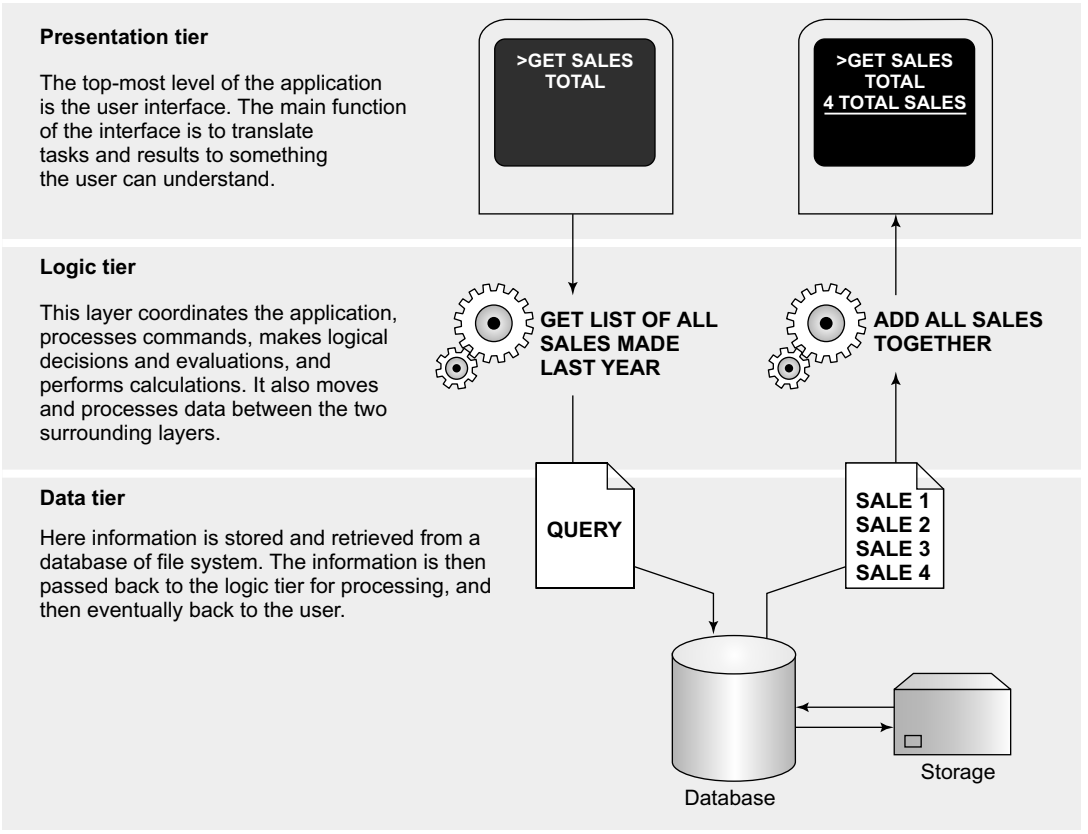


Figure 8.23 A Three-tier Model

8.4.12 JavaScript and Client-side Scripting

JavaScript is a widely used Client-side Scripting Language that runs inside a browser. Inside a normal Web page, we can place some JavaScript code. When the browser loads the page, the browser has a built-in interpreter that reads the JavaScript code in the page and runs it. One of the most common uses of Java Script is to do field validation in a form. For example, the programmer might validate that a person's age entered into a form falls between 1 and 120.

8.4.13 Server-side Scripting

Server-side scripting is a web server technology in which a user's request is verified by running a script directly on the web server to generate dynamic web pages. It is usually used to provide interactive web sites that act as interface to databases or other data stores. This is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript. The primary advantage to server-side scripting is the ability to highly customise the response based on the user's requirements, access rights, or queries into data stores. When the server serves data in a commonly used manner, for example according to the HTTP or FTP protocols, users may have their choice of a number of client programs (most modern web browsers can request and receive data using both of those protocols).

8.4.13.1 PHP and Server-side Scripting

PHP is a general-purpose server-side scripting language originally designed for web development to produce dynamic web pages. It is among one of the first developed server-side scripting languages that is embedded into a HTML source document, rather than calling an external file to process data. Ultimately, the code is interpreted by a web server with a PHP processor module which generates the resulting web page. It can also be used as a command-line interface capability and can be used in standalone graphical applications. PHP can be deployed on most web servers and also as a standalone shell on almost every operating system and platform free of charge.

QUESTIONS

1. **What is middleware and what does it do?**

   Answer: Middleware sits between the application software on the client and the application software on the server. The two functions of middleware are (1) to provide a standard way of communicating that can translate between software from different vendors, and (2) to manage the message transfer between clients and servers so that clients do not need to "know" which server contains the application's data.

2. **How does a "thin" client differ from a "fat" client?**

   Answer: Evidently, "thin" client architecture has little or no application logic in the client at all. A "fat" client approach places all or almost all of the application logic in the client. "Thin" clients are easier to manage. If an application changes, only the server with the application logic needs to be updated.

3. **How does a two-tier client server network differ from a n-tier client server network?**

   Answer: In a two-tiered architecture, the server is responsible for the data and the client handles the application and presentation. An n-tiered architecture uses more than three sets of computers. In this case, the client is responsible for presentation, a database server(s) is responsible for the data access logic and data storage, and the application logic is spread across two or more different sets of servers.

4. **Compare two-tier and three-tier in terms of scalability.**

   Answer: The primary advantage of a three-tiered client-server architecture compared to a two-tiered architecture is that it separates out the processing that occurs to better balance the load on the different servers; it is more "scalable."

5. **What is meant by Horizontal and Vertical scaling?**

   Answer: Horizontal scaling means adding or removing client workstations with only a slight performance impact. Vertical scaling means migrating to a larger and faster server machine or multi-servers.

6. **What are the functions of the typical server program?**

   Answer:

   - It waits for client-initiated requests.
   - Executes many requests at the same time.
   - Takes care of VIP clients first.
   - Initiates and runs background task activity.
   - Keeps running.
   - Can be grown bigger and faster.

7. **What are Super servers?**

   Answer: These are fully-loaded machines which include multiprocessors, high-speed disk arrays for interleaved I/O and fault tolerant features.

8. **Compare two-tier and three-tier architecture.**

Characteristics	two-tier	three-tier
System Administration	Complex	Less Complex
Security	Low	High
Encapsulation of data	Low	High
Performance	Poor	Good
Scale	Poor	Excellent
Application reuse	Poor	Excellent
Ease of development	High	Getting Better
Server to server infrastructure	No	Yes
Legacy application Integration	No	Yes
Internet support	Poor	Excellent
Heterogeneous database support	No	Yes
Rich communication scale	No	Yes
Hardware architecture flexibility	Limited	Excellent
Availability	Poor	Excellent

9. **What are the advantages of component based application?**

   Answer:

   - You can develop big applications in small steps.
   - You can reuse components.
   - Clients can access data and function easily and safely.
   - Custom application can incorporate off the self components.

10. **When should you use three-tier?**

    Answer:

    - Many application services or classes.
    - Applications programmed in different languages or written by different organisations.
    - Two or More heterogeneous data sources-such as two different DBMS or one DBMS and a file system.
    - An application life that is longer than 3 years.
    - A high volume workload—more than 50,000 transactions per day or more than 300 concurrent users on the same system accessing the same database.
    - Significant inter–application communication-including inter–enterprise communication such as EDI.
    - The expectation that the above conditions will be met soon and then the application will have to be scaled.

11. What is the role of Client?

Answer:

- Handle the user interface.
- Translate the user's request into the desired protocol.
- Send the request to the server.
- Wait for the server's response.
- Translate the response into "human-readable" results.
- Present the results to the user.

OBJECTIVE TYPE QUESTIONS

1. A centralised system is one in which the components of an information system are located on multiple locations that are connected through a computer network. (Y/N)

2. A thin client is a personal computer that does not have to be very powerful (or expensive) in terms of processor speed and memory because it is supposed to present only the interface (screens) to the user. (Y/N)

3. A fat client is more powerful (in terms of processor speed, memory, and storage capacity) than a thin client, thus it is used as a server. (Y/N)

4. A fat client is more powerful (in terms of processor speed, memory, and storage capacity) than a thin client, thus it is used in two-tier C/S systems. (Y/N)

5. An application server hosts application logic and services and communicates with the clients (for presentation) and at the back end with database servers for data access and update. (Y/N)

6. In a distributed presentation client/server system application logic, data manipulation and data layers are

A. in server B. in client
C. in middleware D. None

7. In contrast to file server systems, in distributed data client/server systems server executes all data manipulation commands to create, read, update and delete records on a server. (Y/N)

8. In distributed data and application client/server system, application logic is available in __.

A. Server B. Fat client
C. Thin client D. None

9. Clients in three-tier will be fatter than clients in two-tier. (Y/N)

10. Data replication, duplication and partitioning are the same. (Y/N)

11. Middleware products force the programmers to be aware of the underlying communication protocols. (Y/N)

12. Presentation middleware is for

A. Dumb terminal
B. Client
C. Desktop GUI or web browser
D. None

13. Application middleware which is essential in multi-tier applications enables two programmer-written processes on different processors to communicate with one another in whatever way is best suited to the overall application. (Y/N)

14. A multi-user computer that hosts all the components of an information system is a

A. Distributed system
B. Communication system
C. Enterprise resource system
D. Centralised system
E. None of these

15. _____ layer actually gives actual user interface.

A. Presentation
B. Presentation logic
C. The data
D. Application

16. _____ layer that implements processing that must be done to generate the user interface.

A. Presentation B. Presentation logic
C. The data D. Application

17. The data and data manipulation layers are for the same purpose. (Y/N)

18. Distributed presentation C/S is the one in which the presentation and presentation logic layers are shifted from the server of a legacy system to the client. (Y/N)

19. Data partitioning distributes rows and columns of a relational database to specific database servers with no duplication between servers. (Y/N)

20. Find the odd man out with respect to datastore.

A. Database table
B. A database
C. A computer file
D. A transport protocol
E. None

21. Find the odd man out.

A. XML B. HTML
C. ODBC D. CSS

22. A "fat" client approach places most of the application logic on the client. (Y/N)

23. The one which does not have any hard disk.

A. Mainframe
B. Cluster
C. Network computer
D. Mini-computer

24. If the client is a thin client then the server becomes a fat a server while if the client becomes a fat client then the server becomes a thin server. (Y/N)

ANSWER KEY

1. N	**2.** Y	**3.** Y	**4.** N
5. Y	**6.** A	**7.** Y	**8.** A
9. N	**10.** N	**11.** N	**12.** C
13. Y	**14.** D	**15.** A	**16.** B
17. N	**18.** Y	**19.** Y	**20.** D
21. C	**22.** Y	**23.** C	**24.** Y

8.5 Introduction to J2EE

Because of its excellent cross platform support, Java language became the language of choice for middleware development; however there were issues in the development of middleware as there was no consistent standard for the development of services among the vendors. In addition, each vendor provided a different set of services for the applications. Thus, in 1999, Java Development Kit (JDK) was split into: Java 2 Standard Edition (J2SE), Java 2 Enterprise Edition (J2EE), and the Java 2 Micro Edition (J2ME). J2EE is built on the J2SE and includes a number of APIs; the most important and the ones most frequently associated with J2EE staudard are EJBs (Enterprise JavaBeans), JSPs (Java Server Pages), and Servlets. The APIs or packages that are commonly used are identified in the Table and explained in the following sections. A J2EE application may use additional APIs or tools to create a distributed application.

8.5.1 Core J2EE Packages

API	Description
JDBC	Provides connectivity to relational databases.
JNDI	Provides Java access to naming services (LDAP, Windows Registry).
XML Processing (JAXP)	Provides for manipulation of XML documents.
Web Services (JAXM)	Provides the ability to send and receive XML messages using protocols such as SOAP.
JSSE	Allows secure SSL communications both as a server and a client.
JCE	Allows common encryption techniques to be used with Java applications.
RMI	Allows Java objects to be invoked remotely using the Remote Method Invocation (RMI) protocol.
Servlets	Provides a Web tier component using the HTTP protocol.
JMS	The Java Messaging Service provides access to message queues and topics both as a client and a server.
JSPs	Provides a scripting language for insertion into HTML documents.
Java-IDL	Allows Java applications to interact with CORBA servers.
EJBs	Allows the use of business tier components operating within an abstract container that provides a number of services.
JavaMail	Allows access to email servers using common protocols such as POP3 and IMAP.

8.5.1.1 JDBC

Java Database Connectivity (JDBC) API is an important part of J2EE applications that provides an open, vendor-independent API for accessing relational databases.

8.5.1.2 Java Naming and Directory Interface (JNDI)

This package provides consistent access to a variety of naming services that provide for storage and access of various objects. They are effectively lightweight databases with very specific uses.

8.5.1.3 JAXP

The JAXP package contains a number of classes that provide parsers and transformation services for Java applications. The Extensible Markup Language (XML) has become the de facto standard for information interchange. This allows data to be stored in a simple text file using tags that are easy to read and understand. Parsers and transformers are used to read XML documents and transform them into other formats. This allows the development of service components that create XML documents for a number of different clients. The client application can access the XML document and transform it in the format they want.

8.5.1.4 JAXM

The Java for XML Messaging (JAXM) package contains classes and interfaces that provide access to Web Services. For instance, this package allows us to develop SOAP clients and servers. Web services combine the data interchange capabilities of XML and the openness of the HTTP protocol to create a service delivered over the Web.

8.5.1.5 RMI

The RMI (Remote Method Invocation) package allows an object to be created which exposes specific methods to remote clients and also provides for client access to these remote methods. A number of J2EE application servers provide access to their EJB components via RMI. RMI also allows a client to provide access to its methods as a call back service to the server.

8.5.1.6 Java Servlets and JSPs

The Servlet was the original Java Web component and is a Java class implementation that is invoked and run within a container. The container (operating in a Web server) provides various services for the servlet, such as lifecycle management and security, whereas JSPs are used for Web page scripting, allowing Java code fragments to be embedded in an HTML page. The JSP is converted into a servlet and runs within the servlet container.

8.5.1.7 JavaMail

JavaEmail package is used by Java applications for a variety of purposes: for example, to send a warning message to an administrator or user or to transmit data to another application. JavaEmail supports common protocols, such as POP3 and IMAP.

8.5.1.8 JMS

The JMS package provides access to asynchronous messaging services.

8.5.1.9 EJB

The EJB (Enterprise Java Beans) is the middleware component of J2EE. This component also runs in a container like servlet. The services for the EJB are provided by the application server and are often significant. Application servers are usually, but not always, used for applications that expect a high usage load and/or must be highly available with very little downtime. EJBs provide a development paradigm that isolates business logic into easily accessible components, which can be accessed by Web tier components, such as JSPs or servlets, or even directly by client tier components such as a Swing GUI.

8.5.1.10 Java-IDL

Java-IDL provides the ability for Java applications to interact with Common Object Request Broker Architecture (CORBA) components written in any language. Applications using Java-IDL can invoke operations on remote services using the OMG IDL (Object Management Group's Interface Definition Language).

8.5.2 Why to use J2EE

8.5.2.1 Low level services are already implemented

An enterprise application needs to implement very complex services to be successful. Some of those services are transaction and state management, resource pooling and multi-threading. J2EE architecture separates low level services from the application logic.

8.5.2.2 J2EE is well documented and understood

J2EE is developed by a consortium formed by several major companies in the industry. For more information on this consortium you can search for "Java Community Process".

8.5.2.3 J2EE is a standardised and reliable software architecture

Using standardised and reliable software architecture in your development will most likely decrease your future costs and ensure longevity of your applications.

8.5.2.4 J2EE gives you a lot of flexibility

We can deploy J2EE application wherever (in any application server) we want.

8.5.2.5 APIs used in J2EE are well documented

Several APIs are used to implement low level details of enterprise applications. As those APIs are already written and well documented, this will save our time.

8.5.3 J2EE Platform Architecture

J2EE platform uses a multi-tiered distributed application model. Application logic is divided into components and each component is installed on a specific machine based on which tier that component resides (Figure 8.24).

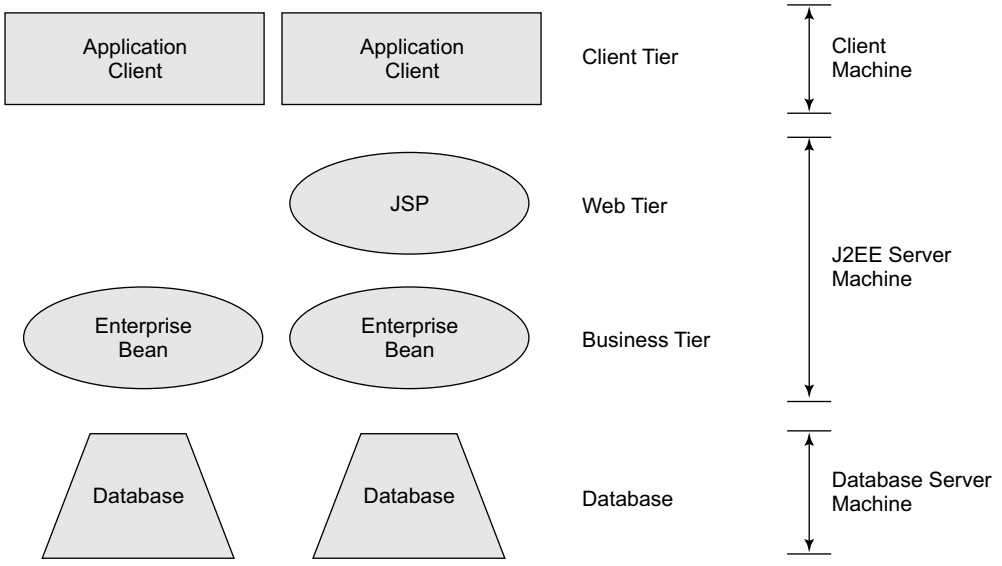


Figure 8.24 J2EE platform Architecture

8.5.3.1 Client tier

In the client tier, Web components, such as Servlets and JavaServer Pages (JSPs), or standalone Java applications provide a dynamic interface to the middle tier.

Client tier can have two types of components: **web client** or **application client**.
Web clients access the components in the web tier namely servlets or java server pages (jsp). Web browsers as a web client are generally used to access web tier components.
Application clients are standalone applications that do not run in browsers (e.g. swing application). They directly access the components in the business tier (Figure 8.25).

8.5.3.2 Middle tier

In the server tier, or middle tier, enterprise beans and Web Services encapsulate reusable, distributable business logic for the application. These server-tier components are contained on a J2EE Application Server, which provides the platform for these components to perform actions and store data.

8.5.3.3 Enterprise data tier

In the data tier, the enterprise's data is stored and persisted, typically in a relational database.

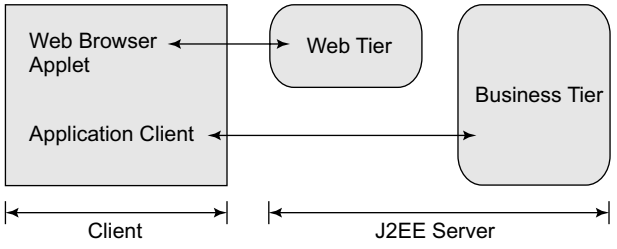


Figure 8.25 Interaction between Client, Web and Business Tiers in J2EE

J2EE applications comprise of components, containers, and services.

COMPONENTS are application-level components.

8.5.4 J2EE Components

J2EE applications are made up of components where a component refers to a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and communicates with other components. The J2EE specification defines the following J2EE components:

- Application clients and applets are client components.
- Java Servlet and Java Server Pages (JSP) technology components are web components.
- Enterprise JavaBeans (EJB) components (enterprise beans) are business components.

Evidently, J2EE components are written in the Java programming language and compiled. These J2EE components are assembled into a J2EE application, verified that they are well-formed and in compliance with the J2EE specification, and deployed to production where they are run and managed by the J2EE server.

8.5.4.1 Client Components

We can have a J2EE application can be web-based or non-web-based. An application client executes on the client machine for a non-web-based J2EE application, while a web browser downloads web pages and applets to the client machine for a web-based J2EE application.

8.5.4.2 Application Clients

An application client runs on a client machine and provides a way for users to handle tasks such as J2EE system or application administration. Application clients directly access enterprise beans running in the business tier. However, if the J2EE application client requirements warrant it, an application client can open an HTTP connection to establish communication with a servlet running in the web tier.

8.5.4.3 Web Browsers

The user's web browser downloads static or dynamic Hypertext Markup Language (HTML), Wireless Markup Language (WML), or Extensible Markup Language (XML) web pages from the web tier. Dynamic web pages are generated by servlets or JSP pages running in the web tier.

8.5.4.4 Enterprise Application Components

These components provide a top-level structure which contains any number of other J2EE components (except other J2EE Enterprise Applications). The JAR file holding an enterprise application has the suffix .ear, so Enterprise Applications are frequently referred to as EARs.

8.5.4.5 Web Application Components

These J2EE components may contain Servlets, Java ServerPages, Tag Library Definitions and plain Java classes or resources as required for proper operation. The documents and Java classes of Web Applications frequently create markup language (HTML, WML, XML, etc.) responses to requests from web browsers or business applications. The JAR file holding a Web application has the suffix .war, so WebApps are frequently referred to as WARs.

8.5.4.6 Enterprise JavaBean Application Components

These components hold server-side business logic packaged within Enterprise JavaBeans.

8.5.4.7 Resource Application Components

These J2EE components hold Java classes that act as drivers or communication gateways to Enterprise Information Systems (EISs). It is highly likely that the manufacturer of an EIS packages its driver into a J2EE resource archive (RAR).
CONTAINERS are the interface between a component and the low-level platform-specific functionality that supports the component. The container also manages non-configurable services such as enterprise bean and servlet life cycles, database connection resource pooling, data persistence, and access to the J2EE platform APIs.
Container Types
J2EE server is the runtime portion of a J2EE product. A J2EE server provides EJB and web containers (Figure 8.26).

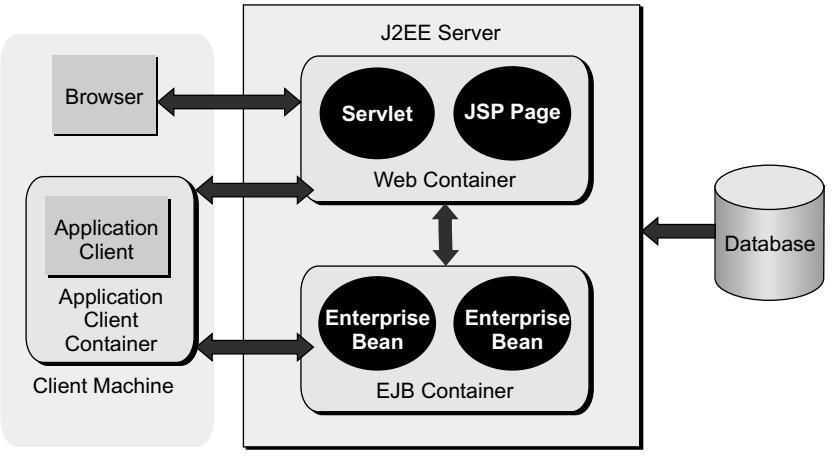


Figure 8.26 J2EE system with EJB

Enterprise JavaBeans (EJB) container manages the execution of enterprise beans for J2EE applications. Enterprise beans and their container run on the J2EE server.
Web container manages the execution of JSP page and servlet components for J2EE applications. Web components and their container run on the J2EE server.
Application client container manages the execution of application client components. Application clients and their container run on the client.

Applet container manages the execution of applets. This consists of a web browser and Java Plug-in running on the client together.

8.5.6 J2EE Standard Services

The J2EE standard services J2EE actually provided by J2SE. Some of the J2EE services are HTTP, Java IDL, JDBC, JMS, JNDI, JavaMail, JAXP.

8.5.7 J2EE Platform Benefits

The J2EE platform offers several benefits:

- Simplified architecture and development
- Scalability to meet demand variations
- Integration with existing information systems
- Choice of servers, tools, components
- Flexible security model

8.5.8 Model-View-Controller (MVC)

MVC is an architectural pattern used in software engineering. In complex computer applications that present a large amount of data to the user, a developer often wishes to separate data (model) and user interface (view) concerns, so that changes to the user interface will not affect data handling, and that the data can be reorganised without changing the user interface. The model-view-controller solves this problem by decoupling data access and business logic from data presentation and user interaction, by introducing an intermediate component: the controller.

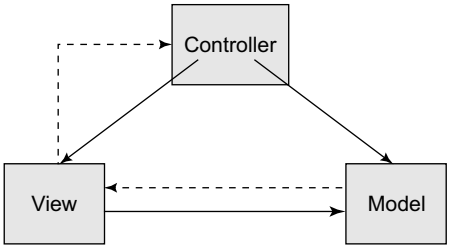


Figure 8.27 MVC Architecture

A simple diagram that depicts the relationship between the Model, View, and Controller is given above (Figure 8.27). Note that the solid lines indicate a direct association, and the dashed lines indicate an indirect association. This pattern was first described in 1979 by Trygve Reenskaug, then working on Smalltalk at Xerox research labs.

8.5.8.1 Pattern Description

It is common to split an application into separate layers: presentation (UI), domain logic, and data access. In MVC the presentation layer is further separated into view and controller. MVC encompasses more of the architecture of an application than is typical for a design pattern.

8.5.8.2 Model

This contains the **domain**-specific representation of the information that the application operates. Domain logic adds meaning to raw data (e.g., calculating whether today is the user's birthday, or the totals, taxes, and shipping charges for shopping cart items). Many applications use a persistent storage mechanism (such as a database) to store data. MVC does not specifically mention the data access layer because it is understood to be underneath or encapsulated by the Model.

8.5.8.3 View

View renders the model into a form suitable for interaction, typically a user interface element. Multiple views can exist for a single model for different purposes.

8.5.8.4 Controller

Controller processes and responds to events, typically user actions, and may invoke changes on the model.

MVC is often seen in web applications, where the view is the actual HTML page, and the controller is the code that gathers dynamic data and generates the content within the HTML. Finally, the model is represented by the actual content, usually stored in a database or XML files. Though MVC comes in different flavors, control flow generally works as follows:

The user interacts with the user interface in some way (e.g., presses a button).

A controller handles the input event from the user interface, often via a registered handler or callback.

The controller accesses the model, possibly updating it in a way appropriate to the user's action (e.g., controller updates user's shopping cart).

A view uses the model (indirectly) to generate an appropriate user interface (e.g., the view produces a screen listing the shopping cart contents). The view gets its own data from the model.

The model has no direct knowledge of the view.

The user interface waits for further user interactions, which begins the cycle anew.

By decoupling models and views, MVC helps to reduce the complexity in architectural design, and to increase flexibility and reuse.

Exemplary MVC implementations in selected languages includes Java Enterprise Edition (Java EE)

8.5.9 Java Enterprise Edition (Java EE)

Unlike the other frameworks, Java EE defines a pattern for model objects.

8.5.9.1 Model

The model is commonly represented by entity beans, although the model can be created by a servlet using a business object framework such as Spring.

8.5.9.2 View

The view in a Java EE application may be represented by a Java Server Page, which may be currently implemented using JavaServer Faces Technology (JSF). Alternately, the code to generate the view may be part of a servlet.

8.5.9.3 Controller

The controller in a Java EE application may be represented by a servlet.

8.5.9.4 MVC Benefits

- MVC separates design concerns (data persistence and behavior, presentation, and control), decreasing code duplication, centralising control, and making the application more easily modifiable.
- MVC also helps developers with different skill sets to focus on their core skills and collaborate through clearly defined interfaces. For example, a J2EE application project may include developers of custom tags, views, application logic, database functionality, and networking.
- An MVC design can centralise control of application facilities such as security, logging, and screen flow.
- New data sources are easy to add to an MVC application by creating code that adapts the new data source to the view API. Similarly, new client types are easy to add by adapting the new client type to operate as an MVC view.
- MVC clearly defines the responsibilities of participating classes, making bugs easier to track down and eliminate.

QUESTIONS

1. **What is a thin-client application?**

   Answer: A thin client does not process data, but instead sends the data to an enterprise bean for processing.

2. **What is a J2EE component?**

   Answer: A J2EE component is a self-contained, functional software unit that is assembled into a J2EE application and interfaces with other application components. The J2EE specification defines the following application components:

   - Application client components
   - Enterprise JavaBeans components
   - Servlets and JavaServer Pages components (also called Web components)
   - Applets

3. **What advantage does an entity bean have over a session bean?**

   Answer: Entity bean data is persistent because it survives crashes. If a crash occurs while the data in an entity bean is being updated, the entity bean data is automatically restored to the state of the last committed database transaction. If the crash occurs in the middle of a database transaction, the transaction is backed out to prevent a partial commit from corrupting the data.

4. **When would you use a session bean?**

   Answer: You would use a session bean to process non-persistent data that represents a transient conversation with a client.

5. **Why would you design a J2EE application so user data is entered by way of a JSP page and managed by an underlying JavaBeans class?**

   Answer: Separating how the data is presented from how the data is managed makes an application much easier to update, maintain, and manage. For example, the person who maintains the JSP page does not have to know Java programming language, and the person who maintains the JavaBean code does not have to understand user interface design.

6. **Why is XML a good way to transfer text-based data from one program or tool to another ?**

   Answer: XML tags represent and describe data. Any tool or program that can read the tags, can handle the data based on what the tags mean. For example, a company might use XML to produce reports so different parties who receive the reports can handle the data in a way that best suits their needs. One party might put the XML data through a program to translate the XML to HTML so it can post the reports to the web; another party might put the XML data through a tool to produce a stockholder booklet; and yet another party might put the XML data through a tool to create a marketing presentation. These highly flexible and cost-effective capabilities are available through XML tags, Document Type Definitions (DTDs) also known as XML schemas, and XML APIs.

7. **What part of the J2EE platform handles data storage and retrieval on behalf of an entity bean?**

   Answer: The container handles data storage and retrieval on behalf of a container-managed entity bean. This is called container-managed persistence, and means that you do not have to write any SQL code to send data to or retrieve data from the database because the container handles all of this for you.

8. **What is bean-managed persistence?**

   Answer: Bean-managed persistence is when you override container-managed persistence and implement entity or session bean methods to use the SQL commands you provide. Bean-managed persistence can be useful if you need to improve performance or map data in multiple beans to one row in a database table.

9. **How are life cycle methods called?**

   Answer: A bean's container calls its life cycle methods. The container is the interface between a bean and the low-level platform-specific functionality that supports the bean.

10. **In a multi-tiered application, which tier is the browser in?**

    Answer: The browser (or thin client) is in the first tier of a multi-tiered application.

OBJECTIVE TYPE QUESTIONS

1. Which language is used for middleware development?

   A. XML B. Java
   C. C# D. C++

2. JDBC provides open, vendor independent API for accessing relational databases in J2EE development. (Y/N)

3. JAXM and JAXP are both used for XML processing. (Y/N)

4. Middleware component of J2EE
A. Java B. EJB
C. Servlet D. None

5. Related to Web tier
A. Java B. JSP
C. EJB D. Applet

ANSWER KEY

1. B **2.** Y **3.** N **4.** B
5. B

8.6 Introduction to JSP

Java Server Pages (JSP) is Java based technology for the development of dynamic web sites. Evidently, JSP files are also HTML files with special tags containing Java source code that extends the dynamic content. The Web server that supports JSP files may be connected to a database also. In a nutshell, JSP source code runs in the web server in the JSP Servlet Engine which in turn dynamically generates the HTML and sends the HTML output to the client's web browser.

Main reasons for using JSP

- Multi platform functionality
- Component reuse by using Java beans and EJB. We can continue to enjoy the advantages of Java.
- We can take one JSP file and move it to another platform, web server or JSP Servlet engine. This means we are not locked with one vendor or platform.

HTML and graphics displayed in the web browser are classed as the presentation layer. The Java code (JSP) on the server is classed as the implementation layer. By having a separation of presentation and implementation, web designers will be made to work only on the presentation aspects while Java developers will be made to concentrate on implementing the application.

8.6.1 JSP ARCHITECTURE

JSPs are essentially special tags embedded in HTML pages. These JSP tags can contain Java code. The JSP file extension is .jsp rather than .htm or .html. The JSP engine parses the .jsp and creates a Java Servlet source file. It then compiles the source file into a class file, which is only done the first time and that is why the JSP is probably slower during the first time it is accessed. Any time after this, the special compiled servlet is executed and is thus faster.

Steps required for a JSP request:

1. The user goes to a web site made using JSP. The user goes to a JSP page (whose file name is ending with .jsp). The web browser makes the request via the Internet.
2. The JSP request is sent to the Web server.
3. The Web server recognises that the file required is special (.jsp), therefore it passes the JSP file to the JSP Servlet Engine.
4. If the JSP file has been called the first time, the JSP file is parsed, otherwise goes to step 7.
5. A special Servlet is generated from the JSP file. All the HTML statements that are required are converted to println statements.
6. The Servlet source code is compiled into a class.
7. The Servlet is instantiated, calling the init and service methods.
8. HTML form, the Servlet output is sent via the Internet.
9. HTML results are displayed on the user's web browser.

Figure 8.28 demonstrates the actions involved with JSP. JSP's life cycle can be grouped into following phases.

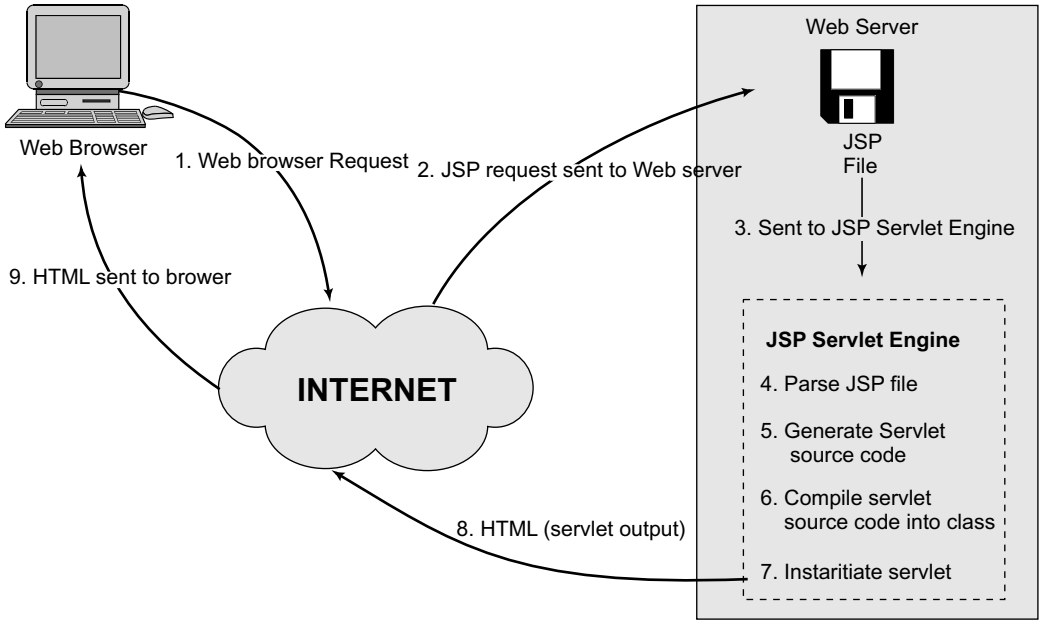


Figure 8.28 Working of JSP based web service

1. **JSP Page Translation:** A java servlet file is generated from the JSP source file. This is the first step in its tedious multiple phase life cycle. In the translation phase, the container validates the syntactic correctness of the JSP pages and tag files. The container interprets the standard directives and actions, and the custom actions referencing tag libraries used in the page.
2. **JSP Page Compilation:** The generated java servlet file is compiled into a java servlet class.
3. **Class Loading:** The java servlet class that was compiled from the JSP source is loaded into the container.
4. **Execution phase:** In the execution phase the container manages one or more instances of this class in response to requests and other events. The interface JSP Page contains jspInit() and jspDestroy(). The JSP specification has provided a special interface HttpJspPage for JSP pages serving HTTP requests and this interface contains _jspService().
5. **Initialisation:** jspInit() method is called immediately after the instance is created. It is called only once during JSP life cycle.
6. **_jspService() execution:** This method is called for every request of a JSP during its life cycle. This is where it serves the purpose of creation. Of course, it has to pass through all the above steps to reach this phase. It passes the request and the response objects. This method cannot be overridden.
7. **jspDestroy() execution:** This method is called when this JSP is destroyed. With this call the servlet serves its purpose and submits itself to heaven (garbage collection). This is the end of jsp life cycle. The jspInit(), _jspService() and jspDestroy() are called the life cycle methods of the JSP.

As explained above, when someone visits a JSP page, the HTML web page is generated and sent back to the visitor. In order to explain the conceptual difference between static HTML pages and dynamic JSP pages, consider the following HTML source code and its output **"The date today is".** In order for a static web page to show today's date, we need to edit the web page every day and upload to the web server. This is very time consuming for such a simple task. This is where the dynamic generation of web pages shows their usefulness. In the JSP code, special syntax is used to signify that the current date needs to be displayed. This special syntax is processed on the web server and sent back to the visitor as a normal (html) web page.

HTML today.htm	**Output**
<HTML> <BODY> The date today is : <BODY> <HTML>	The date today is :

JSP today.htm	**Output**
<HTML> <BODY> The date today is : <%= new java, util.Date()%> <BODY> <HTML>	The date today is : Sun Sep 30 15:05:35 BST 2006

Creating first JSP page

```
<html>
<head>
<title>My first JSP page </title>
</head>
<body>
<%@ page language="java" %>
<% out.println("Hello World"); %>
</body>
</html>
```

We can type the above code into a text file with the name say **helloworld.jsp** and place this in the correct directory of our JSP web server and invoke it via our browser.

8.6.2 Using JSP Tags

There are five main tags that are used in JSP. They are given as:

1. Declaration tag
2. Expression tag
3. Directive tag
4. Scriptlet tag
5. Action tag

We shall learn about these JSP tags with suitable examples in the following pages. However, the use of each tag is summarised below.

1. **Directives:** In the directives we can import packages, define error handling pages or the session information of the JSP page.
2. **Declarations:** This tag is used for defining the functions and variables to be used in JSP.
3. **Scriplets:** In this tag we can insert any amount of valid java code and these codes are placed in *_jspService* method by the JSP engine.
4. **Expressions:** We can use this tag to output any data on the generated page. These data are automatically converted to string and printed on the output stream.
5. **Action Tag:** There are three main roles of action tags :
   - To enable the use of server side Javabeans
   - To transfer control between pages
   - For browser independent support for applets.

8.6.2.1 Directive Tags

Syntax of JSP directive tags are given as:

```
<%@directive attribute="value" %>
```

Where directive can be:

1. **page:** page is used to provide the information about it.

■ **Example** `<%@page language="java" %>`

2. **include:** include is used to include a file in the JSP page.

■ **Example** `<%@ include file="/header.jsp" %>`

3. **taglib:** taglib is used to use the custom tags in the JSP pages (custom tags allow us to define our own tags).

■ **Example** `<%@ taglib uri="tlds/taglib.tld" prefix= "mytag" %>`

and attribute can be:

1. **language="java".** This tells the server that the page is using the java language.

■ **Example** `<%@page language="java" %>`

2. **extends="mypackage.myclass".** This attribute is used when we want to extend any class. We can use comma(,) to import more than one package.

■ **Example** `<%@page language="java" import="java.sql.*,nbvpackage.myclass" %>`

3. **session="true".** When this value is true, session data is available to the JSP page otherwise not. By default this value is true.

■ **Example** `<%@page language="java" session="true" %>`

4. **errorPage="error.jsp".** This is used to handle the un-handled exceptions in the page.

■ **Example** `<%@page language="java" session="true" errorPage="error.jsp" %>`

5. **contentType="text/html;charset=ISO-8859-1".** We have to use this attribute to set the mime type and character set of the JSP.

■ **Example** `<%@page language="java" session="true" contentType="text/html;charset=ISO-8859-1"%>`

More about Page directive

This directive has the following optional attributes (refer to Table below) that provide the JSP Engine with special processing information.

Attribute	Description	Example
language	**Which language the file uses.**	**<%@ page language = "java" %>**
extends	Superclass used by the JSP engine for the translated Servlet.	<%@ page extends = "com.taglib... %>
import	Import all the classes in a java package into the current JSP page. This allows the JSP page to use other java classes.	<%@ page import = "java.util.*" %>
session	Does the page make use of sessions. By default all JSP pages have session data available. There are performance benefits to switching session to false.	Default is set to true.
buffer	Controls the use of buffered output for a JSP page. Default is 8kb	<%@ page buffer = "none" %>
autoFlush	Flush output buffer when full.	<%@ page autoFlush = "true" %>
isThreadSafe	Can the generated Servlet deal with multiple requests? If true a new thread is started so requests are handled simultaneously.	
info	Developer uses info attribute to add information/document for a page. Typically used to add author, version, copyright and date info.	<%@ page info = "visualbuilder.com test page,copyright 2001. " %>
errorPage	Different page to deal with errors. Must be URL to error page.	<%@ page errorPage = "/error/error.jsp" %>
IsErrorPage	This flag is set to true to make a JSP page a special Error Page. This page has access to the implicit object exception (see later).	
contentType	Set the mime type and character set of the JSP.	

More about Include directive

With the help of include directive, a JSP developer can include contents of a file inside another. Typically include files are used for navigation, tables, headers and footers that are common to multiple pages. For example, the following includes privacy.html found in the include directory into the current jsp page.

```
<%@ include file = "include/privacy.html" %>
```

More about Tag Lib directive

A tag lib is a collection of custom tags that can be used by the page. Custom tags were introduced in JSP 1.1 and allow JSP developers to hide complex server side code from web designers.

```
<%@ taglib uri = "tag library URI" prefix = "tag Prefix" %>
```

8.6.2.2 Declarative Tags

Syntax of JSP Declaratives are given as:

```
<%!
//java code
%>
```

JSP Declaratives begins with <%! and ends with %> .We can embed any amount of java code in the JSP Declaratives. Variables and functions defined in the declaratives are class level and can be used anywhere in the JSP page. Code placed in this tag must always end with a semicolon (;). Declarations do not generate output so are used with JSP expressions or scriptlets.

■ **Example** The following code declares a variable counter and a method setAccount.

```
<%!
     private int counter = 0 ;
     private String setAccount (int accountNo);
%>
```

8.6.2.3 Scriplet Tags

Syntax of JSP **Scriptles** is as follows.

```
<%  //java code   %>
```

We can embed any amount of java code in the JSP Scriptlets. JSP Engine places this code in the *_jspService()* method. Variables available to the JSP Scriptlets are:

- **request:** request represents the clients request and is a subclass of HttpServletRequest. We can use this variable to retrieve the data submitted along with the request.

■ **Example**

```
<%
//java code
String userName=null;
userName=request.getParameter("userName");
%>
```

- **response:** response is subclass of *HttpServletResponse.*
- **session:** session represents the HTTP session object associated with the request.
- **out:** out is an object of output stream and is used to send any output to the client.
- Other variables available to the scriptlets are *pageContext, application, config and exception.*

The scriplet code can access any variable or bean that has been declared. For example, to print a variable.

```
<%
        String username = "visualbuilder";
        out.println ( username ) ;
  %>
```

8.6.2.4 Expression Tags

Syntax of the JSP Expressions are:

```
<%=     "Anything"  %>
```

That is, anything between <%, and %> is considered as an expression and the same will be converted into a String which will be displayed.

■ **Example**

```
<%="Hello JSP World!" %>
```

Above code will display 'Hello JSP World!'.
Expression tag behaves as System.out.println(). A semicolon (;) does not appear at the end of the code inside the tag. For example, to show the current date and time.

```
Date : <%= new java.util.Date() %>
```

8.6.2.5 Action tags

Action tags are mainly used :

(1) to enable the use of server side Javabeans
(2) to transfer control between pages
(3) to have browser independent support for applets.

The following are some of the action tags:

- **jsp:include** This action works as a subroutine; the Java servlet temporarily passes the request and response to the specified JSP/Servlet. Control is then returned back to the current JSP page.
- **jsp:param** This action is used to add the specific parameter to current request. This tag can be used inside a jsp:include, jsp:forward or jsp:params blocks.
- **jsp:forward** This tag is used to hand off the request and response to another JSP or servlet. Do remember that the request never returns to the calling JSP page.
- **jsp:plugin** This tag actually generates the appropriate HTML code to embed the Applets correctly. With older browsers this may be needed.
- **jsp:fallback** This tag is used to specify what message is to be shown on the browser if the applet is not supported by the browser.

■ **Example**

```
<jsp:fallback>
      <p>Sorry Unable to load applet</p>
  </jsp:fallback>
```

- **jsp:getProperty** This action is used to get specified property from the JavaBean object.
- **jsp:setProperty** This tag is used to set a property in the JavaBean object.
- **jsp:useBean** This tag is used to instantiate an object of Java Bean or it can re-use existing java bean object.

8.6.3 Javabeans

A Javabean is a special type of class that has a number of methods which a JSP page can call. For example, to make a feedback form that automatically sends emails to visitors, create a JSP page with a form. When the visitor presses the submit button the browser sends the details to a Javabean that sends out the email. This way there would be no code in the JSP page dealing with sending emails (JavaMail API) and our Javabean could be used in another page (promoting reuse). To use a Javabean in a JSP page use the following syntax:

```
<jsp : usebean id = " ...." scope = "application" class = "com..." />
```

The following is a list of Javabean scopes:

page - valid until page completes.

request - bean instance lasts for the client request

session - bean lasts for the client session

application - bean instance created and lasts until application ends.

8.6.4 Simple JSP Examples

8.6.4.1 Creating a Second JSP Page

This example is used to illustrate how to declare variables in JSP page. Here, two variable are used, one string and an integer x that displays HAPPY NEW YEAR a number of times when the page is accessed.

```
<HTML> <HEAD>
<!-- Ex2 -->
<TITLE> JSP loop</TITLE>
</HEAD>
<BODY>
<font face=verdana color=blue>
JSP loop
<BR> <BR>
<%!
public String writeThis(int x)
{        String myText="";
        for (int i = 1; i < x; i++ )
        myText = myText "<font size=" i " color=darkred face=verdana>HAPPY NEW YEAR</font><br>" ;
            return myText;    }     %>
This is a loop example from the
<br>
<%= writeThis(8) %>
</font>  </BODY>   </HTML>
```

8.6.4.2 Implicit Objects

These objects are automatically available in JSP. The implicit objects are:

Variable	Of type
Request	Javax.servlet.http.httpservletrequest
Response	Javax.servlet.http. httpservletresponse
Out	Javax.servlet.jsp.JspWriter
Session	Javax.servlet.http.httpsession
PageContent	Javax.servlet.jsp.pagecontext
Application	Javax.servlet.http.ServletContext
Config	Javax.servlet.http.ServletConfig
Page	Java.lang.Object

Page object Represents the JSP page and is used to call any methods defined by the servlet class.

Config object Stores the Servlet configuration data.

Request object Access to information associated with a request. This object is normally used in looking up parameter values and cookies.

```
<% String devStr = request.getParameter("dev"); %>
```

Development language = <%= devStr %>

This code snippet stores the parameter "dev" in the string devStr.

8.6.4.3 Creating a Form

Let us learn how to create a form and process an html form. Copy the code below and place in a file named: myform.jsp. Go to myform.jsp in our browser. You will see the form you just created.

```
<html> <head>
<!–Ex3 -->
<title>Sample form </title>
</head><body>
<form action="myformconfirm.jsp" method="post">
Enter in a website name:<br>
<input type="text" name="website"><br>
<input type="submit" name="submit">
</form></body></html>
```

8.6.4.4 Processing a Form

Now, we show how to process the html form that is just created. Copy the code below and place in a file named: myform-confirm.jsp. Go. to myform.jsp. Fill in some details and submit the form. We should see the results of our submission

```
<html>   <head>
<!–Ex4-->
<title>Form processing </title>
</head>   <body>
<font size=3>
Your info has been received:
<br><br>
<% String sName = request.getParameter("website");
out.print(sName); %>
</font>   </body> </html>
```

This example shows how to create and process more form elements. Copy the code below and place in a file named: full-form.jsp

```
<html>
<head>
<!-- Ex5 -->
<title>Form with more fields </title>
</head>
<body>
<h1>
Website submission form
</h1>
<form action="fullformconfirm.jsp" method="post">
```

```
Enter in the website name:
<input type="text" name="website">
<br> <br> Enter in the url:
<input type="text" name="url">
<br> <br> category:
<select name="category" size="1">
<option selected value="java">java</option>
<option value="ejb">ejb</option>
<option value="servlet">servlet</option>
<option value="jsp">jsp</option>
<option value="jdbc">jdbc</option>
</select>
<br> <br> Description:
<textarea rows="4" cols='42' name="desc"></textarea>
<br> <br> Search engines:
<input type="checkbox" name="yahoo" value="T">Yahoo
<input type="checkbox" name="google" value="T" CHECKED>Google
<input type="checkbox" name="altavista" value="T">Altavista
<br> <br>
<input type="submit" name="submit" value="Go">
</form>
</body> </html>
```

Now, we have to process the field data from the above page. Copy the code below and place in a file named: fullformconfirm.jsp. Go to fullform.jsp. Fill in some details and submit the form. You should see the results of your submission

```
<html>  <head>
<!-- Example4 -->
<title>processing many fields</title>
</head> <body>
<font size=3> Thank you for your submission,it has been successfully received: <br><br>
<%    String sName = request.getParameter ("website");
String sUrl = request.getParameter("url");
String sCategory = request.getParameter ("category");
String sDesc = request.getParameter("desc");
String sGoogle = request.getParameter("google");
String sYahoo = request.getParameter("yahoo");
String sAltavista = request.getParameter ("altavista");     %>
Name:<%=sName%><br>
Url:<%=sUrl%><br> Desc:<%=sDesc%><br>
```

```
Category:<%=sCategory%><br> Desc:<%=sDesc%><br>
Google:<%=sGoogle%><br>
Yahoo:<%=sYahoo%><br>
Altavista:<%=sAltavista%><br>
</font> </body> </html>
```

8.6.4.5 Getting Client Info

We can get information about a clients computer. Copy the code below and place in a file named: clientinfo.jsp. Run it from our browser. You should see the results of your submission.

```
<html>
<head>
<!—Ex6 -->
<title>Client Info </title>
</head>
<body>
Client computer details:
<br><br>
<b>Ip address</b>:
<br>
<%=request.getRemoteAddr()%>
<br><br>
<b>Computer name</b>:
<br>
<%=request.getRemoteHost()%>
<br><br>
</body>
</html>
```

8.6.5 Advantages and Disadvantages of JSP

8.6.5.1 Advantages

- Simple to understand and develop initially

8.6.5.2 Disadvantages of JSP

- The JSP page is very difficult to maintain. It contains HTML and Java code with queries to the database. The business logic should not be in the JSP; otherwise many pages will have to be changed every time business requirements change.
- Need to have data connectivity code in every JSP page.
- Does not scale up very well.
- Security issues - If a hacker gains access to the web server, all the confidential business logic can be read by opening the JSP files.

QUESTIONS

1. **What are the benefits of using J2EE?**

   Answer: There are several reasons for using the J2EE set of technologies:

- Extensibility and maintainability
- Division of work along skill lines
- Scalability, portability, availability
- Code reuse
- Interoperability – legacy integration
- Focus on implementing business logic
- Separation of code with differing rates of change.

2. **What makes J2EE suitable for distributed multi-tiered Applications?**

   Answer: The J2EE platform uses a multi-tiered distributed application model. Application logic is divided into components according to function, and the various application components that make up a J2EE application are installed on different machines depending on the tier in the multi-tiered J2EE environment to which the application component belongs. The J2EE application parts are:

- Client-tier components run on the client machine.
- Web-tier components run on the J2EE server.
- Business-tier components run on the J2EE server.
- Enterprise information system (EIS)-tier software runs on the EIS server.

3. **What is J2EE?**

   Answer: J2EE is an environment for developing and deploying enterprise applications. The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multi-tiered, web-based applications.

4. **What are the components of J2EE application?**

   Answer: A J2EE component is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and communicates with other components. The J2EE specification defines the following J2EE components:

- Application clients and applets are client components
- Java Servlet and JavaServer Pages technology components are web components
- Enterprise JavaBeans components (enterprise beans) are business components
- Resource adapter components provided by EIS and tool vendors

5. **What do Enterprise JavaBeans components contain?**

   Answer: Enterprise JavaBeans components contain Business code, which is logic that solves or meets the needs of a particular business domain such as banking, retail, or finance, and is handled by enterprise beans running in the business tier. All the business code is contained inside an Enterprise Bean which receives data from client programs, processes it (if necessary), and sends it to the enterprise information system tier for storage. An enterprise bean also retrieves data from storage, processes it (if necessary), and sends it back to the client program.

6. **Does J2EE application sell only a web-based application?**

   Answer: No, It depends on the type of application that the client wants. A J2EE application can be web-based or non-web-based. If an application client executes it on the client machine, it is a non-web-based J2EE application. The J2EE application can provide a way for users to handle tasks such as J2EE system or application administration. It typically has a graphical user interface created from Swing or AWT APIs, or a command-line interface. When user requests, it can open an HTTP connection to establish communication with a servlet running in the web tier.

7. **Are JavaBeans J2EE components?**

   Answer: No. JavaBeans components are not considered J2EE components by the J2EE specification. They are written to manage the data flow between an application client or applet and components running on the J2EE server or between server components and a database. JavaBeans components written for the J2EE platform have instance variables and get and set methods for accessing the data in the instance variables. JavaBeans components used in this way are typically simple in design and implementation, but should conform to the naming and design conventions outlined in the JavaBeans component architecture.

8. **Is HTML page a web component?**

   Answer: No. Static HTML pages and applets are bundled with web components during application assembly, but are not considered web components by the J2EE specification. Even the server-side utility classes are not considered as web components.

9. **What can be considered as a web component?**

   Answer: J2EE Web components can be either servlets or JSP pages. Servlets are Java programming language classes that dynamically process requests and construct responses. JSP pages are text-based documents

that execute as servlets but allow a more natural approach to creating static content.

10. What are containers?

Answer: Containers are the interface between a component and the low-level platform specific functionality that supports the component. Before a Web, enterprise bean, or application client component can be executed, it must be assembled into a J2EE application and deployed into its container.

11. What are container services?

Answer: A container is a runtime support of a system-level entity. Containers provide components with services such as lifecycle management, security, deployment, and threading.

12. What is a web container?

Answer: Servlet and JSP containers are collectively referred to as Web containers. It manages the execution of JSP page and servlet components for J2EE applications. Web components and their container run on the J2EE server.

13. What is Enterprise JavaBeans (EJB) container?

Answer: It manages the execution of enterprise beans for J2EE applications. Enterprise beans and their container run on the J2EE server.

14. What is Applet container?

Answer: It manages the execution of applets. It consists of a Web browser and Java Plugin simultaneously running on the client.

15. How do we package J2EE components?

Answer: J2EE components are packaged separately and bundled into a J2EE application for deployment. Each component, its related files such as GIF and HTML files or server-side utility classes, and a deployment descriptor are assembled into a module and added to the J2EE application. A J2EE application is composed of one or more enterprise bean,Web, or application client component modules. The final enterprise solution can use one J2EE application or be made up of two or more J2EE applications, depending on design requirements. A J2EE application and each of its modules has its own deployment descriptor. A deployment descriptor is an XML document with an .xml extension that describes a component's deployment settings.

16. What are the types of J2EE clients?

Answer: Following are the types of J2EE clients:

- Applets
- Application clients
- Java Web Start-enabled rich clients, powered by Java Web Start technology.
- Wireless clients, based on Mobile Information Device Profile (MIDP) technology.

17. What is deployment descriptor?

Answer: A deployment descriptor is an Extensible Markup Language (XML) text-based file with an .xml extension that describes a component's deployment settings. A J2EE application and each of its modules has its own deployment descriptor. Because deployment descriptor information is declarative, it can be changed without modifying the bean source code. At run time, the J2EE server reads the deployment descriptor and acts upon the component accordingly.

18. What is an EAR file?

Answer: An EAR file is a standard JAR file with an .ear extension, named from Enterprise ARchive file. A J2EE application with all of its modules is delivered in EAR file.

19. What is JTA and JTS?

Answer: JTA is the abbreviation for the Java Transaction API. JTS is the abbreviation for the Jave Transaction Service. JTA provides a standard interface and allows you to demarcate transactions in a manner that is independent of the transaction manager implementation. The J2EE SDK implements the transaction manager with JTS. The code doesn't call the JTS methods directly. Instead, it invokes the JTA methods, which then call the lower-level JTS routines. Therefore, JTA is a high level transaction interface that the application uses to control transaction. And JTS is a low level transaction interface that Enterprise JavaBeans use behind the scenes (client code doesn't directly interact with JTS. It is based on Object Transaction Service (OTS) which is part of CORBA.

20. What is JAXP?

Answer: JAXP stands for Java API for XML. XML is a language for representing and describing text-based data which can be read and handled by any program or tool that uses XML APIs. It provides standard services to determine the type of an arbitrary piece of data, encapsulate access to it, discover the operations available on it, and create the appropriate JavaBeans component to perform those operations.

21. What is J2EE Connector?

Answer: The J2EE Connector API is used by J2EE tools vendors and system integrators to create resource adapters that support access to enterprise information systems that can be plugged into any J2EE product. Each type of database or EIS has a different resource adapter.

NOTE: A resource adapter is a software component that allows J2EE application components to access

and interact with the underlying resource manager. Because a resource adapter is specific to its resource manager, there is typically a different resource adapter for each type of database or enterprise information system.

22. What is JAAP?

Answer: The Java Authentication and Authorisation Service (JAAS) provides a way for a J2EE application to authenticate and authorise a specific user or group of users to run it. It is a standard Pluggable Authentication Module (PAM) framework that extends the Java 2 platform security architecture to support user-based authorisation.

23. What is Java Naming and Directory Service?

Answer: The JNDI provides naming and directory functionality. It provides applications with methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes. Using JNDI, a J2EE application can store and retrieve any type of named Java object. Because JNDI is independent of any specific implementations, applications can use JNDI to access multiple naming and directory services, including existing naming and directory services such as LDAP, NDS, DNS, and NIS.

24. What is Struts?

Answer: Struts is a Web page development framework. Struts combines Java Servlets, Java Server Pages, custom tags, and message resources into a unified framework. It is a cooperative, synergistic platform, suitable for development teams, independent developers, and everyone between.

25. How is the MVC design pattern used in Struts framework?

Answer: In the MVC design pattern, application flow is mediated by a central Controller. The Controller delegates requests to an appropriate handler. The handlers are tied to a Model, and each handler acts as an adapter between the request and the Model. The Model represents, or encapsulates, an application's business logic or state. Control is usually then forwarded back through the Controller to the appropriate View. The forwarding can be determined by consulting a set of mappings, usually loaded from a database or configuration file. This provides a loose coupling between the View and Model, which can make an application significantly easier to create and maintain.

Controller: Servlet controller is supplied by Struts itself; View: what you can see on the screen, a JSP page and presentation components; Model: System state and business logic JavaBeans.

26. What is a benefit of using JavaBeans to separate business logic from presentation markup within the JSP environment ?

Answer: It provides the developer with full access to the Java 2 Platform Enterprise Edition (J2EE), which is unavailable from outside the JavaBean environment.

27. What type of scriptlet code is better-suited to being factored forward into a servlet ?

Answer: Code that deals with logic and that common across requests.

OBJECTIVE TYPE QUESTIONS

1. Web server serves only static pages. (Y/N)
2. Apache is an application server. (Y/N)
3. Web-logic server is an application server. (Y/N)
4. JSPs are built on servlet semantics and all JSPs are compiled to servlets for runtime usage. (Y/N)
5. Actions cannot be used with scriplets. (Y/N)
6. When a JSP page is compiled, it becomes as a servlet. (Y/N)

ANSWER KEY

1. Y	**2.** N	**3.** Y	**4.** Y
5. Y	**6.** Y		

Previous Years' GATE Questions

1. Which of the following statements is FALSE? **(GATE 2004)**

A. HTTP runs over TCP
B. HTTP describes the structure of web pages
C. HTTP allows information to be stored in a URL
D. HTPP can be used to test the validity of a hypertext link.

2. Which of the following objects can be used in expressions and scriplets in JSP (Java Server Pages) without explicitly declaring them? **(GATE 2004)**

A. Session and request only
B. Request and response only
C. Response and session only
D. Session, request and response

3. Consider the following statements **(GATE 2004)**

(I) Telnet, ftp and http are application layer protocols.

(II) EJB (Enterprise Java Beans) components can be deployed in a J2EE (Java2 Enterprise Edition) application server.

(III) If two languages conform to the common language specification (CLS) of the Microsoft .NET framework, then a class defined in any one of them may be inherited in the other.

Which statements are true?

A. I and II only B. II and III only

C. I and III only D. I, II and III

4. An HTML form is to be described to enable purchase of office stationeries. Required items are to be selected (checked). Credit card details are to be entered and then the submit button is to be pressed. Which one of the following options would be appropriate for sending the data to the server? Assume that security is handled in a way that is transparent to the form design. **(GATE 2005)**

A. Only GET B. Only POST

C. Either GET or POST D. Neither GET nor POST

5. Given below is an excerpt of an xml specification.

<!DOCTYPE library SYSTEM "library.dtd">

<Book>

<title>GATE 2005</title>

<type value="BROCHURE"/>

<accno>10237623786</accno>

</Book>

<Book>

<type value="FICTION"/>

<accno>0024154807</accno>

Given below are several possible excerpts from "library.dtd". For which excerpt would the above specification be valid? **(GATE 2005)**

A. <!ELEMENT Book (title+, type, accno)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT type EMPTY>
<!ATTLIST type value(BROCHURE FICTION/ TECHNICAL)>
<!ELEMENT accno(#PCDATA)>

B. <!ELEMENT Book(title?, type, accno)>
<!ELEMENT title(#PCDATA)>
<!ELEMENT type ATTLIST>
<!ATTLIST type value(BROCHURE/FICTION/ TECHNICAL)>
<!ATTLIST accno(#PCDATA)>

C. <!ELEMENT Book(title*, type, accno)>
<!ELEMENT title(#PCDATA)>
<!ELEMENT type ATTLIST>
<!ATTLIST type value(BROCHURE/FICTION/ TECHNICAL)>
<!ELEMENT accno(#PCDATA)>

D. <!ELEMENT Book(title?, type, accno)>
<!ELEMENT title(#PCDATA)>
<!ELEMENT type EMPTY>
<!ATTLIST type value(BROCHURE/FICTION/ TECHNICAL)>
<!ELEMENT accno (#PCDATA)>

6. Consider the following XML DTD describing course information in a university.

<!ELEMENT Univ(Course+, Prof+)>

<!ELEMENT Course(Title, Eval*)>

<!ATTLIST Course Number ID #REQUIRED Instructor IDREF #IMPLIED)>

<!ELEMENT Title(#PCDATA)>

<!ELEMENT Eval(#PCDATA)>

<!ATTLIST Eval Score CDATA #REQUIRED>

<!ELEMENT Prof EMPTY>

<!ATTLIST Prof Name ID#REQUIRED Teaches IDREF#IMPLIED>

What is returned by the following XQuery? **(GATE 2006)**

let $as :=//@Score

for $c in /Univ/Course[Eval]

let $cs :=$c/Eval?@Score

where min($cs)>avg($as)

return $c

A. The professor with the lowest course evaluation

B. Professors who have all their course evaluations above the university average

C. The course with the lowest evaluation

D. Courses with all evaluations above the university average

7. Given below are some HTML lines.

<map name="map">

<area shape="poly" cords="50,50,50,100,100,100,75,75,100,50" href="f1.html">

<area shape="circle" cords="100,75,5" href="../cgi-bin/f2.pl?v1=ask abc's age">

<area shape="default" href="fd.html">

</map>

With reference to the HTML lines given above, consider the following statements.

(I) Clicking on the point <80,75> does not have any effect.

(II) The web browser can identify the area applicable to the mouse-click within the image and the subsequent action to be taken without additional responses from the web server.

(III) The dots in the cgi-bin URL will be resolved by the web browser before it is sent to the web server.

(IV) The "fd.html" request when sent to the web server will result in a GET request.

Exactly how many of the statements given above are correct? **(GATE 2007)**

A. 0 B. 1

C. 2 D. 3

8. Consider the XML document fragment given below.

<Book>

<title> GATE 2K7 Example</title>

<Content>

One of many lines

</Content>

<TOC>

One of many content entries

</TOC>

</Book>

Consider the XPath expression:

*{not (self)::TOC}

What would be the result of the given XPath expression when the current node is Book? **(GATE 2007)**

A. The Title and Content elements

B. The Content and TOC elements

C. The Title and TOC elements

D. The Title Content and TOC elements

9. Which of the following is TRUE only of XML but not HTML? **(GATE 2008)**

A. It is derived from SGML.

B. It describes content and layout.

C. It allows user defined tags.

D. It is restricted only to be used with the browsers.

ANSWER KEY

1. B	**2.** A	**3.** A	**4.** C
5. A	**6.** D	**7.** D	**8.** D
9. C			

CHAPTER NINE 9

Engineering Mathematics

9.1 The Foundations: Logic and Proofs

9.1.1 Propositional Logic

Propositions – a declarative sentence – either ***true*** or ***false***
e.g., (i) "Today is Monday."
(ii) "Two is less than four."
(iii) "What day is today?'
(iv) "n is less than four." Not propositions

Proposition variables – letters representing propositions, e.g., "p" "q" "r" …
Note that proposition itself contains no variables. Predicate has.

Truth value – e.g., If today is Monday, then (i) is **T** "true;" otherwise, (i) is **F** "false."
Truth value is a reflection of the domain of discourse.

Compound propositions – new propositions formed from existing propositions using the following ***logical operators:***

- **Negation** – "$\neg p$" "$\overline{p}$" "not p"
  (i) ("Today is not Monday.")
- **Conjunction** – "$p \wedge q$" "p and q"
  (ii) $\wedge$ "2 is greater than 1"
- **Disjunction** – "$p \vee q$" "p or q"
  (i) "Today is Tuesday" $\vee$
- **Exclusive or** – "$p \oplus q$"
  (i) $\oplus$ "It is raining today."

"$\vee$" – Inclusive or
What are the truth values of these compound propositions?

Conditional Statements –

"$p \rightarrow q$" "if p, then q" "***implication***" $\boldsymbol{p \vee q}$

p	q	$\boldsymbol{p \rightarrow q}$
T	T	T
T	F	F
F	T	T
F	F	T

p: hypothesis (antecedent, premise)
q : conclusion (consequence)

p	q	$\boldsymbol{\neg p \vee q}$
T	T	T
T	F	F
F	T	T
F	F	T

■ **Example** How to express "If today is Monday or Wednesday then we have class"?
p_1: Today is Monday;
p_2: Today is Wednesday;
q: We have class

■ **Answer:** $(p_1 \vee p_2) \rightarrow q$

- **Converse** of "$p \rightarrow q$" – "$q \rightarrow p$"
  Note: "$q \rightarrow p$" is NOT *equivalent* to "$p \rightarrow q$"
- **Inverse** of "$p \rightarrow q$" – "$\neg p \rightarrow \neg q$"
  Note: "$\neg p \rightarrow \neg q$" is NOT *equivalent* to "$p \rightarrow q$"
- **Contrapositive** of "$p \rightarrow q$" – "$\neg q \rightarrow \neg p$"
  Note: "$\neg q \rightarrow \neg p$" is *equivalent* to "$p \rightarrow q$"

Biconditional statement – "$p \leftrightarrow q$" "$(p \rightarrow q) \wedge (q \rightarrow p)$"

p	q	$\boldsymbol{p \leftrightarrow q}$
T	T	T
T	F	F
F	T	F
F	F	T

Precedence of Logical Operators

Operator	*Precedence*
¬	1
∧	2
⊕	?
∨	3
→	4
↔	5

Note that $p \wedge q \vee r$ means $(p \wedge q) \vee r$; but $p \vee q \wedge r$ means $p \vee (q \wedge r)$

9.1.2 Propositional Equivalences

- **Tautology** – a proposition its truth values is always true. e.g., $p \vee \neg p$
- **Contradiction** – a proposition its truth values is always false. e.g., $p \wedge \neg p$
- **Contingency** – neither tautology nor contradiction e.g., p, q

Logically Equivalent – "$p \equiv q$" "$p \Leftrightarrow q$" If $p \leftrightarrow q$ is a *tautology*, then $p \equiv q$

De Morgan's Laws

- $\neg(p \wedge q) \equiv \neg p \vee \neg q$
- $\neg(p \vee q) \equiv \neg p \wedge \neg q$

For example, Use truth tables to show that the above logical equivalent relations.

Extensions of **De Morgan's Laws**

- $\neg(p \wedge q \wedge r \wedge \ldots) \equiv \neg p \vee \neg q \vee \neg r \vee \ldots$
- $\neg(p \vee q \vee r \vee \ldots) \equiv \neg p \wedge \neg q \wedge \neg r \wedge \ldots$

A truth table with n propositional variables requires 2^n rows

Logical equivalences

$p \wedge \mathbf{T} \equiv p$ $p \vee \mathbf{F} \equiv p$	Identity laws
$p \vee \mathbf{T} \equiv \mathbf{T}$ $p \wedge \mathbf{F} \equiv \mathbf{F}$	Domination laws
$p \wedge p \equiv p$ $p \vee p \equiv p$	Idempotent laws
$\neg(\neg p) \equiv p$	Double negation law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative laws
$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	Associative laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's laws
$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Absorption laws
$p \vee \neg p \equiv \mathbf{T}$ $p \wedge \neg p \equiv \mathbf{F}$	Negation laws

$p \rightarrow q \equiv \neg p \vee q$
$p \rightarrow q \equiv \neg q \rightarrow \neg p$
$p \vee q \equiv \neg p \rightarrow q$
$p \wedge q \equiv \neg(p \rightarrow \neg q)$
$\neg(p \rightarrow q) \equiv p \wedge \neg q$
$(p \rightarrow q) \wedge (p \rightarrow r) \equiv p \rightarrow (q \wedge r)$
$(p \rightarrow r) \wedge (q \rightarrow r) \equiv (p \vee q) \rightarrow r$
$(p \rightarrow q) \vee (p \rightarrow r) \equiv p \rightarrow (q \vee r)$
$(p \rightarrow r) \vee (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$

$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$
$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

9.1.3 Predicates and Quantifiers

Predicates – Statements (propositions) with variables

$P(x)$, $Q(x, y)$

The truth value of a predicate depends on what value the variable takes.

e.g., "A *month* with less than 30 days"

P: Less-than-30-days. *x*: *month* *month* is a variable

$P(x)$: Less-than-30-days(*month*)

P(February) – **T**

The predicate is true if the *month* takes the value "February."

Note that the truth value of a proposition is determined by the domain of discourse, no variable.

e.g., "February has less than 30 days." – **T**. "March has less than 30 days." – **F**

e.g., How to express/find "Who is taller than James"

(1) Let *P*: taller-than-James *x*: Who $P(x)$

(2) Let *P*: taller-than *x*: Who $P(x$, James)

Answers to the question are the values of x that make the predicate *true*.

Quantifiers

- ***Universal quantifier***: ∀ – "for all" "for every"
- ***Existential quantifier***: ∃ – "there exist"

e.g., ∀ × Greater-than-3(x); "Every x that is greater than 3." ∀ × Greater-than(x, 3);

∃ × Greater-than-3(x); "Every x that is greater than 3." ∃ × Greater-than(x, 3);

Note that ∃ × Greater-than(x, 3) is *true*

∀ x Greater-than(x, 3) could be *true* or *false,* depending on *domain of discourse*

is *false* if the *domain of discourse* is all integers

is *true* if the *domain of discourse* is a set {4, 5, 6, …}

- **Counter example**

It is often hard to find if ∀ predicate is true; but easier to find it is false. Why?

It is often hard to find if a ∃ predicate is false; but easier to find it is true. Why?

- Uniqueness quantifier: ∃! – "there exist a unique"

e.g., ∃!x($\sqrt{x}$ = x) This predicate is true because only "1" satisfy $\sqrt{x}$ = x.

∃! × (x^2 = 4) This predicate is false because both "2" and "–2" satisfy x^2 = 4.

Quantifiers with Restricted Domains

e.g., ∀**x<0**(x^2 > 0) This predicate is true.

∀x<0(x^2 > x) – T? F?

∃!**x>0**(x^2 = 4) This predicate is true because only "2" satisfy x> 0 and x^2=4.

Note: ∀**x<0(**x^2 **> 0)** ≡ ∀**x(x<0** → x^2 **> 0)**

∃**x>0(**x^2 **= x)** ≡ ∃ **x (x > 0** ∧ x^2 **= x)**

Precedence of Quantifiers

Operator	*Precedence*
∀, ∃	
¬	1
∧	2
⊕	?
∨	3
→	4
↔	5

Note: "≡**" is not an operator**. It is a notation.

e.g., ∃x *P*(x) ∧ *Q*(x) ≡ (∃x*P*(x)) ∧ *Q*(x)

NOT≡ ∃ × (*P*(x) ∧ *Q*(x))

e.g., ∃x∀y (x + y > y)

Binding Variables

e.g., ∃x*P*(x) ∧ Q(x) the x in *P*(x) is ***bound***, the x in Q(x) is NOT bound - ***free***.

e.g., ∃x∀y (x + y > y) both x and y are bound.

Logical Equivalences Involving Quantifiers

If and only if they have the same truth table.

Negating Quantified Expressions

Negation	*Equivalent*	*Meaning*
¬∃x*P*(x)	∀x¬*P*(x)	There exist NO x that makes *P*(x) true
¬∀x*P*(x)	∃x¬*P*(x)	NOT all x makes *P*(x) true

DeMorgan's laws for quantifiers

¬∃x(P_1(x) ∧ P_2(x) ∧ … ∧ P_n(x)) ≡ ∀x(¬P_1(x) ∨ ¬P_2(x) ∨ … ∨ ¬P_n(x))

¬∃x(P_1(x) ∨ P_2(x) ∨ … ∨ P_n(x)) ≡ ∀x(¬P_1(x) ∧ ¬P_2(x) ∧ … ∧ ¬P_n(x))

¬∀x(P_1(x) ∧ P_2(x) ∧ … ∧ P_n(x)) ≡ ∃x(¬P_1(x) ∨ ¬P_2(x) ∨ … ∨ ¬P_n(x))

¬∀x(P_1(x) ∨ P_2(x) ∨ … ∨ P_n(x)) ≡ ∃x(¬P_1(x) ∧ ¬P_2(x) ∧ … ∧ ¬P_n(x))

9.1.4 Nested Quantifiers

Translating from Nested Quantifiers into English

e.g., ∀x ∃y(x + y = 2 ∧ 2x – y = 1)

For every x there exists a y such that x + y = 2 and 2x – y = 1

This predicate is *false* in real domain (for x and y being real numbers)

∃x ∃y(x + y = 2 ∧ 2x – y = 1)

There exists an x and there exists a y such that x + y = 2 and 2x – y = 1

This predicate is true. x = y = 1

Translating English Sentences into Logical Expressions

e.g., "Some students in this class have solved every exercise in this book."

Let: x – student, P(x) – student x is in this class.

y – exercise, Q(x, y) – exercise y in this book solved by a student x.

∃x ∀y(P(x) ∧ Q(x, y))

"No student in this class can pass this course without doing lots of exercises."

x – student, P(x) – student x passes this class

#e(x) – number of exercises done by x Note that #e(x) is NOT a predicate n – a large number

¬∃x (P(x) ∧ #e(x) < n) Note #e(x) < n is a predicate

≡ ∀x(¬P(x) ∨ ¬(#e(x) < n)) ≡ ∀x(P(x) → ¬(#e(x) < n))

≡ ∀x(P(x) → (#e(x) ≥ n))

9.1.5 Rules of Inference

Basis of inference

(***p*** → ***q***) ∧ ***p*** → ***q*** **modus ponens** (a rule of inference), or **law of detachment**

e.g., ***p*****:** student in CS major; ***q*****:** must take CSCI 2030 class

(1) $p \rightarrow q$ If a student is in CS major, then he/she must take CSCI 2030 class

(2) p _____"a student in CS major" is true

(3) $\therefore q$ therefore he/she must take CSCI 2030 class.

In practice, we have

$\forall x(P(x) \rightarrow Q(x))$ x – a variable representing any student in CS major

$P(J)$ J – a specific student in CS major

$Q(J)$ J must take CSCI 2030 class

Syllogism – "All men are mortal. Socrates is a man, therefore Socrates is mortal."

(1) $p \rightarrow q$ ***premise***

(2) p ***premise***

(3) $\therefore q$ ***conclusion***

If the premises are all true, then the conclusion becomes true. – a ***valid*** argument

Note: (1) For an ***argument form*** of {premises: (p_1, p_2, …, p_n), conclusion: q} to be valid, the compound predicate $p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q$ must be a tautology.

$(p \rightarrow q) \wedge p \rightarrow q$ is a tautology because

$(p \rightarrow q) \wedge p \rightarrow q \equiv (\neg p \vee q) \wedge p \rightarrow q \equiv (\neg p \wedge p) \vee (q \wedge p) \rightarrow q \equiv (\mathbf{F} \vee (q \wedge p)) \rightarrow q \equiv (q \wedge p) \rightarrow q \equiv \neg(q \wedge p) \vee q \equiv \neg q \vee \neg p \vee q \equiv \mathbf{T}$

Note that (2) If the premises are not all true, then the conclusion could be false, even the argument form is valid.

Rules of Inferences

Tautology	*Name*
$[(p \rightarrow q) \wedge p] \rightarrow q$	Modus ponens
$[(p \rightarrow q) \wedge \neg q] \rightarrow \neg p$	Modus tollens
$[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$	Hypothetical syllogism
$[(p \vee q) \wedge \neg p] \rightarrow q$	Disjunctive syllogism
$p \rightarrow (p \vee q)$	Addition
$(p \wedge q) \rightarrow p$	Simplification
$[(p) \wedge (q)] \rightarrow (p \wedge q)$	Conjunction
$[(p \vee q) \wedge (\neg p \vee r)] \rightarrow (q \vee r)$	Resolution

Rules of Inference for Quantified Statements

Name	*Tautology*	*Meaning*
Universal instantiation	$\forall x P(x)$ $\therefore P(C)$	P is true for any x in the domain, so P is true for a member C in the domain
Universal generalization	$P(C)$ for an arbitrary C $\therefore \forall x P(x)$	P is true for an arbitrary member C in the domain, so P is true for any x in the domain
Existential instantiation	$\exists x P(x)$ $\therefore P(C)$ for some C	There are x that make P true, so P is true for some member C in the domain
Existential generalization	$P(C)$ for some C $\therefore \exists x P(x)$	P is true for some member C in the domain, so there exist x that make P true

Name	*Tautology*	*Method*
Universal instantiation	$\forall x P(x)$ $\therefore P(C)$	Deduction
Universal generalization	$P(C)$ for an arbitrary C $\therefore \forall x P(x)$	Induction
Existential instantiation	$\exists x P(x)$ $\therefore P(C)$ for some C	Deduction
Existential generalization	$P(C)$ for some C $\therefore \exists x P(x)$	Induction

Combining Rules of Inference for Propositions and Quantified Statements

$\forall x(P(x) \rightarrow Q(x))$ x – a variable representing any student in CS major

$P(J)$ J – a specific student in CS major

$Q(J)$ J must take CSCI 2030 class

9.1.6 Introduction to Proofs

Proof – A ***valid argument*** that demonstrates a theorem is true (or false)

Some Terminology

- **Theorem** – a statement that can be shown to be true
- **Axiom** – a statement that is assumed to be true
- **Lemma** – a theorem (less important or easier to prove) used to prove other theorem
- **Corollary** – a consequent proposition of a theorem that has just been proved
- **Conjecture** – a statement proposed to be true, but has not been proved

Proof by Contraposition

Note: $p \rightarrow q \equiv \neg q \rightarrow \neg p$

Method:

First argument: Assume that $\neg q$ is true.

Middle arguments: Apply a sequence of inference rules.

Final argument: Show that $\neg p$ must be true.

e.g., "if m and n are integers and mn is even, then m is even or n is even"

Here, p is "mn is even" and q is "m is even or n is even"

First: Assume $\neg q$ true: $\neg q$ = ¬(m is even or n is even)
= (m is odd) ∧ (n is odd)

i.e, $m = 2k_m + 1$, $n = 2k_n + 1$

Middle:

(1) Let x be the product, x = mn

(2) Replace m and n by their definition: x = mn

$= (2k_m + 1)(2k_n + 1)$

$= 4k_mk_n + 2k_m + 2k_n + 1 = 2(2k_mk_n + k_m + k_n) + 1$

(3) According to definitions of even/odd integers, x = mn is odd

(let $k = 2k_mk_n + k_m + k_n$, x = mn = 2k + 1)

Final**:** Therefore mn is odd, i.e., $\neg \boldsymbol{p}$ must be true
Proof is done.

Proofs by Contradiction
Apply to $\boldsymbol{p} \rightarrow \boldsymbol{q}$

Method:

First argument:	Assume that $\boldsymbol{p}$ and $\neg \boldsymbol{q}$ are true.
Middle arguments:	Apply a sequence of inference rules.
Final argument:	Show that $\boldsymbol{q}$ must be true, which contradicts $\neg \boldsymbol{q}$**.**

e.g., The **pigeonhole** problem (or pigeonhole principle)
"If k is a positive integer and k + 1 or more objects are placed into k boxes, then there is at least one box containing two or more of the objects."
Here: $\boldsymbol{p}$ – "k + 1 or more objects are placed into k boxes"
$\boldsymbol{q}$ – "at least one box containing two or more of the objects"
First: Assume that $\boldsymbol{p}$ and $\neg \boldsymbol{q}$ are true.
Based on the above definition of $\boldsymbol{q}$ we have
$\neg \boldsymbol{q}$ – "NO box containing two or more of the objects"

Middle:
Based on *p*

(1) Place one object into each of the k boxes, that is, k objects are placed.

(2) We have k + 1 or more objects to be placed, that is, there is at least one box that should contain more than one object.

Final: Therefore $\boldsymbol{q}$ is true, which contradicts with our assumption that $\neg \boldsymbol{q}$ is true.
Proof is done.

If we try to prove that "Suppose none of the k boxes contains more than one object ($\neg \boldsymbol{q}$), then the total number of objects would be at most k ($\neg \boldsymbol{p}$)," then it is a **proof by contraposition**.

- **Exhaustive Proof**

Making arguments on all (finite number of) possible cases of the domain
e.g., "Prove that there are no positive perfect cubes less than 1000 that are the sum of the cubes of two positive integers."

(1) There are nine positive perfect cubes less than 1000
$1^3=1$, $2^3=8$, $3^3=27$, $4^3=64$, $5^3=125$, $6^3=216$, $7^3=343$, $8^3=512$, $9^3=729$

(2) Check each of them (actually, we start with $3^3=27$) with the pair of cubes at left
The maximum number of pair that needs to be checked is for $9^3=729$, which is C(2, 8) = 28. But we only need to check three pairs only (why?)

(3) No such cube exists among the 9, so the theorem is proved.

9.2 Basic Structures: Sets, Functions, Sequences and Sums

9.2.1 Sets

Definition: A set is an **unordered** collection of objects (called *elements* or *members*).
Often, objects placed in a set have some similar properties.
e.g., A = {a, b, c, …, z}; B = {0, 1};
C = {"T", "F"}; D = {0, 1, 2, …, 9}
X = {x | x is a positive integer};
Y = {y | y = }

Notations:

- "$\in$" *membership*: "a member of," "belongs to"

e.g., $a \in A$; $"T" \in C$; $(0 \in B) \wedge (0 \in D)$
Let $\mathbf{Z} = \{-\infty, \ldots, -1, 0, 1, 2, \ldots, \infty\}$; i.e., a set of all integers, then we can have
$\mathbf{Z+} = \{x \mid (x \in \mathbf{Z}) \wedge (x > 0)\}$ all positive integers
Do remember that "$\in$" is a logical operator, i.e., it leads to a result of either "true" or "false"

- "$\notin$" *not a membership*: "not a member of," "not belongs to"

e.g., $a \notin A$ is "false" and $0 \notin \mathbf{Z}^+$ is "true" according to the definitions above
Here, "$\notin$" is also a logical operator, i.e., it leads to a result of either "true" or "false"

- "$\subseteq$" *subset* $A \subseteq B$: $\forall x(x \in A \rightarrow x \in B)$

All members of set A are members of set B
e.g., $B \subseteq D$; $A \subseteq E$; $\mathbf{Z}^+ \subseteq \mathbf{Z}$; in above examples
Note: $\mathbf{A} \subseteq A$; $B \subseteq B$; **Any set is a subset of itself.**
Also, "$\subseteq$" is a logical operator, i.e., it leads to a result of either "true" or "false"
"=" *equal* $A = B$: $\forall x(x \in A \leftrightarrow x \in B)$

All members of A are Members of B, and *vice versa*
Note: $(A = B) \rightarrow [(A \subseteq B) \wedge (B \subseteq A)]$
Here, "=" is a logical operator, i.e., it leads to a result of either "true" or "false"
"$\subset$" *proper subset* $A \subset B$: $\forall x(x \in A \rightarrow x \in B) \wedge \exists x(x \in B \wedge x \notin A)$

All members of A are Members of B, but NOT *vice versa*
Here,"$\subset$" is also a logical operator, i.e., it leads to a result of either "true" or "false"

- "$\varnothing$" "{}" *empty set*

Note: $\{\varnothing\}$ is NOT an empty set It is a singleton set
singleton set: A set with one element. e.g., G = $\{x \mid x + 3 = 0\}$
For any set S: (i) $\varnothing \subseteq S$ and (ii) $S \subseteq S$.

Empty set is a subset of any set. **Any set is a subset of itself.**

- "$|S|$" *cardinality* number of distinct elements in a set

e.g., $|\varnothing| = 0$; $|A| = 26$; $|B| = 2$; $|E| = \mathbf{4}$. (38? 40?)
The set Z (and Z^+) is *infinite*. An infinite set could be countable or uncountable.

Venn Diagrams: To show the relationships between sets and their members
The Power Set: $P(S)$ - The set of all subsets of S
e.g., $P(\{0, 1\}) = \{\varnothing, \{0\}, \{1\}, \{0, 1\}\}$
Note: $P(\{0, 1\}) \neq \{\varnothing, 0, 1, \{0, 1\}\}$ (why?)
$\boldsymbol{|P(S)| = 2^{|S|}}$
$P(\varnothing) = \{\varnothing\}$; $P(\{\varnothing\}) = \{\varnothing, \{\varnothing\}\}$

9.2.1.1 Cartesian Product

Given two sets A and B,
$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$
Given sets A, B, and C,
$A \times B \times C = \{(a, b, c) \mid a \in A \wedge b \in B \wedge c \in C\}$
Given sets $A_1, A_2, \ldots, A_n$,
$A_1 \times A_2 \times \ldots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in A_i; i = 1, 2, \ldots, n\}$
e.g., A = {a, b, c, …, z}; B = {0, 1}; C = {"T", "F"}
A × B = {(a, 0), (b, 0), …, (z, 0), (a, 1), (b, 1), …..., (z, 1)}
A × B × C = {(a, 0, "T"), (a, 0, "F"), (b, 0, "T"), …, (z, 0, "F"), (a, 1. "T"), (b, 1, "T"), …, (z, 1, "F")}
Note: $A \times \varnothing = \varnothing \times A = \varnothing$; $B \times \{\varnothing\} = \{(0, \varnothing), (1, \varnothing)\}$

$|A \times B| = |A| \cdot |B|$ $\quad |A_1 \times A_2 \times \ldots \times A_n| = \prod_{i=1}^{n} |A_i|$

Relation
A relation R between the elements of set A and set B is a subset of A × B
$R \subseteq A \times B$
e.g., X = {Jane, Jim, June}
Y = (CS1620, CS 1840, CS 2030}
R_{xy} = {(Jane, CS1620), (Jane, CS 2030), (Jim, CS 1840), (June, CS 2030)}

Using Set Notation with Quantifiers
Set notation often helps to clearly define the domain of the quantifiers.
e.g., $\forall x(x^2 > 0)$ is ***false*** in mathematics generally, because if $x = \sqrt{-1}$
$\forall x \in \mathbf{R}(x^2 > 0)$ is ***true***. **R** – set of all real numbers

Truth Sets of Quantifiers
The truth set of a predicate *P* is the set of elements x in domain D for which $P(x)$ is true.
e.g., in above example, **R** is the truth set for the *P* of $x^2 > 0$.

9.2.1.2 Set Operations

- "$\cup$" ***union***: $A \cup B = \{x \mid x \in A \vee x \in B\}$

e.g., B = (0, 1); C = {"T", "F"}; $B \cup C = \{0, 1, \text{"T"}, \text{"F"}\}$
B = (0, 1); D = {0, 1, 2, …, 9}; $B \cup D = \{0, 1, 2, \ldots, 9\}$

- "$\cap$" intersection: $A \cap B = \{x \mid x \in A \wedge x \in B$

e.g., B = (0, 1); D = {0, 1, 2, …, 9}; $B \cap D = \{0, 1\}$
$|A \cup B| = |A| + |B| - |A \cap B|$

- $(A \cap B = \varnothing) \rightarrow$ Disjoint(A, B) Sets A and B are ***disjoint***.

e.g., B = (0, 1); C = {"T", "F"}; $B \cap C = \varnothing$

- "–" ***difference***: $A - B = \{x \mid x \in A \wedge x \notin B\}$

e.g., B = (0, 1); D = {0, 1, 2, …, 9}; $B - D = \varnothing$
$D - B = \{2, 3, \ldots, 9\}$
B = (0, 1); C = {"T", "F"}; $B - C = \{0, 1\} = B$

- "$\overline{\mathbf{A}}$" ***complement***: $\overline{\mathbf{A}} = U - A = \{x \mid x \notin A\}$

U – universal set $A \subseteq U$
e.g., Let U be the D = {0, 1, 2, …, 9}, and B = (0, 1);
$\overline{\mathbf{B}} = \{2, 3, \ldots, 9\}$
Let U be $\mathbf{Z}^+ \cup \{0\}$, then $\overline{\mathbf{B}} = \{x \mid x \in \mathbf{Z}^+ \wedge x > 1\}$

Note: $(\overline{\overline{\mathbf{A}}}) = A$; $A \cup \overline{\mathbf{A}} \equiv U$; $A \cap \overline{\mathbf{A}} \equiv \varnothing$

Set Identities

$A \cup \varnothing \equiv A$ $A \cap U \equiv A$	Identity laws
$A \cup U \equiv U$ $A \cap \varnothing \equiv \varnothing$	Domination laws
$A \cup A \equiv A$ $A \cap A \equiv A$	Idempotent laws
$(\overline{\overline{\mathbf{A}}}) \equiv A$	Complementation law
$A \cup B \equiv B \cup A$ $A \cap B \equiv B \cap A$	Commutative laws
$(A \cup B) \cup C \equiv A \cup (B \cup C)$ $(A \cap B) \cap C \equiv A \cap (B \cap C)$	Associative laws
$A \cap (B \cup C) \equiv (A \cap B) \cup (A \cap C)$ $A \cup (B \cap C) \equiv (A \cup B) \cap (A \cup C)$	Distributive laws

$\overline{A(B} \equiv \overline{A} \cap \overline{B}$ $\overline{A(B} \equiv \overline{A} \cup \overline{B}$	De Morgan's laws
$A \cup (A \cap B) \equiv A$ $A \cap (A \cup B) \equiv A$	Absorption laws
$A \cup \overline{A} \equiv U$ $A \cap \overline{A} \equiv \varnothing$	Complement laws

Generalised Unions and Intersections

- **Union**: $A_1 \cup A_2 \cup \ldots \cup A_n = \bigcup_{i=1}^{n} A_i = \{a_i \mid \exists i(a_i \in A_i);$ $i = 1, 2, \ldots, n\}$
- **Intersection**: $A_1 \cap A_2 \cap \ldots \cap A_n = \bigcap_{i=1}^{n} A_i = \{a_i \mid \forall i(a_i \in A_i); i = 1, 2, \ldots, n\}$

e.g., Let $A_i = \{\ldots, -2, -1, 0, 1, \ldots, i\}$. Find (p.131, Ex # 46)

$\bigcup_{i=1}^{n} A_i = \{\ldots, -2, -1, 0, 1, \ldots, n\}$

$\bigcap_{i=1}^{n} A_i = \{\ldots, -2, -1, 0, 1\}$

9.2.1.3 Functions

f: A → B A ***function*** f from set A to set B.
An assignment of ***exactly one*** element of B to each element of A
(Not necessary *exactly one* element of A)
Also called ***mapping*** or ***transformation***

- $f(a) = b$ the unique element b of B is assigned to element a of A.
  ≡ f maps a of A to b of B.

e.g., $f(4) = 2$. f is a square root function that maps 4 of set A to 2 of set B.
2 of set B is assigned by 4 of set A.
Expressed in variable: $f(x) = \sqrt{x}$.
Two functions are **equal** *if* their domain, range and mapping are all the same.

e.g., $f(x) = x^2; x \in \mathbf{Z}$ and $f(x) = x^2; x \in \mathbf{R}$ do not equal

$f(x) = x^2; x \in \mathbf{Z}$ and $f(x) = x^2; x \in \mathbf{Z}^+$ do not equal

A **relation** could be a function, or may not be a function.

e.g., A = {a, b, c}; B = {0, 1}

R = {(a, 0), (b, 0), (c, 1)} defines a function f: A → B.

R = {(a, 0), (a, 1), (b, 0), (c, 1)} is not a function.

Why?

Function Addition and Multiplication:
For f_1: A → **R** and f_2: A → **R**

$(f_1 + f_2)(x) = f_1(x) + f_2(x)$ function addition

$(f_1 f_2)(x) = f_1(x) f_2(x)$ function multiplication

Let ***S*** be a subset of A, i.e., ***S*** ⊆ A. $f(\boldsymbol{S})$ then is a subset of B, such that

$f(\boldsymbol{S}) = \{t \mid \exists s \in \boldsymbol{S}\ (t = f(s))\}$ $f(\boldsymbol{S}) \subseteq B$

One-to-One and Onto Functions

One-to-One (*injective*) f: A → B

$\forall a \forall b(f(a) = f(b) \rightarrow a = b)$ or $\forall a \forall b(a \neq b \rightarrow f(a) \neq f(b))$

Increasing / Decreasing f: **R** → **R**

Increasing: $\forall x \in \mathbf{R}\ \forall y \in \mathbf{R}(x < y \rightarrow f(x) \leq f(y))$
Strictly increasing: $\forall x \in \mathbf{R}\ \forall y \in \mathbf{R}(x < y \rightarrow f(x) < f(y))$
Decreasing: $\forall x \in \mathbf{R}\ \forall y \in \mathbf{R}(x > y \rightarrow f(x) \geq f(y))$
Strictly decreasing: $\forall x \in \mathbf{R}\ \forall y \in \mathbf{R}(x > y \rightarrow f(x) > f(y))$

Onto (*surjective*) f: A → B
$\forall b \in B\ \exists a \in A(f(a) = b)$
e.g., Multiprocessor CPU (B) to job assignment (A).

One-to-One correspondence (*bijection*) f: A → B
Both One-to-One and Onto
$\forall b \in B[\ \exists a \in A(f(a) = b) \wedge \forall a(f(a) = f(b) \rightarrow a = b)]$

Identity function l_A: A → A
One-to-One and Onto
$l_A(x) = x$ A self mapping

Inverse Functions and Compositions of Functions
Let f: A → B be a ***one-to-one onto*** function with $f(a) = b$
The ***inverse function*** of f is denoted by f^{-1}: B → A with $f^{-1}(b) = a$.

Note: Inverse function can only be defined on a ***one-to-one onto*** function.
A ***one-to-one onto*** function is **invertible**; otherwise, not invertible.
e.g., $f(x) = x^2$ and $g(x) = \sqrt{x}$ are ***invertible*** if $(x \in \mathbf{R} \wedge x > 0)$
$f^{-1}(x) = g(x)$ and $g^{-1}(x) = f(x)$

Note: if $(x \in \mathbf{R})$ $f(x) = x^2$ and $g(x) = \sqrt{x}$ are ***not invertible***

Compositions of Functions
Let f: A → B and g: B → C
The ***composition*** of f and g, denoted by $f \circ g$,
$f \circ g(a) = f(g(a))$
e.g., $f(x) = x^2$ and $g(x) = x + 1$
$f \circ g(x) = f(g(x)) = (x + 1)^2 = x^2 + 2x + 1$
$g \circ f(x) = g(f(x)) = (x^2) + 1 = x^2 + 1$ $f \circ g \neq g \circ f$ in general

Note: for $f(x) = x^2$ and $g(x) = \sqrt{x}$ with $(x \in \mathbf{R} \wedge x > 0)$
$f \circ g(x) = f(g(x)) = (\sqrt{x})^2 = x$ an identity function
$g \circ f(x) = g(f(x)) = \sqrt{x^2} = x$ $f \circ g = g \circ f$ because $f^{-1} = g$

i.e., $\boldsymbol{f \circ f^{-1}(x) = l_A}$ and $\boldsymbol{(f^{-1})^{-1} = f}$

Some Important Functions

floor function "$\lfloor x \rfloor$"

$f_{\lfloor x \rfloor}$: $\mathbf{R} \to \mathbf{Z}$; $\forall x \in \mathbf{R}\ \exists n \in \mathbf{Z}\ [(f_{\lfloor x \rfloor}(x) = n) \leftrightarrow (n \le x < n + 1)]$

ceiling function "$\lceil x \rceil$"

$f_{\lceil x \rceil}$: $\mathbf{R} \to \mathbf{Z}$; $\forall x \in \mathbf{R}\ \exists n \in \mathbf{Z}\ [(f_{\lceil x \rceil}(x) = n) \leftrightarrow (n - 1 < x \le n)]$

Note: $\forall x>0 \forall y>0[\lfloor x + y \rfloor \ge (\lfloor x \rfloor + \lfloor y \rfloor)]$ $\forall x>0 \forall y>0[\lceil x + y \rceil \le (\lceil x \rceil + \lceil y \rceil)]$

The $\lfloor x \rfloor$ and $\lceil x \rceil$ are useful for converting real numbers to integers

e.g., Calculation of memory location

factorial function "!"

$f_!$: $\mathbf{N} \to \mathbf{Z}+$; $f_!(n) = n! = n(n-1)! = \prod_{i=1}^{n} i$

$f_!(0) = 0! = 1$

Relationship of (1) relation, (2) function, (3) proposition, and (4) predicate is shown in Fig. 9.1.

(1) relation – ordered n-tuples in general, ordered pairs in particular.

(2) function – a mapping of objects from domain to range with or without variables.

(3) proposition- a function of truth value without variable.

(4) predicate – a function of truth value with variables.

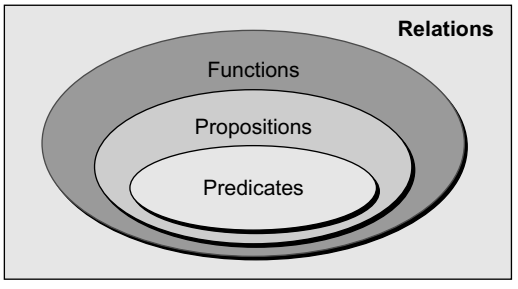


Figure 9.1

Cardinality

The sets A and B have the same ***cardinality*** if and only if there is a ***one-to-one*** correspondence from A to B.

The set **Z+** is often used as the A to measure the cardinality of B.

- ***Countable*** set – (1) is finite, or (2) has the same cardinality as $\mathbf{Z}^+$
- ***Uncountable*** set – not countable
- "$\aleph_0$" – "aleph null"

The cardinality of an infinite set S if it is countable, that is, $|S| = \aleph_0$

e.g., The set $\left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \cdots\right\}$ is countable, i.e.,

$$\left|\left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \cdots\right\}\right| = \aleph_0$$

So is for any $\{ax^n\}$ The set $\{(-1)^n\}$ is also countable

Any other countable infinite set?

- Set of prime numbers
- Fibonacci numbers

The set of all real numbers is not countable.

Any other uncountable set?

- An infinite set of random numbers
- The set of people's names

9.2.2 Relations

Definition Let A and B be sets. A *binary relation* from A to B is a subset of A × B.

e.g, A = {**T**, **F**} B = {0, 1}

A × B = {(**T**, 0), (**F**, 0), (**T**, 1), (**F**, 1)}

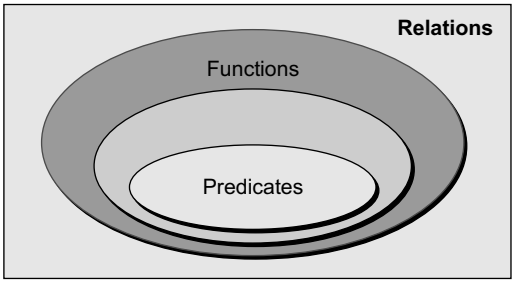
R = {(**F**, 0), (**T**, 1)}

Set R defines a relation from A to B

(**F**, 0) ∈ R, (**T**, 1) ∈ R; or (**F** R 0), (**T** R 1)

(**F**, 1) ∈/ R, (**T**, 0) ∈/ R; or (**F** R/ 0), (**T** R/ 1)

Functions as Relations



Major difference between relations and functions:

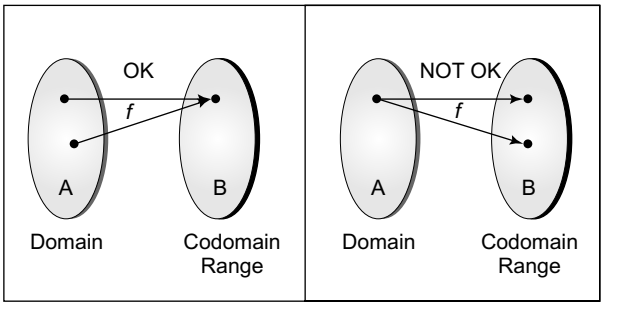


Figure 9.2 Functions as Relations

Both are OK with relations

Only left is OK with functions

Relations on a Set

A relation can be defined on a set itself $R \subseteq A \times A$

e.g., $A = \mathbf{Z}^+$; $R = \{(1, 1), (2, 4), (3, 8), (4, 16), \ldots, (n, n^2)\}$

Square relation

9.2.2.1 Properties of Relations

Reflective $\forall a((a, a) \in R)$

e.g., $A = \{a, b\}$ $R = \{(a, a) (a, b) (b, b)\}$ is reflective

The *equal*, *divide*, *greater-than-or-equal* relations on integers are reflective

Symmetric $\forall a \forall b((a, b) \in R \rightarrow (b, a) \in R)$

Antisymmetric $\forall a \forall b((a, b) \in R \wedge (b, a) \in R \rightarrow (a = b))$

Symmetric only on reflective ones – Not really symmetric

e.g., $A = \{a, b\}$

$R = \{(a, a) (a, b) (b, b)\}$ is antisymmetric, not symmetric

$R = \{(a, a) (a, b) (b, a)\}$ is symmetric, not antisymmetric

$R = \{(a, a) (b, b)\}$ is both antisemmetric and symmetric

$R = \{(a, b)\}$ is also antisemmetric, not symmetric

The *equal*, *divide*, *greater-than-or-equal* relations on integers are antisymmetric.

The *friend*, *classmate* relations on people are symmetric.

The *transpose* relation on matrices is symmetric.

Transitive $\forall a \forall b \forall c(((a, b) \in R \wedge (b, c) \in R) \rightarrow (a, c) \in R)$

e.g., $A = \{a, b, c\}$ $R = \{(a, a) (a, b) (b, b)\}$ is transitive

$R = \{(a, a) (a, b) (b, a)\}$ is not transitive

$R = \{(a, a) (a, b) (a, c) (b, b) (b, c)\}$ is transitive

$R = \{(a, a) (a, b), (b, a) (b, b), (b, c), (c, b)\}$ is not transitive

The *equal*, *divide*, *greater-than-or-equal* relations on integers are *transitive*

The *friend*, *classmate* relations on people may be *transitive*, maybe not.

Note: How may total relations can be defined on two sets A and B with $|A| = |B| = n$?

It equals to the total number of subsets on $A \times B$, which is 2^{n^2}

$|A \times B| = n^2$. $|\text{Subsets of } A \times B| = |1 \text{ element sets}| + |2 \text{ element sets}| + \ldots = 2^{|A \times B|}$

Combining Relations using "$\cup$", "$\cap$", "$-$"

e.g., $R_1 = \{(a, a) (a, b) (b, b)\}$

$R_2 = \{(a, a) (a, b) (a, c) (b, b) (b, c)\}$

$R_1 \cup R_2 = \{(a, a) (a, b) (a, c) (b, b) (b, c)\}$

$R_1 \cap R_2 = \{(a, a) (a, b) (b, b)\}$

$R_1 - R_2 = \varnothing$

$R_2 - R_1 = \{(a, c) (b, c)\}$

Composite " $\circ$"

$S \circ R = \{(a, c) \mid \forall a \forall b \forall c((a, b) \in R \wedge (b, c) \in S)$

e.g., $R = \{(a, a) (a, b) (b, b)\}$

$S = \{(a, a) (a, b) (a, c) (b, b) (b, c)\}$

$S \circ R = \{(a, a) (a, b) (a, c) (b, b) (b, c)\}$

Powers "R^n"

$R^1 = R$, $R^2 = R^1 \circ R$, $\ldots$ $R^{n+1} = R^n \circ R$

e.g., $R = \{(a, a) (a, b) (b, b)\}$

$R^2 = \{(a, a) (a, b) (b, b)\} = R^3 = \ldots R^n$

$S = \{(a, a) (a, b) (a, c) (b, b) (b, c)\}$

$S^2 = \{(a, a) (a, b) (a, c) (b, c)\}$

$S^3 = \{(a, a) (a, b) (a, c)\}$

$S^4 = \{(a, a) (a, b) (a, c)\} = \ldots S^n$

The relation R on a set A is transitive if and only if $R^n \subseteq R$ for $n = 1, 2, 3, \ldots$

The S and R above are transitive

9.2.2.2 *n*-ary Relations and Their Applications

Let $A_1, A_1, \ldots, A_n$ be sets. An *n-ary relation* is a subset of $A_1 \times A_2 \times \ldots \times A_1 \times A_n$

i.e, $R \subseteq A_1 \times A_2 \times \ldots \times A_1 \times A_n$

The set $A_1, A_1, \ldots, A_n$ is called the *domain* of the relation, and n is called its *degree*.

9.2.2.2.1 *Operations on n-ary Relations*

Selection *operator* S_C

Find all records (n-tuples) from R that satisfy the condition C.

Projection *operator* $P_{i_1 \cdot i_2 - i_m}$

A new table is formed by including the fields $i_1, i_2, \ldots, i_m$; but excluding the other n-m fields from a table of n fields.

Join *operator* $\boldsymbol{J}_P(R, S)$

Combine two tables into one.

Let $R \subseteq A_1 \times A_2 \times \ldots \times A_m$ and $S \subseteq B_1 \times B_2 \times \ldots \times B_n$; that is, $(a_1, a_2, \ldots, a_m) \in R$ and $(b_1, b_2, \ldots, b_n) \in S$.

A *join* $\boldsymbol{J}_P(R, S)$ consists of $(m + n - p)$ tuples $(a_1, a_2, \ldots, a_{m-p}, c_1, c_2, \ldots, c_p, b_1, b_2, \ldots, b_{n-p})$; where $p \le m$ and $p \le n$, and $(a_1, a_2, \ldots, a_{m-p}, c_1, c_2, \ldots, c_p) \in R$ and $(c_1, c_2, \ldots, c_p, b_1, b_2, \ldots, b_{n-p}) \in S$.

9.2.2.3 Representing Relations Using Matrices

Let $A = \{a_1, a_2, \ldots, a_m\}$, $B = \{b_1, b_2, \ldots, b_n\}$,

$R = \{(a_i, b_j) \mid (a_i, b_j) \in A \times B\}$

$$\mathbf{M}_R = \begin{array}{c|cccc} & b_1 & b_2 & \cdots & b_n \\ a_1 & 1 & & & \\ a_2 & & 1 & & \\ \vdots & & & \ddots & \\ a_m & & & & \end{array}$$

$$m_{ij} = \begin{cases} 1 & \text{if } (a_i, b_j) \in R \\ 0 & \text{if } (a_i, b_j) \notin R \end{cases}$$

e.g., $R1 = \{(a_i, b_j) \mid (a_i, b_j) \in A \times B \text{ and } i \geq j\}$

Zero-One Matrix

$$\mathbf{M}_{R1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & \ddots & 0 & 0 \\ 1 & 1 & 1 & \cdots & 0 \end{bmatrix}$$

$R2 = \{(a_i, b_j) \mid (a_i, b_j) \in A \times B \wedge i \neq j\}$

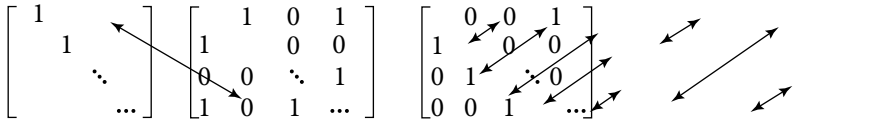
$A = B = \{a, b, c\}$

$$\mathbf{M}_{R2} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & \ddots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

$R = \{(a, a)\ (a, b), (b, a)\ (b, b), (b, c), (c, b)\}$

$$\mathbf{M}_R = \begin{array}{c|ccc} & a & b & c \\ a & 1 & 1 & 0 \\ b & 1 & 1 & 1 \\ c & 0 & 1 & 0 \end{array}$$

Matrices of *Reflexive* *Symmetric* *Antisymmetric* *Rs*



Boolean operations on Matrices of Relations

- "$\vee$" **Join:** $\mathbf{M}_{R1 \cup R2} = \mathbf{M}_{R1} \vee \mathbf{M}_{R2}$
- "$\wedge$" **Meet:** $\mathbf{M}_{R1 \cap R2} = \mathbf{M}_{R1} \wedge \mathbf{M}_{R2}$
- "$\odot$" **Boolean product:** $\mathbf{M}_{R1 \circ R2} = \mathbf{M}_{R1} \odot \mathbf{M}_{R2}$
- "$^{[r]}$" **Boolean power** $\mathbf{M}_R^{\mathbf{n}} = \mathbf{M}_R^{[n]} = \mathbf{M}_R \odot \mathbf{M}_R \odot \cdots \odot \mathbf{M}_R$

■ **Example**

$$\mathbf{M}_R = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{M}_S = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M}_{R \circ S} = \mathbf{M}_R \odot \mathbf{M}_S = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

How the numbers are obtained

$$m_{11} = = \begin{bmatrix} 1 & 0 & 1 \\ & & \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= (1 \wedge 0) \vee (0 \wedge 0) \vee (1 \wedge 1) = 1$$

$$m_{12} = = \begin{bmatrix} 1 & 0 & 1 \\ & & \end{bmatrix} \odot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= (1 \wedge 1) \vee (0 \wedge 0) \vee (1 \wedge 0) = 1$$

$$m_{21} = = \begin{bmatrix} 1 & 1 & 0 \\ & & \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= (1 \wedge 0) \vee (1 \wedge 0) \vee (0 \wedge 1) = 0$$

■ **Example**

$$\mathbf{M}_R = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$M_{R^2} = M_R^{[2]} = \mathbf{M}_R \odot \mathbf{M}_R$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \odot \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

9.2.2.4 Representing Relations Using Digraphs

Digraph **- Directed graph:-** **(V, E)**

V: a set of ***vertices*** (or *nodes*)

e.g., $V = \{a, b, c, \ldots\}$

E: a set of ***edges*** (or *arcs*)

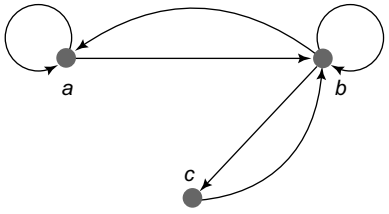
e.g., $E = \{(x, y) \mid x \in V \wedge y \in V\}$

For an edge (a, b)

a: *initial vertex*; b: *terminal vertex*

edge (a, a) called **loop**

e.g., A = B = {a, b, c}
R = {(a, a) (a, b), (b, a) (b, b), (b, c), (c, b)}



9.2.2.5 Closures of Relations

Closures:

- A ***closure*** of relation **R** with respect to a property **P** is a relation **S** such that
  (i) **S** is a subset of every relation with the property **P**,
  (ii) **S** contains **R** i.e., $\mathbf{R} \subseteq \mathbf{S}$
- ***Reflective closure***

The set S that contains all reflective relations R could have

e.g., A = {a, b, c}

R = {(a, a) (a, b), (b, a) (b, b), (b, c), (c, b)}

The ***Reflective closure*** of R is S = {(a, a) (a, b), (b, a) (b, b), (b, c), (c, b), (c, c)}

Which adds the reflective relation (c, c) to R

Note that the reflective relations (a, a) and (b, b) are already in R.

- ***Symmetric closure***

The set S that contains all symmetric relations that R does not have

e.g., A = {a, b, c}

R = {(a, a) (a, b), (b, a) (b, b), (b, c), (c, b)}

R is a ***symmetric closure*** of itself

Because for every relation in R, the symmetric pair is also contained

Such as (a, b) ⇔ (b, a), (b, c) ⇔ (c, b)

The *symmetric closure* S of relation R can be constructed by making $S = R \cup R^{-1}$

Where $R^{-1} = \{(b, a) \mid (a, b) \in R\}$

- ***Transitive closure***

A set **S** is a *transitive closure* of **R** if for every pair of relations (a, b) and (b, c) in **R**, the set S contains the relation (a, c) in addition to (a, b) and (b, c).

e.g., A = {a, b, c}

R = {(a, a) (a, b), (b, a) (b, b), (b, c), (c, b)}

The ***transitive closure*** of R is S = {(a, a) (a, b), (b, a) (b, b), (b, c), (c, b), (a, c)}

Which adds the reflective relation (a, c) to R

9.2.2.6 Equivalence Relations

A relation R on a set A is called an *equivalence relation* if it is reflexive, symmetric, and transitive.

Equivalent: "~" a ~b: Elements a and b are related by an equivalent relation.

e.g., Obviously, the "equal" relation on integer and real numbers is an equivalence relation.

Equivalence Classes: "**[a]**" Let $R \subseteq A \times A$ be an equivalence relation. The set of all elements that are related to an element a of A is called the *equivalence class* of a, denoted as $[a]_R$, or [a].

$[a]_R = \{s \mid (a, s) \in R\}$

If $b \in [a]_R$, then b is a representative of the equivalence class.

Equivalence Classes and Partitions

- Let $R \subseteq A \times A$ be an equivalence relation. These statements for elements a and b of A are equivalent:
  (i) aRb (ii) [a] = [b]
  (iii) $[a] \cap [b] \neq \varnothing$
- Let $R \subseteq S \times S$ be an equivalence relation. The equivalent classes of R form a partition of S.
  - Conversely, given a partition $\{A_i \mid i \in I\}$ of the set S, there is an equivalence relation R that has the set A_i, $i \in I$, as its equivalence classes.

e.g., "Bit strings that have the first three bits the same." is an equivalence relation

There are 8 equivalence classes: [000…], [001…], [010…], [100…], …

The set of bit strings with 3 or more bits in each of its elements is partitioned by these 8 equivalence classes into 8 partitions.

9.2.2.7 Partial Orderings

A relation R on a set S is called a *partial ordering* or *partial order* if it is reflexive, antisymmetric, and transitive.

Partially ordering: "$\preceq$"

$a \preceq b$: denotes two elements a and b of S are in partial ordering

$a \prec b$: denotes $a \preceq b$ but $a \neq b$

Poset: A set S together with a partial ordering R is called a *partially ordered set*, or *poset*, and is denoted by (S, R).

Members of S are called *elements* of the poset.

e.g., "≥", "≤" on integer set **Z**, "⊆" on a power set **P**, is a poset.

We write (**Z**, ≥), (**Z**, ≤), and (**P**, ⊆)

Comparable: The elements a and b of a poset ($S \preceq$) are called *comparable* if either $a \preceq b$ or $b \preceq a$.

When a and b are elements of S such that neither $a \preceq b$ nor

$b \preceq a$, a and b are called *incomparable*

: $\preceq$ b: a $\preceq$ b: b: a $\preceq$ b: "~" a ~b: Elements a and b are related by an equivalent relation.

***Totally ordered* set**: If $(S \preceq)$ is a poset and every elements of S are comparable, S is called a *totally ordered* or *linearly ordered* set, and $\preceq$ is called a *totally order* or *linear order*.

A totally ordered set is also called a *chain*.

***Well-ordered* set**: $(S \preceq)$ is a *well-ordered* set if it is a poset such that $\preceq$ is a totally ordering and every nonempty subset of S has a least element.

Principle of Well-ordered Induction: Suppose that S is a well-ordered set.

Then $P(x)$ is true for all $x \in S$, if

INDUCTIVE STEP: For every $y \in S$, if P(x) is true for all $x \in S$ with $x \prec y$, then P(y) is true.

Lexicographic Order: $(a_1, b_1) \prec (a_2, b_2)$ if $a_1 \prec_1 a_2$ or $(a_1 = a_2)$ and $(a_2 \prec_2 b_2)$

e.g., $(3, 5) \prec (4, 8)$ $(4, 5) \prec (4, 8)$ $(1, 3, 5) \prec (2, 3, 5)$

$(a, b, c) \prec (b, c, d)$ $(a) \prec (b)$ $(abc) \prec (abd)$

discreet $\prec$ discrete discrete $\prec$ discretion

Hasse Diagram:

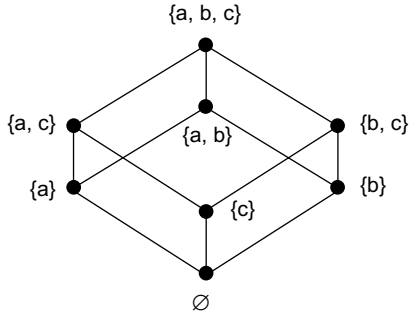


Figure 9.3 (a) Hasse Diagram of $\{P(\{a, b, c\}), \subseteq\}$

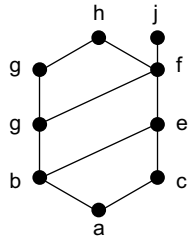


Figure 9.3 (b) Hasse Diagram of a poset

Maximal and Minimal Elements

- An element a is *maximal* in a poset $(S, \preceq)$ if there is no $b \in S$ such that $a \prec b$.
- An element a is *miniimal* in a poset $(S, \preceq)$ if there is no $b \in S$ such that $b \prec a$.

e.g., {a, b, c} is the maximal of all the posets in Hasse diagram (a)

g is the maximal of poset {a, b, d, g} of diagram (b)

h is the maximal of poset {a, b, d, g, h}, {a, c, e, f , h} of diagram (a)

j is the maximal of poset {a, c, e, f, j} of diagram (b)

- An element a is the *greatest element* of a poset $(S, \preceq)$ if $b \preceq a$ for all $b \in S$.
- An element a is the *least element* of a poset $(S, \preceq)$ if $a \preceq b$ for all $b \in S$.

e.g., {a, b, c} is the greatest element of all the posets in Hasse diagram (a) above.

$\varnothing$ is the least element of all the posets in Hasse diagram (a)

h is the greatest element of poset {a, b, d, g, h}, {a, c, e, f , h} of diagram (b)

a is the least element of poset {a, b, d, g, h}, {a, c, e, f, j} of diagram (b)

- An element *u* of S is an *upper bound* of subset A of poset $(S, \preceq)$ if $a \preceq u$ for all $a \in A$.
- An element *l* of S is a *lower bound* of subset A of poset $(S, \preceq)$ if $l \preceq a$ for all $a \in A$.

e.g., {a, b, c} is the upper bound of all the subsets in Hasse diagram (a)

$\varnothing$ is the lower bound of all the subsets in Hasse diagram (a)

h is the upper bound of subset {a, b, d, g, h} or {a, c, e, f , h} of diagram (a)

a is the lower bound of subset {a, b, d, g, h} or, {a, c, e, f, j} of diagram (a)

- An element *x* of S is a *least upper bound* of subset A of poset $(S, \preceq)$ if x is an upper bound that is less than every other upper bound of A.
- An element *y* of S is a *greatest lower bound* of subset A of poset $(S, \preceq)$ if *y* is a lower bound that is greater than every other lower bound of A.

e.g., g is the least upper bound of subset {b, d, g} of diagram (a)

Because between the two upper bounds g and h of {b, d, g}, g is less.

b is the greatest lower bound of subset {b, d, g} of diagram (b)

Because between the two lower bounds a and b of {b, d, g}, b is greater.

9.2.3 Lattices

A partially ordered set in which every pair of elements has both a least upper bound and a greatest lower bound is called a *lattice*.

e.g., The Hasse diagram (a) above is a *lattice*.

Actually, it is one of the most widely used lattice

The Hasse diagram (b) above is not a *lattice*

Because h and j both are upper bounds of elements e and f, so element pair e and f has no least upper bound.

9.2.4 Groups, Rings and Fields

A group G, sometimes denoted by {G, • }, is a set of elements with a binary operation, denoted by • , that associates to each ordered pair (a,b) of elements in G an element (a • b) in G, such that the following axioms are obeyed:

(A1) Closure: If a and belong to G, then a • b is also in G

(A2) Associative: a • (b • c)= (a • b) • c for all a,b,c in G

(A3) Identity element: There is an element e in G such that a • e= e • a=a for all a in G

(A4) Inverse element: For each a in G there is an element a' in G such that a • a' = a' • a = e

■ **Example** set S_N of permutations on the set {1, 2, .., N} with operation • - composition of permutations is a group with e = (1, 2, .., N). For N = 3, (123) • (321) = (321); (213) • (132) = (312)

If a group has a finite number of elements, it is referred to as a finite group, and the order of the group is equal to the number of elements in the group. Otherwise, the group is infinite group.

A group is said to be abelian if it satisfies the following additional condition:

(A5) Commutative: a • b = b • a for all a, b in G

The set of integers (positive, negative, and 0) under addition is an abelian group. The set of real numbers under multiplication is an abelian group.

The set S_N of permutations is not an abelian group.

When the group operation is addition, the identity element is 0; the inverse element of a is –a; and the subtraction is defined as: a – b = a + (–b).

Exponentiation within a group is defined as repeated application of the group operation, so that $a^3 = a \bullet a \bullet a$. We define also $a^3 = 0$, the identity element, and $a^{-n} = (a')^n$, where a' is inverse element for a. A group G is cyclic if every element of G is a power a^k (k – integer) of a fixed element $a \in G$. The element a is said to generate the group G, or to be a generator of G. A cyclic group is always abelian, and may be finite or infinite.

A **ring** R, sometimes denoted by {R,+, × }, is a set of elements with two binary operations, called addition and multiplication, such that for all a, b, c in R the following axioms are obeyed:

(A1-A5) R is an abelian group with respect to addition; that is, R satisfies axioms A1 through A5, For this case of an additive group we denote the identity element as 0 and the inverse of a as –a.

(M1) Closure under multiplication: If a and b belong to R, then ab is also in R (multiplication, as usually, is shown by concatenation of its operands)

(M2) Associativity of multiplication: a(bc) = (ab)c

(M3) Distributive laws: a(b + c) = ab + ac

(a + b)c = ac + bc

With respect to addition and multiplication, the set of all n-square matrices over the real numbers is a ring R.

The ring is said to be commutative if it satisfies the following additional condition:

(M4) Commutativity of multiplication: ab = ba

Let S be the set of all even integers under the usual operations of addition and multiplication. S is a commutative ring. The set of all n-square matrices over the real numbers is not a commutative ring.

We define integral domain, which is commutative ring that obeys the following axioms:

(M5) Multiplicative identity: There is an element 1 such that a1=1a=a for all a in R

(M6) No zero divisors: If a,b in R and ab=0, then, either a=0 or b=0.

Let S be the set of integers, positive, negative, and 0, under the usual operations of addition and multiplication. S is an integral domain.

A field F, sometimes denoted by {F, +, ×}, is a set of elements with two operations, called addition and multiplication, such that for all a, b, c in F the following axioms are obeyed:

(A1-M6) F is an integral domain; that is, F satisfies axioms A1 through A5 and M1 through M6.

(M7) Multiplicative inverse: For each a in F, except 0, there is an element a^{-1} in F, such that $a^{-1} = a^{-1}\, a = 1$

In essence, a field is a set in which we can do addition, subtraction, multiplication and division without leaving the set. Division is defined as: $a/b = a(b^{-1})$.

■ **Examples**

1. Do the natural numbers with addition form a group? No. Not all natural numbers have additive inverses. In fact, only 0 has an additive inverse. Thus (**N**, +) does not form a group.

2. The integers with addition form a group. As we have already seen, **Z** is closed under addition and all integers have additive inverses. Furthermore, 0 is the additive identity and addition is associative and commutative on **Z**. Thus, the integers with addition form an Albelian group.
3. The rationals and addition form an Albelian group.

Definition: A field is a set S with two operations, often called addition (+) and multiplication (x), with the following properties:

1. S is closed under both + and x
2. The operations + and x are both associative and commutative on S
3. There exists a + identity element, (we'll call it e_+) and ax identity element, (we'll call it e_x) such that $b + e_+ = e_+ + b = b$, and $b \times e_x = e_x \times b = b$ for all elements b in S.
4. For each element in S, there is an additive inverse. That is to say, for each b in S, there is a (–b) such that $b + (-b) = (-b) + b = e_+$
5. For each element in S, except e_+, there is a multiplicative inverse. That is to say, for each $b \neq e_+$ there is a b^{-1} such that $b \times b^{-1} = b^{-1} \times b = e_x$.
6. x is distributive over +

Note

The first 5 properties are equivalent to saying S and + form an Albelian group, and the set S without the additive identity element, with x form an Albelian group. In other words, a field is basically a set that forms a commutative (or Albelian) group with two different operations, and one of those operations distributes over the other.

■ **Examples**

1. Do the integers with addition and multiplication form a field? No. While the integers from an Albelian group under addition, the non-zero integers do not form an Albelian group under multiplication since there are not multiplicative inverses for all non-zero integers.
2. The rationals with addition and multiplication do form a field. We have already seen that **Q** with addition is an Albelian group. In addition, **Q** is closed under multiplication. Multiplication is associative and commutative on **Q**. 1 is the multiplicative identity. All non-zero rationals have a multiplicative inverse (namely, b/a is the inverse of a/b). Thus the non-zero rationals form an Albelian group with multiplication. And multiplication distributes over addition. Thus the rationals with + and x form a group.
3. Tables describes the action of • and * on the set {a, b, c, d, e}, does {a, b, c, d, e} with • and * form a field?

Table 9.1 (a) and (b)

•	a	b	c	d	e
a	a	b	c	d	e
b	b	c	d	e	a
c	c	d	e	a	b
d	d	e	a	b	c
e	e	a	b	c	d

*	a	b	c	d	e
a	a	a	a	a	a
b	a	b	c	d	e
c	a	c	e	b	d
d	a	d	b	e	c
e	a	e	d	c	b

Yes. All entries in both tables are in the set {a, b, c, d, e}, thus the set is closed under both operations. Furthermore the ijth entry always equals the ji^{tb} entry. So • and * are both commutative on the set {a, b, c, d, e}. Examining the tables you should be able to convince yourselves that the operations are also both associative. We not that a • a = a, a • b = b, a • c = c, a • d = d, a • e = e, and b* a = a, b* b = b, b * c = c, b * d = d, and b * e = e. Thus both operations have identity elements. In particular, a is the identity element for •, and b is the identity element form *. Furthermore, a • a = a, b • e = a, and c • d = a. Thus all elements have • inverses. (This leads us to think that • will be the additive operation for the set). b * b = b, c * d = b, and e* e = b. Thus, except for the • identity element a, all elements have * inverses. Examining the tables, we can see that * distributes over •. Thus {a, b, c, d, e} with • and * form a field.

9.3 Number Theory and Combinatorics

9.3.1 Integer Division

"a | b" a divides b i.e., b = ac, or c = b/a;
a, b, c: integers, $a \neq 0$
a: factor; b: multiple

"a |/ b" a does not divide b no integer c exists such that b = ac, or c = b/a.
$\neg\exists c(b = ac)$

■ **Example** How many positive integers not exceeding n are divisible by d?
i.e., how many k's exist, such that $0 < dk \leq n$

■ **Answer:** $\lfloor n/d \rfloor$

Division properties: a. b. c are integers
(i) if a | b and a | c, then a | (b + c);
(ii) if a | b, then a | bc;
(iii) if a | b and b | c, then a | c;
(iv) if a | b and a | c, then a | mb + nc; m and n are integers

9.3.1.1 Modular Arithmetic

Congruent: $\mathbf{a \equiv b \pmod m}$

a is *congruent* to *b* modulo *m* if (a–b)/m=0; a∈ **Z**, b∈ **Z**, and m∈ $\mathbf{Z}^+$

e.g., a = 131 and b = 29 are congruent to modulo 2; also modulo to 3, 6, 12, …

since a–b = 131–29 = 102 is dividable by 2, 3, 6, 12, 18, 24, …

Not Congruent: $\mathbf{a \not\equiv b \pmod m}$

Properties of modular arithmetic:

$[a \equiv b \ (\mathbf{mod}\ m)] \leftrightarrow [a \ \mathbf{mod}\ m = b \ \mathbf{mod}\ m]$

$[a \equiv b \ (\mathbf{mod}\ m)] \leftrightarrow [\exists k(a = b + km)]$

$[a \equiv b \ (\mathbf{mod}\ m) \wedge c \equiv d \ (\mathbf{mod}\ m)] \rightarrow [a+c \equiv b+d \ (\mathbf{mod}\ m)) \wedge ac \equiv bd \ (\mathbf{mod}\ m)]$

$(a + b) \ (\mathbf{mod}\ m) = (a \ (\mathbf{mod}\ m) + (b \ \mathbf{mod}\ m)) \ \mathbf{mod}\ m$

$ab \ \mathbf{mod}\ m = (\ (a \ \mathbf{mod}\ m)\ (b \ \mathbf{mod}\ m)) \ \mathbf{mod}\ m$

Applications of Congruence

- Hashing Functions
- Pseudorandom Numbers

Cryptology

Primes: $p \in \mathbf{Z}^+$ and *p* is dividable by 1 and *p* only.

Composite: $p \in \mathbf{Z}^+$ and p is not a *prime*.

9.3.1.2 The Fundamental Theorem of Arithmetic

Every positive integer greater than 1 can be written uniquely as a prime or as the product of two or more primes where the prime factors are written in order of non-decreasing size.

If n is a composite integer, then n has a prime divisor less than or equal to $\sqrt{n}$

Mersenne prime – of form $\mathbf{2^p–1}$ method to find large prime numbers

The Prime Number Theorem The ratio of the number of primes not exceeding x and x/lnx approaches 1 as x grows without bound.

Meaning: The number of primes less than x is approximately x/ln(x)

e.g.,x = 10, # of primes < 10 ~ 10/ln10 = 4.3 (actually 3

x = 100, # of primes < 100 ~ 100/ln100 = 21.7 (actually 25)

x = 1000, # of primes < 1000 ~ 1000/ln1000 = 144.76

Note that the approximation should be getting more accurate when x grows bigger.

This theorem gives the odds = 1/lnx that a randomly chosen number is prime.

Goldbach' Conjecture –"Every even integer greater than 2 is the sum of two primes."

The Twin Prime Conjecture –"There are infinitely many twin primes"

e.g., (3, 5), (5, 7), (11, 13), (17, 19), (29, 31), (41, 43), (59, 61), (71, 73)

Greatest Common Divisors and Least Common Multiples

If gcd(a, b) = 1, a and b are *relatively prime*

Why "relatively prime?' - Because, could be <u>both</u>, <u>one</u> of them, or <u>none</u>

e.g., <u>both</u>: (2, 3), (3, 5), …

<u>one</u>: (2, 9), (4, 11), …

<u>none</u>: (4, 9), (22, 39), …

If $\gcd(a_i, a_j) = 1$; $1 \le i \le j \le n$, $a_1, a_2, \ldots, a_n$ are *pairwise relatively prime*

e.g., (2, 3, 5 11), (3, 4, 10, 13)

Do remember that the term is "*pairwise*," nothing to do with "*twin prime*"

Using prime factorisations to find gcd(a, b) and lcm(a, b)

$$a = p_1^{a_2} p_2^{a_2} \ldots p_n^{a_n} \text{ and } b = p_1^{b_2} p_2^{b_2} \ldots p_n^{b_n}$$

$\forall 1 \le i \le n(a_i, b_i \in \mathbf{Z}^+ \cup \{0\})$

$$\gcd(a, b) = = p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \ldots p_n^{\min(a_n,b_n)}$$

$$\text{lcm}(a, b) = = p_1^{\max(a_1,b_1)} p_2^{\max(a_2,b_2)} \ldots p_n^{\max(a_n,b_n)}$$

9.3.1.3 Modular Exponentiation

Calculate b^n **mod** m

N is in binary expansion: $n = (a_{k-1}a_{k-2\ldots}a_1\,a_0)_2$

Thus, $b^n = b^{a_{k-1}a_{k-2}-a_1 2_0} = b^{a_{k-1}2^{k-1}+\ldots+a_1 2_0}$

$$= b^{a_{k-1}2^{k-1}} \ldots b^{a_1 2} b^{a_0}$$

Algorithm for Modular Exponentiation

procedure *modular exponentiation*(b: integers, $n = (a_{k-1}a_{k-2\ldots}a_1a_0)_2$, m: positive integer)

x := 1

power := b **mod** m

for i := 0 **to** k–1

begin

if $a_i = 1$ **then** x := (x·*power*) **mod** m

power := (*power*·*power*) **mod** m

end

{x equals b^n **mod** m }

9.3.1.4 Mathematical Induction

Principle

Two steps to prove $P(n)$ is true for all positive integers **n**,

1. **Basis Step**: verify that $P(1)$ is true
2. **Inductive Step**: show that $P(k) \rightarrow P(k+1)$ is true $\forall k \in Z$

$[P(1) \wedge \forall k(P(k) \rightarrow P(k+1))] \rightarrow \forall n(P(n))$

■ **Example** Prove $P(n)$: $n! < n^n$, $\forall n>1$

1. **Basis Step**: $P(2)$ is true $2! = 2 < 2^2 = 4$
2. **Inductive Step**:

Assume $P(k)$ is true, that is $k! < k^k$

$(k+1)! = (k+1)\, k!$ $(k+1)^{(k+1)} = (k+1)^k(k+1) > k^k(k+1)$

Since we assumed $k! < k^k$,

$(k+1)! = (k+1)\, k! < k^k(k+1) < (k+1)^k(k+1) = (k+1)^{(k+1)}$.

Hence Proved.

9.3.1.5 Recursive Algorithms

Modular Exponential: b^n **mod** m; $(b, n, m \in Z) \wedge (m \geq 2) \wedge (n \geq 0) \wedge (1 \leq b < m)$

Algorithm Recursive Modular Exponentiation

procedure *mpower*(b, n, m: integers with $m \geq 2$, $n \geq 0$)

if n = 0 **then** *mpower*(b, n, m) := 1

else if n is even **then** *mpower*(b, n, m) := *mpower*(b, n/2, m)^2 **mod** m

else

mpower(b, n, m) := *mpower*(b, $\lfloor n/2 \rfloor$, m)^2 **mod** m × b mod m) **mod** m

Algorithm A Recursive Algorithm for Computing gcd (a, b)

procedure *gcd*(a, b: nonnegative integers with a < b)

if n = 0 **then** *gcd*(a, b) := b

else *gcd*(n) := *gcd* (b **mod** a, a)

9.3.2 Counting

9.3.2.1 The Basics of Counting

Basic Counting Principles

- **Product rule** "sequent choices"

Total number of ways = number of ways at 1st place × number of ways at 2nd place

In general: Total number of ways

$$= \prod_{i=1}^{n} \textbf{number of ways at } i^{\textbf{th}} \textbf{ place}$$

e.g., How many ways to select two people out of a group of 9? $9 \cdot 8 = 72$

Auto license plates: three letters (26) followed by 3 digits (10):

$26 \cdot 26 \cdot 26 \cdot 10 \cdot 10 \cdot 10 = 17{,}576{,}000$

Similarly, number of passwords, addresses, variable names, …

- **Sum rule** "either choices"

Total number of ways = number of ways at 1st place + number of ways at 2nd place

In general: Total number of ways

$$= \sum_{i=1}^{n} \textbf{number of ways at } i^{\textbf{th}} \textbf{ place}$$

e.g., How many ways to select one people from a group of 9 teachers or 8 students?

9 + 8 = 17

■ **Example 13** What is the final value of k?

```
2k
k := 0
for i1 = 0 to n1
k := k + 1
for i2 = 0 to n2
k := k + 1
```

Number of elements in a union of two ***disjoint finite*** sets $|A_1 \cup A_2| = |A_1| + |A_2|$

More Complex Counting Problems

Password of (1) 6 to 8 characters long, (2) at least one digit

P: total number of password;

P_6, P_7, P_8: number of password with 6, 7, 8 characters, respectively.

$$P_6 = \sum_{i=1}^{6} \textbf{number of password with } i \textbf{ digits} = ?+?+?+?+?+?$$

= # with either letters of digits – # without digit

$= 36^6 - 26^6$

The same for P_7 and P_8

$P = P_6 + P_7 + P_8 = (36^6 - 26^6) + (36^7 - 26^7) + (36^8 - 26^8) = 2,684,483,063,360$

The Inclusion-Exclusion Principle (subtraction principle)

e.g., $$P_6 = \sum_{i=1}^{6} \textbf{number of password with } i \textbf{ digits}$$

$= ?+?+?+?+?+?$

$=$ **# with either letters of digits – # without digit**

$= 36^6 - 26^6$

Number of elements in a union of two ***non-disjoint finite*** sets

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|$$

e.g., $A_1 = \{a, b, c, d, e\}$, $A_2 = \{c, d, e, f, g\}$; $|A_1 \cup A_2| = 5 + 5 - 3 = 7$

$$|A_1 \cup A_2 \cup A_3| = |A_1| + |A_2| + |A_3| - |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 \cap A_3| + |A_1 \cap A_2 \cap A_3|$$

In general

$$|A_1 \cup A_2 \cup \ldots \cup A_n| = (|A_1| + |A_2| + \ldots + |A_n|)$$

$$- (|A_1 \cap A_2| + |A_1 \cap A_3| + \ldots + |A_{n-1} \cap A_n|)$$

$$+ (|A_1 \cap A_2 \cap A_3| + |A_1 \cap A_2 \cap A_4| + \ldots + |A_{n-2} \cap A_{n-1} \cap A_n|)$$

$$- \ldots$$

$$+ (-1)^{n-1}(|A_1 \cap A_2 \cap \ldots \cap A_n|)$$

9.3.3 The Pigeonhole Principle

If k is a positive integer and k + 1 or more objects are placed into k boxes, then there is at least one box containing two or more of the objects.

How many people in a group would have the same birthday?

- For a group of 367 peoples, the probability that there are two peoples having the same birthday is equal to 1.
- For a group of over 183 peoples, the probability that there are two peoples having the same birthday is greater than or equal to 0.5.

(In fact, the minimum number of people needed for two people among them having the same birthday is 23.

■ **Example**

e.g., n = 2, the multiple: 5n = 10
n = 3, the multiple: 37n = 111
n = 4, the multiple: 25n = 100

(These are just some instances. The textbook gives a formal proof)

A function *f* from a set with k + 1 or more elements to a set with k elements is not one-to-one.

The Generalised Pigeonhole Principle

If N objects are placed into k boxes, then there is at least one box containing at least $\lceil N/k \rceil$ objects.

e.g., Let (x_i, y_i), i = 1, 2, 3, 4, 5, be a set of five distinct points with integer coordinates in the xy plane. Show that the midpoint of the line joining at least one pair of these points has integer coordinates.

(1) For a line l_{ab} that joins points (x_a, y_a) and (x_b, y_b), a, b ∈ {1, 2, 3, 4, 5} to have integer coordinates at midpoint, it requires that the parity (even or odd) of the (x_a, y_a) and (x_b, y_b) must be the same. (i.e., both x_a and x_b are even (or odd), e.g., if $x_a = 12$ and $x_b = 28$, then $x_{mid} = 20$; if if $x_a = 11$ and $x_b = 27$, then $x_{mid} = 19$. The same for y_a and y_b)

(2) There are four possible parity cases (combinations) for the (x_i, y_i)'s: (i) [even, even], (ii) [Even, odd], (iii) [odd, even], and (iv) [odd, odd].

(3) We have five points, i = 1, 2, 3, 4, 5. According to the Pigeonhole principle, there must be at least two points with the same parity. So there must be at least one line joining a pair of these pints with integer coordinate midpoint.

Some Elegant Applications of the Pigeonhole Principle

- Every **sequence** of n2 + 1 distinct real numbers contains a subsequence of length n + 1 that is either **strictly increasing** or **strictly decreasing**
- **Ramsey Theory**

  "How many elements of some structure must there be to guarantee that a particular property will hold?"

- **Ramsey number R(m, n)**

  "The minimum number of guests $R(m, n)$ that must be invited so that at least m will know each other or at least n will not know each other."

Ramsey numbers

m	*n*	*R(m, n)*
3	3	6
3	4	9
3	5	14
3	6	18
3	7	23
3	8	28
3	9	36
3	10	[40, 43]
3	11	[46, 51]
3	12	[52, 59]

9.3.4 Permutations and Combinations

Permutations

- **r-permutation** An **ordered** arrangement of **r** elements of a set (n elements, r ≤ n)

  $P(n, r) = n\,(n-1)\,(n-2)\ldots(n-r+1)$

  Note: r multiplying terms

  $$P(n, r) = \frac{n!}{(n-r!)}$$

  Note: $P(n, 0) = 1$ $P(n, n) = n!$ 0! = 1; 1! = 1; 2! = 2; 3! = 6; …

■ **Example** How many ways are there for 10 women and six men to stand in a line so that no two men stand next to each other? [*Hint*: First position the women and then consider possible positions for the men.]

(1) First, line 10 women (Note: each has a different identity – so it is a permutation problem), we have $P(10, 10)$ ways. $P(10, 10) = 10! = 3{,}628{,}800$

(2) To make no two men standing next to each other, we should line them individually, that is each man should line between any two women, except the left-most and right-most position, where one woman is next.

There are 11 such positions after the 10 women are lined. That is the men can be lined in $P(11, 6) = (11 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1)/(5 \cdot 4\ 3 \cdot 2 \cdot 1) = 332{,}640$

(3) Since each way of woman's lining up leads to the $P(11, 6)$ ways of man's lining up, the total ways of lining the women and men = $P(10, 10)\ P(11, 6)$

= 1,207,084,032,000

Combinations

- **r-combination** An **unordered** selection of **r** elements of a set (n elements, $r \le n$)

$$\boldsymbol{C(n, r) = (n\ (n-1)\ (n-2)\ldots(n-r+1))/(r\ (r-1)\ \ldots\ 1)}$$

Note: r multiplying terms on both numerator and denominator

$$\boldsymbol{C(n, r)} = \frac{n!}{r!(n-r!)} = \frac{n\ (n-1)\ (n-2)\ \cdots\ (n-r+1)}{r!}$$

Note: $\boldsymbol{C(n, 0) = C(n, n) = 1}$

$\boldsymbol{C(n, r) = C(n, n-r)}$

■ **Example** How many bit strings of length 10 have (a) exactly three 0s? (b) more 0s than 1s? (c) at least seven 1s? (d) at least three 1s?

■ **Answer:**

(a) exactly three 0s?

$C(10, 3) = (10 \cdot 9 \cdot 8)/(3 \cdot 2 \cdot 1) = 120$

Note: this is not a permutation because the "0"s are indistinct. So is "1"s.

(b) more 0s than 1s? $C(10, 6) + C(10, 7) + C(10, 8) + C(10, 9) + C(10, 10) = (10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5)/(6 \cdot 5 \cdot 4\ 3 \cdot 2 \cdot 1) + (10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4)/(7 \cdot 6 \cdot 5 \cdot 4\ 3 \cdot 2 \cdot 1) + (10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3)/(8 \cdot 7 \cdot 6 \cdot 5 \cdot 4\ 3 \cdot 2 \cdot 1) + (10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2)/(9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4\ 3 \cdot 2 \cdot 1) + (10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1)/(10\ 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4\ 3 \cdot 2 \cdot 1)$

$= 210 + 120 + 45 + 10 + 1 = 387$

Could be easier to compute 'less 1s than 0s" $C(10,0) + C(10,1) + C(10, 2) + C(10, 3) + C(10,4)$

$= 1 + 10/1 + (10 \cdot 9)/(2 \cdot 1) + (10 \cdot 9 \cdot 8)/(3 \cdot 2 \cdot 1) + (10 \cdot 9 \cdot 8 \cdot 7)/(4\ 3 \cdot 2 \cdot 1)$

$= 1 + 10 + 45 + 120 + 210 = 386$

(c) at least seven 1s? $C(10, 7) + C(10, 8) + C(10, 9) + C(10, 10)$

From (b) above, we know that it would be easier if we compute "at most three 0s"

$C(10,0) + C(10,1) + C(10, 2) + C(10, 3) = 1 + 10/1 + (10 \cdot 9)/(2 \cdot 1) + (10 \cdot 9 \cdot 8)/(3 \cdot 2 \cdot 1)$

$= 1 + 10 + 45 + 120 = 176$

(d) at least three 1s?

It would need to compute $\sum_{i=3}^{10} C(10, i)$

Again from (b) and (c) above, we know that it would be easier if we compute "at most two 1s"

$C(10,0) + C(10,1) + C(10, 2) = 1 + 10/1 + (10 \cdot 9)/(2 \cdot 1) = 1 + 10 + 45 = 56$

The answer for "at least three 1s" = Total possibilities – "at most two 1s"

$= 2^{10} - 56 = 1024 - 56 = 968$

9.3.5 Recurrence Relations

$a_n = f(a_0, a_1, ..., a_{n-1})$ for a sequence $\{a_n\}$, $n \ge n_0 \in \mathbf{Z}^+$

Let a sequence $\{a_n\} = \{a_0, a_1, ..., a_{n-1}, a_n\}$. The sequence has a *recurrence relation* if the a_n can be expressed by an *equation* defined on $\{a_0, a_1, ..., a_{n-1}\}$.

Some most familiar examples are:

Sequence	Recurrence relation	Initial conditions	Non-recurrent express
Fibonacci numbers	$a_n = a_{n-1} + a_{n-2}$	$a_0 = 0, a_1 = 1$	??? (to be seen later)
Factorial	$a_n = n \cdot a_{n-1}$	$a_0 = 1$	$a_n = n!$
Exponential	$a_n = a \cdot a_{n-1}$	$a_0 = 1, a_1 = a$	$a_n = a^n$

■ **Example** Give the sequence represented by the recurrence: $a_n = na_{n-1} + (a_{n-2})^2$, $a_0 = -1$, $a_1 = 0$.

■ **Answer:** Sequence: –1, 0, 1, 3, 13, 74, 613, …

The Inclusion-Exclusion Principle (subtraction principle)

e.g., $P_6 = \sum_{i=1}^{6}$ **number of password with *i* digits**

$= ?+?+?+?+?+?$

= **# with either letters of digits – # without digit**

$= 36^6 - 26^6$

$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|$

e.g., Number of elements in a union of two ***non-disjoint finite*** sets

$A_1 = \{a, b, c, d, e\}$, $A_2 = \{c, d, e, f, g\}$;

$|A_1 \cup A_2| = 5 + 5 - 3 = 7$

$$|A_1 \cup A_2 \cup A_3| = |A_1| + |A_2| + |A_3| - |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 \cap A_3| + |A_1 \cap A_2 \cap A_3|$$

In general

$|A_1 \cup A_2 \cup \ldots \cup A_n| = (|A_1| + |A_2| + \ldots + |A_n|) - (|A_1 \cap A_2| + |A_1 \cap A_3| + \ldots + |A_{n-1} \cap A_n|)$

$+ (|A_1 \cap A_2 \cap A_3| + |A_1 \cap A_2 \cap A_4| + \ldots + |A_{n-2} \cap A_{n-1} \cap A_n|)$

$- \ldots$

$+ (-1)^{n-1}(|A_1 \cap A_2 \cap \ldots \cap A_n|)$

Formally, the Principle of Inclusion-Exclusion can be represented as:

$$|A_1 \cup A_2 \cup \ldots \cup A_n| = \sum_{1\le i\le n} |A_i| - \sum_{1\le i\le j\le n} |A_i \cap A_j| + \sum_{1\le i\le j\le k\le n} |A_i \cap A_j \cap A_k| - \ldots + (-1)^{n-1}(|A_1 \cap A_2 \cap \ldots \cap A_n|)$$

■ **Example** How many permutations of the 26 letters of the English alphabet do not contain any of the strings ***fish***, ***cat***, or ***dog***?

■ **Answer :**

Let A be the set of strings of all 26 letters

A_1 be the set of strings containing *fish*

A_2 be the set of strings containing *cat*

A_3 be the set of strings containing *dog*

We have

$|A| = 26!$

$|A_1| = 23!$ Consider placing the four letter *f*, *i*, *s*, *h*, together taking one space of the permutation, then count the permutations of it with the other 22 letters.

$|A_2| = 24!$

$|A_3| = 24!$

$|A_1 \cap A_2| = 21!$ Consider the two words *fish* and *cat* taking two spaces of the permutation, along with the other 19 letters.

$|A_1 \cap A_3| = 21!$

$|A_2 \cap A_3| = 22!$

$|A_1 \cap A_2 \cap A_3| = 19!$

The solution is

$|A| - |A_1 \cup A_2 \cup A_3| = |A| - (|A_1| + |A_2| + |A_3| - |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 \cap A_3| + |A_1 \cap A_2 \cap A_3|)$

$= 26! - (23! + 24! + 24! - 21! - 21! - 22! + 19!)$

$= 4.02025938 \times 10^{26}$

■ **Example** How many permutations of the 26 letters of the English alphabet do not contain the strings ***fish***, ***rat***, or ***bird***?

■ **Hint:** The sets $A_{fish} \cap A_{bird}$, $A_{rat} \cap A_{bird}$, and $A_{fish} \cap A_{rat} \cap A_{bird}$ are empty.

$|A| - |A_1 \cup A_2 \cup A_3| = |A| - (|A_1| + |A_2| + |A_3| - |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 \cap A_3| + |A_1 \cap A_2 \cap A_3|)$

$= 26! - (23! + 24! + 23! - 21!)$

$= 4.02024763 \times 10^{26}$

9.4 Graph Theory

Important Points

1. **In-degree** of a node is no. of edges coming into the node.
2. **Out-degree** of a node is no. of edges going out of the node.
3. **Degree** of a node is sum of in and out degrees.
4. **Self Loops** are the ones in which there exists an edge for a node from itself. If u is a node and (u,u) is element of edge set E. A **self-loop** is a **cycle** of **length 1**.
5. If **(u,v)** is an edge then it is said to be **incident from** or **leaves** vertex **u** and is **incident to** or **enters** vertex **v**.
6. A path is **simple** if it contains distinct nodes.
7. Two paths $(v_0,v_1,v_2,\ldots,v_{k-1},v_o)$, and $(v_0^1,v_p^1\ldots,v_{k-1}^1, v_0^1)$ form the same cycle if there exists an integer j such that $v_i^1 = v_{i+j \bmod k}$ for **k=0,1,…k-1**.
8. A **undirected** graph is **connected** if every pair of vertices are connected by a path.
9. **Connected components** of a graph are vertices under " is reachable from" relation.
10. A directed graph is **strongly connected** if every two vertices are reachable from others.
11. A directed graph is **strongly connected** then it will have only one strongly connected component.
12. Two graphs are isomorphic if there exists bijection f: $V \to V^1$ such that a edge in G also in G^1 only labels may change.
13. If two graphs are **isomorphic** then their degrees are same.
14. A **complete graph** is an undirected graph in which every pair of vertices is adjacent.
15. In a **complete graph** max path (possible) length is 1.
16. **Forest** is acyclic, undirected graph.
17. **Tree** is an acyclic, connected, undirected graph.
18. **Multigraph** is an undirected graph with multiple edges between vertices and with self-loops.
19. **Hypergraph** is undirected with hyperedges connecting subset of vertices.
20. In an undirected graph the length of a cycle must be at least 3.
21. In a directed graph if there is a cycle then it contains simple cycle.
22. In any connected undirected graph G(V,E) $|E| >= |V| - 1$ is valid.
23. In a complete graph with N nodes there will be $N(N-1)/2$ edges.
24. A graph is weekly connected if we suppress direction and resulting undirected graph is connected.

25. A graph is regular if every vertex has valence (order) that it is adjacent to same no. of other vertices.

26. The diameter of a graph is largest of all shortest path distances in the tree.

27. A path is said to be simple if all the vertices are distinct except the first and the last.

28. A cycle in a directed graph is a path of at least 1 and the path with length 1 is simple path.

29. The sum of the degrees of the vertices of a un-directed graph G is equal to twice the no. of edges in G.

30. In the case of undirected graphs the edges should be distinct. The path u,v,u in an undirected graph can not be a cycle.

31. Adjacency matrix of an undirected graph is symmetric about its main diagonal. In undirected graphs self loops are not allowed. Thus, the adjacency matrix contains diagonal elements as zeros.

32. Independent of what is the type of the graph, a graph with V vertices will requires $\Theta(V^2)$ memory locations to store in adjacency matrix.

33. A graph with E directed edges will be having the sum of the lengths of the adjacency lists as E, whereas undirected graph with the same no. of edges (E) will be having 2E.

34. In order to find the existence of an edge between two stations v, u requires $\Theta(1)$ time in adjacency matrix representation whereas in adjacency matrix representation it needs O(E).

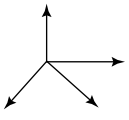
35. Adjacency matrix of a graph (with nodes N) contains only 1's in principal diagonal, then we can say that there are N connected components or N isolated points with self cycles.

36. Adjacency matrix of a graph (with nodes N) contains all zeros, then we can say that there are N connected components or N isolated points with no self cycles.

37. Adjacency matrix of a graph with N nodes contains all 1's in I'th row (except I'th element) and all zeros in I'th column then I'th station can be called as source.

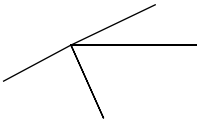
38. Adjacency matrix of a graph with N nodes contains all 0's in I'th row (except I'th element) and all ones in I'th column then I'th station can be called as sink.

39. Adjacency matrix of a graph with N nodes contains all 1's in I'th row (except I'th element) and all zeros all over then the graph can be said as star shaped with I'th station as center and directed edge for all other stations.



40. For the above type of graph, path matrix is same as adjacency matrix. Moreover, if A is adjacency matrix, then A^2, A^3, … A^N will be having all zeros.

41. Adjacency matrix of a graph with N nodes contains all 1's in I'th row (except I'th element) and all 1's in I'th column (except I'th element) and zeros all over then the graph can be said as star shaped with I'th station as center and un-directed edges for all other stations.



42. For the above type of graph, path matrix will be having all 1's. If A is adjacency matrix, I is the center node index the value of (I, I) element in A^2 will be N–1.

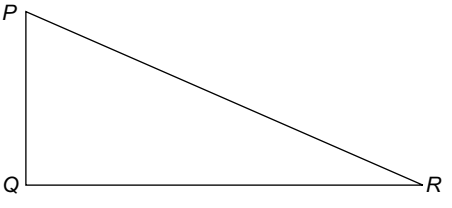
43. Regular graph will have its vertexes with same degree. For example, a graph is K-regular if and only if every vertex of it is of k-degree. A 0-regular graph is an isolated point (which can be also called as degenerate tree). A complete graph V-1 regular, where V is no. of vertexes.

44. The connected 2-regular graph with n vertices is the graph which contains a single n-cycle.

45. 3-regular graphs must have an even number of vertices.

46. A graph G is said to be complete of every vertex in G is connected to every other vertex. Thus, obviously a complete graph is connected. Its adjacency matrix will have 0's in diagonal and all other elements will be 1's. If A is such an adjacency matrix then A^2 will have all its off diagonal elements as 1's and where as diagonal elements as V-1, where V is no. of vertexes.

$$A = \begin{matrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{matrix} \qquad A^2 = \begin{matrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{matrix}$$



47. The following adjacency matrices pattern indicates that all the nodes in the respective graphs are in a cycle (directed), max cycle length is N.

01000	00001
00100	00010
00010	00100
00001	01000
01000	00001

48. The weight of minimal spanning tree is unique but the minimal spanning tree itself is not because of two or more edges having same weight.

49. A planar graph is the one which can be drawn in the plane so that its edges do not cross.

50. **Eulers's Formula:** V–E+R=2, where V, E, R are number of vertices, edges and regions.

51. In order to apply Eulers formula the underlying graph should be of connected type.

52. If G is a connected planar graph with p vertices and q edges, where p>=3, then q<=3p-6.

53. **Kuratowski Theorem:** A graph is nonplanar if and only if it contains a subgraph homeomorphic to $K_{3,3}$ or K_5, where $K_{3,3}$ is bipatrite graph with vertexes of degree 3 in both sets and K_5 is the complete graph with all vertexes of order 4 with 5 nodes.

54. Vertex colouring or colouring of graph is the process of assigning a color to each vertex such that no two adjacent vertexes are going to have same colour. The minimum number of colors needed to colour the graph is known as the chromatic number of that graph.

55. For a complete graph with n nodes requires n colours to colour it. Thus chromatic number is n.

56. A planar graph is 5-colourable (vertex colouring).

57. A planar graph is 4-colourable (Appel and Hasken). Here, regions are assumed to be coloured.

58. A bipartite graph is 2-colourable.

59. A graph G is having all cycles with even length is 2-colourable.

60. An undirected graph with a path from every node to another node is called as connected. A directed graph with this property is called as Directed Acyclic Graph (DAG) and is also called as strongly connected. If a di-graph is not strongly connected, but the underlying graph (without the directions to the arcs) is connected, then the graph is called as weakly connected.

61. Adjacency matrix representation needs $\Theta(|V^2|)$ memory locations and is preferred if the graph is dense. Otherwise, adjacency list representation is preferable. The space requirements will be $O(|V|+|E|)$, where V is no. of nodes, E is no. of edges.

62. Topological ordering is applicable to only directed acyclic graphs only. That is, graphs without cycles. Further, the ordering is not necessarily unique; any leagal ordering will do.

63. A simple algorithm to find a topological ordering is first find any vertex with no incoming edges and print the same and remove it along with its edges from the graph. Then we apply this same strategy to the rest of the graph.

```
Void toposort1(Graph G)
{
  Vetex V, W;
For(int counter=1; counter <=No_of_vertices; count++)
{          V=FIND_Vetex_with_zero_indegree();
If(V is not null)
{  Error("Graph has a cycle");
    return;
}
else
{
print V
for each W adjacent to V reduce indegree value by one
}
}
```

Because finding a node with zero in-degree is a simple scan of the an array having in-degrees, each call it takes O(V). Since, there are V such calls, the running time of the algorithm will be $O(V^2)$.

```
Toposort2(Graph G)
{
int counter=1; Vertex V, W;  Queue Q;
for each vertex V push into Q if their in-degree is zero.
While(Q is not empty)
{
 V=Q.dequeue();
Counter++;
For each adjacent vertex W of V decrease in-degree by one and if it becomes zero push into Q
}
if(counter<=no of vertexes) printf("Graph has a cycle);
}
```

64. Finding un-weighted shortest paths

a. Select some vertex, s, and set path length to 0.

b. Look for all adjacent vertices of s and having path length of 1.

c. Look for all vertices which are having path length of 2.

d. Repeat till all vertices are calculated

This algorithm behaves similar to breadth-first. The vertices closest to a node are evaluated first, and the most distant vertices are evaluated last. Once a vertex is processed, no other cheaper path will be found later.

```
void unweighted(Graph g){
Vertex V, W;
For(curr_dist=0;curr_dist<N; curr_dist++)
    {  for each vertex V
        if(V is not known and V's distance is
        curr_dist
            {  set V is known
               for each adjacent vertex W of V
                  if (W's distance is infinity)
                        {
                        set W's distance as
                        curr_dist+1;
                        W's path is via V;
                        }
            }
    }
```

PS:- A node is known or unknown can be decided by having one bit. Known means its distance is calculated and is not going to change.

The running time of this algorithm also $O(V^2)$. However, using Queue the same can be modified to have cost of O(E+V).

65. In the absence of **negative-cost cycle**, the shortest path from s to s is zero.

66. Normally, finding shortest path from a node s to other node necessitates finding the path from s to all vertices.

67. Dijkstra's algorithms running time is $O(E+V^2)=O(V^2)$. If the graph is dense, with $|E|=\Theta(V^2)$, this algorithm runs in linear time in the number of edges. In the case of sparse case $|E|=\Theta(V)$, this algorithm is too slow.

68. Some efficient implementations of Dijkstra's algorithms are available with O(E log V) for sparse graphs which employs priority queues.

69. Some efficient implementations of Dijkstra's algorithms are available with O(E + V log V) for sparse graphs which employs Fibnocci heaps.

70. A graph of V vertices can have V^{V-2} minimum spanning trees.

71. **Bipartite Graph** A bipartite graph, G(V,E), is a graph such that V can be partitioned into two subsets V1, V2, and no edge has both its vertices in the same subset.

72. An algorithm to find the minimum number of edges that need to be removed from an undirected graph so that the resulting graph is acyclic is a NP-complete problem.

73. A planar graph is a graph that can be drawn in a plane without any two edges intersecting.

74. A multigraph is a graph in which multiple edges are allowed between pairs of vertices.

75. When we traverse an undirected graph using depth first search we may get tree and back edges only. Whereas, if you traverse a directed graph forward and cross edges my be seen.

76. A necessary and sufficient condition for determining if an undirected graph is acyclic is that when running DFS() on it, one back edges are encountered.

77. PRIMS ALGORITHM (undirected weighted graph G=(V,E))

```
T=null (start with empty tree)
For each v ∈ V do
Key[v]= ∞
Key[r]=0   where r is a arbitrary vertex
    MakepriorityQueue(P,V)  // initialise P to contain
    the elements of V
While Empty(P) = false  do
   u=deleteMin(P)
   Min_wt = ∞
        For each v e adjacent[u] do
            If w(u,v) < key[v] then
                    DecreaseKey(P,v,w(u,v))
            If w(u,v) < Min_wt then
                    Min_wt = w(u,v)
                    Vmin=v
        T=T ∪ {(u,Vmin)} // add edge (u,vmin) to T
  Return T
```

Initialisation, i.e creation of priority heap takes Θ(V) if binary heap is used

DeleteMin operation takes O(log V); so he total time taken by DeleteMin is O(V log V).

DecreaseKey operation takes O(log V). At most one decreasekey operation is done per edge, thus it requires O(E log V).

Therefore, total running time of Prims algorithm is O(V log V + E log V), i.e O(E log V)

78. Algorithm to find the minimum number of edges that need to be removed from a directed graph so that the resulting graph is acyclic is an NP-complete problem.

79. Bi-connected components of a graph G is a partition of the edges into the sets such that the graph formed by each set of edges is bi-connected.

80. A directed graph has an Euler circuit if and only if it is strongly connected and every vertex has equal in-degree and out-degree.

81. A multigraph is said to be finite if it has a finite number of vertices and a finite number of edges. The finite graph with one vertex and no edges, i.e., a single point is called as trivial graph.

82. A vertex v in G is called as cutpoint or articulation point if G-v is disconnected. That is, if we remove vertex v and all edges of it the graph becomes disconnected. Similarly, an edge e of G is said to be a bridge if if G-e is disconnected.

83. A multigraph is said to be traversable if it can be drawn without any breaks in the curve and without repeating any edges. Thus, clearly traversable multigraph must be finite and fully connected.

84. A finite connected graph is Eulerian if and only if each vertex has even degree.

85. Any finite connected graph with two odd vertices is traversable. A traversable trail may begin at either odd vertex and will end at the other odd vertex.

86. Hamiltonian circuit traverses every vertex exactly once and my repeat the edges. Where as Eulerian circuit traverses every edge exactly once but may repeat vertices.

87. A quiver or "multidigraph" is a directed graph which may have more than one arrow from a given source to a given target. A quiver may also have directed loops in it.

88. A **simple graph** is an undirected graph that has no loops (edges connected at both ends to the same vertex) and no more than one edge between any two different vertices. In a simple graph the edges of the graph form a set (rather than a multiset) and each edge is a distinct pair of vertices. In a simple graph with n vertices every vertex has a degree that is less than n (the converse, however, is not true — there exist non-simple graphs with n vertices in which every vertex has a degree smaller than n).

89. A regular graph is a graph where each vertex has the same number of neighbours, i.e., every vertex has the same degree or valency. A regular graph with vertices of degree k is called a k-regular graph or regular graph of degree k.

90. Complete graphs have the feature that each pair of vertices has an edge connecting them.

91. A graph is called ***k*-vertex-connected** or ***k*-edge-connected** if no set of *k-1* vertices (respectively, edges) exists that, when removed, disconnects the graph. A *k*-vertex-connected graph is often called simply ***k*-connected**.

92. A directed graph is called **weakly connected** if replacing all of its directed edges with undirected edges produces a connected (undirected) graph. It is **strongly connected** or **strong** if it contains a directed path from *u* to *v* and a directed path from *v* to *u* for every pair of vertices *u*, *v*.

93. If every component of a graph is bipartite, then the graph is bipartite.

94. If u is a vertex of odd degree in a graph, then there exists a path from u to another vertex v of the graph where v also has odd degree.

95. If the distance d(u; v) between two vertices u and v that can be connected by a path in a graph is defined to be the length of the shortest path connecting them, then the distance function satisfies the triangle inequality: d(u, v) + d(v, w) >=d(u,w).

96. Any graph where the degree of every vertex is even has an Eulerian cycle.

97. In a directed graph where every vertex has the same number of incoming as outgoing paths there exists an Eulerian path for the graph.

98. Any tree with at least two vertices is bipartite.

99. An n-cube is a cube in n dimensions. A cube in one dimension is a line segment; in two dimensions, it's a square, in three, a normal cube, and in general, to go to the next dimension, a copy of the cube is made and all corresponding vertices are connected. If we consider the cube to be composed of the vertices and edges then every n-cube has a Hamiltonian circuit.

100. Every closed odd walk in a graph contains an odd cycle.

101. Every 2-connected graph contains at least one cycle.

102. If A is the adjacency matrix for the graph G, show that the (j, j) entry of A^2 is the degree of v_j .

103. If *u* and *v* are adjacent vertices in a graph, prove that their eccentricities differ by at most one.

104. The handshaking theorem states that the sum of the degrees of an undirected graph is twice the number of edges of the graph.

■ **Example** How do you make an undirected graph as a directed graph using (1) adjacency list (2) adjacency matrix?

■ **Answer :**

Adjacency Matrix: If A is adjacency matrix traverse A and if A(i,j) is 1 set A(j,i) as 1.

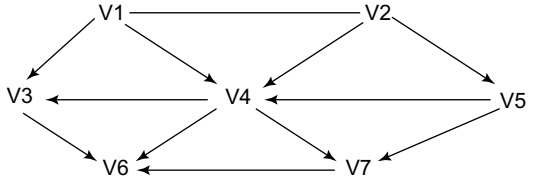
Adjacency List: Traverse adjacency list node by node if a node V_j is in adjacency list of V_i and then keep V_i in adjacency list of V_j.

■ **Example** In a party every member gave shake hand to every other. Find out in total how many shake hands took place?

Assuming each member as a node and shaking with other one as an edge, then no. of shake hands which took place given as

Sum of degree values of all nodes. That is 2 times of |E| where E is edge set.

■ **Example** Apply topological ordering and display the results for the following figure

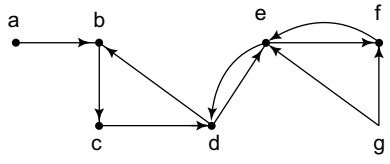


■ **Answer :** v1,v2,v5,v4,v3,v7,v6 and v1,v2,v5,v4,v7,v3,v6 are both topological orderings.

■ **Example** If T is a trail from vertex x to vertex y in a simple graph G (recall that a trail may repeat vertices but does not repeat edges), then prove that there exists a path P (with no vertices repeated) from x to y formed by a subset of vertices and edges of T.

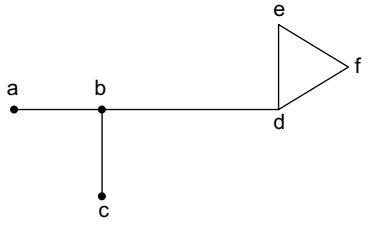
■ **Answer:** Start along trail and let v be the first vertex that is repeated. Remove edges on trail between first and second visit to v. Now go back to the start of the shortened trail and repeat the process again and again until no repeated vertices exist.

■ **Example** List strongly connected components of the graph.



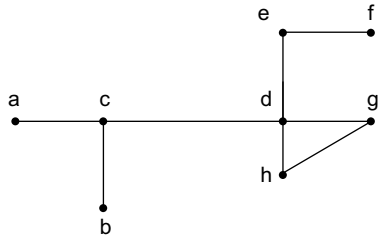
■ **Answer:** {a}, {g}, {b,c,d,e,f}

■ **Example** What is cut vertex?. Find out cut vertices in the following graph.



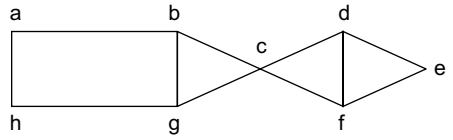
■ **Answer:**
A cut vertex is a vertex the removal of which would disconnect the remaining graph. The cut vertices are b,d.

■ **Example** What are the cut edges of the following graph?



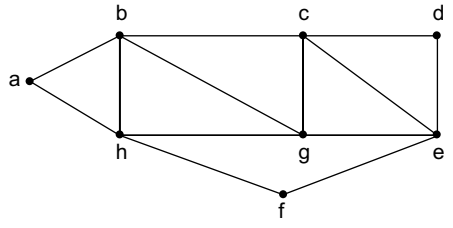
■ **Answer:** { (c,d), (e,d), (e,f) }

■ **Example** Does the following graph contains Euler circuit or Euler path?



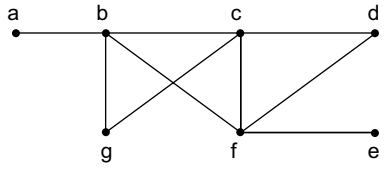
■ **Answer :** No. Four vertices are having degree 3.

■ **Example** Does the following graph have Euler circuit, Euler path?



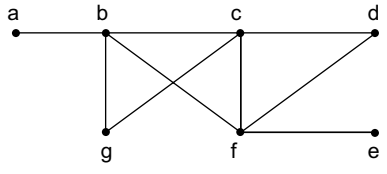
■ **Answer:** Yes. All vertices are of degree 2 or 4.

■ **Example** Does the following graph contain Hamilton circuit?



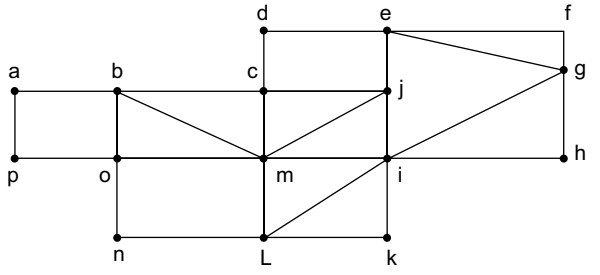
■ **Answer:** No. It contains 2 pendant vertices.

■ **Example** Does the following graph contain Hamilton path?



■ **Answer:** Yes. $e \to f \to d \to c \to g \to b \to a$

■ **Example** Does the following graph contains Hamilton circuit, Hamilton path?



■ **Answer:** Yes. Hamilton circuit is: a → b → m → j → c → d → e → f → g → h → i → k → l → n → o → p → a. Hamilton circuit implies Hamilton path, so Hamilton path also exists.

■ **Example** Does the existence of a Hamilton circuit imply that a Hamilton path exists?

■ **Answer :** YES

9.5 Matrices

A **matrix** is a rectangular organisation of set of elements . The elements in a horizontal line are called **rows**. The elements in a vertical line are called **columns**. The **dimension** of the matrix is determined by the number of rows and the number of columns.

A **square matrix** is one in which the number of rows equals the number of columns. If a matrix is comprised of a single column, i.e. its dimensions are m × 1, we call it a **column vector**. If a matrix is comprised of a single row, i.e. its dimensions are 1 × n, we call it a **row vector**.

Transpose Matrix is obtained by switching (exchanging) the rows and columns of the matrix. That is, a matrix which has the rows of matrix A as its columns, and the columns of matrix A as its rows, is called the **transpose** of A, and is denoted as A' (or A^T). It follows that if the dimension of A is m x n, then the dimension of A^T is n x m.

- **Properties of transpose:**
  - $(\mathbf{A^T})^T = \mathbf{A}$
  - $(\mathbf{kA})^T = \mathbf{k}(\mathbf{A^T})$
  - $(\mathbf{A + B})^T = \mathbf{A^T + B^T}$
  - $(\mathbf{A\ B})^T = \mathbf{B^T\ A^T}$ **and** $(\mathbf{A\ B\ C})^T = \mathbf{C^T\ B^T\ A^T}$

Symmetric Matrix: Any matrix A with the property that $A = A^T$ is a **symmetric** matrix. Hence, all identity matrices are also symmetric. Furthermore, notice that a necessary condition for a matrix to be symmetric is that the matrix must be square.

Identity (Unity) Matrix: A special symmetric matrix is the IDENTITY MATRIX, in which only diagonal elements are 1's and all remaining elements are zeros as shown below. It is often referred with I.

Properties of Identity matrix:

- For any matrix $\boldsymbol{A}$, $\boldsymbol{AI} = \boldsymbol{A}$, and $IA = A$. (Where we use **In** of suitable dimension to make the multiplication possible).
- Inverse of an Identity matrix is identity matrix itself. That is, $\mathbf{I^{-1}=I}$
- $\mathbf{I_n^2 = I}$.
- Determinant of any Identity matrix is 1.
- Inverse of a matrix: **B** is the inverse of **A** if **A B = B A = I**. If **B** is the inverse of **A**, then **A** is also the inverse of **B**. Both matrices are invertible.

  $\mathbf{A\ A^{-1} = A^{-1}\ A = I}$

  $\mathbf{(A^{-1})^{-1} = A}$

Diagonal Matrix: Diagonal Matrix is the one in which only diagonal elements are meaningful while others are zeros as shown below. The identity matrix is also a type of **Diagonal Matrix**.

Inverse of a diagonal matrix **A**, is the matrix whose diagonal elements are reciprocals of diagonal elements of **A.** Of course, inverse of a diagonal matrix is again diagonal.

Tridiagonal Matrix: A tridiagonal matrix is a square matrix in which all elements not on the following are zero – the major diagonal, the diagonal above the major diagonal, and the diagonal below the major diagonal.

■ **Example** Does non-square matrices will have diagonal entries?

■ **Answer :** Yes, for a $m \times n$ matrix $\boldsymbol{A}$, the diagonal entries are a_{11}, a_{22} ..., $a_{k-1, k-1}, a_{kk}$ where k=min $\{m,n\}$.

Diagonally Dominant Matrix **:** A $n \times n$ square matrix $\boldsymbol{A}$ is a diagonally dominant matrix if

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ i \neq j}}^{n} |a_{ij}| \quad \text{for all } i = 1, 2, \ldots, n \text{ and}$$

$$|a_{ii}| > \sum_{\substack{j=1 \\ i \neq j}}^{n} |a_{ij}| \quad \text{for at least one } i,$$

that is, for each row, the absolute value of the diagonal element is greater than or equal to the sum of the absolute values of the rest of the elements of that row, and that the inequality is strictly greater than for at least one row. **Diagonally dominant matrices are important in ensuring convergence in iterative schemes of solving simultaneous linear equations.** For example,

$$[A] = \begin{bmatrix} 15 & 6 & 7 \\ 2 & -4 & -2 \\ 3 & 2 & 6 \end{bmatrix}$$

is a diagonally dominant matrix as

$$|a_{11}| = |15| = 15 \geq |a_{12}| + |a_{13}| = |6| + |7| = 13$$

$$|a_{22}| = |-4| = 4 \geq |a_{21}| + |a_{23}| = |2| + |2| = 4$$

$$|a_{33}| = |6| = 6 \geq |a_{31}| + |a_{32}| = |3| + |2| = 5$$

and for at least one row, that is Rows 1 and 3 in this case, the inequality is a strictly greater than inequality. Also,

$$[A] = \begin{bmatrix} -15 & 6 & 9 \\ 2 & -4 & 2 \\ 3 & -2 & 5.001 \end{bmatrix}$$

is a diagonally dominant matrix as

$$|a_{11}| = |-15| = 15 \geq |a_{12}| + |a_{13}| = |6| + |9| = 15$$

$$|a_{22}| = |-4| = 4 \geq |a_{21}| + |a_{23}| = |2| + |2| = 4$$

$$|a_{33}| = |5.001| = 5.001 \geq |a_{31}| + |a_{32}| = |3| + |-2| = 5$$

The inequalities are satisfied for all rows and it is satisfied strictly greater than for at least one row (in this case it is Row 3).

$$[A] = \begin{bmatrix} 25 & 5 & 1 \\ 64 & 8 & 1 \\ 144 & 12 & 1 \end{bmatrix}$$

is not diagonally dominant as

$$|a_{22}| = |8| = 8 \leq |a_{21}| + |a_{23}| = |64| + |1| = 65$$

Null Matrix: A **null** matrix is a matrix in which every element of the matrix equals 0. It can be of any dimension. Addition and subtraction of a null matrix leaves the original matrix unchanged, multiplication with a null matrix results in another null matrix.

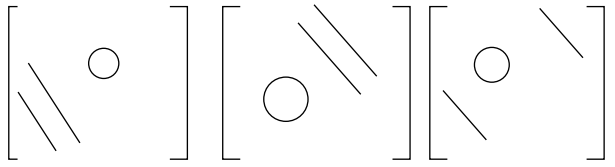
Sparse Matrix : It contains very less number of meaningful elements (not more than 1% of the total elements) while all others are zeros.

Lower, Upper Triangular and Band Matrices: These matrices contain meaningful elements in a particular part of the matrix. For example, in the following, first matrix is upper triangular, second is lower triangular while the third one is band matrix with meaningful elements along the principal diagonal. It is possible that the meaningful elements can be along the other diagonal.

$$\text{Lower Triangular} = \begin{bmatrix} d_1 & 0 & 0 \\ t_1 & d_2 & 0 \\ t_2 & t_3 & d_3 \end{bmatrix}$$

$$\text{Upper Triangular} = \begin{bmatrix} d_1 & t_1 & t_2 \\ 0 & d_2 & t_3 \\ 0 & 0 & d_3 \end{bmatrix}$$

$$|UT| = |LT| = d_1 d_2 d_3$$



Triangular Matrix: Inverses of triangular matrices are triangular. For example, for Lower Triangular matrices:

$$\begin{pmatrix} x & 0 \\ y & z \end{pmatrix}^{-1} = \begin{bmatrix} \frac{1}{x} & 0 \\ \frac{-y}{(x \cdot z)} & \frac{1}{z} \end{bmatrix}$$

$$\begin{pmatrix} x & 0 & 0 \\ w & y & 0 \\ v & u & z \end{pmatrix}^{-1} = \begin{bmatrix} \frac{1}{x} & 0 & 0 \\ \frac{-w}{(x \cdot y)} & \frac{1}{y} & 0 \\ \frac{w \cdot u - y \cdot v}{(x \cdot y \cdot z)} & \frac{-u}{(y \cdot z)} & \frac{1}{z} \end{bmatrix}$$

For Upper Triangular matrices:

$$\begin{pmatrix} x & y \\ 0 & z \end{pmatrix}^{-1} = \begin{bmatrix} \frac{1}{x} & \frac{-y}{(x \cdot z)} \\ 0 & \frac{1}{z} \end{bmatrix}$$

$$\begin{pmatrix} x & w & v \\ 0 & y & u \\ 0 & 0 & z \end{pmatrix}^{-1} = \begin{bmatrix} \frac{1}{x} & \frac{-w}{(x \cdot y)} & \frac{w \cdot u - y \cdot v}{(x \cdot y \cdot z)} \\ 0 & \frac{1}{y} & \frac{-u}{(y \cdot z)} \\ 0 & 0 & \frac{1}{z} \end{bmatrix}$$

Skew Symmetric Matrix: Skew symmetric matrix: $\underline{\mathbf{A}}^T = -\underline{\mathbf{A}}$ (Note: $a_{ij} = -a_{ji} \Rightarrow$ The diagonal elements = 0). An example:

$$\begin{bmatrix} 0 & 3 & -2 \\ -3 & 0 & -5 \\ 2 & 5 & 0 \end{bmatrix}^T = -\begin{bmatrix} 0 & -3 & 2 \\ 3 & 0 & 5 \\ -2 & -5 & 0 \end{bmatrix}$$

Idempotent Matrix: Any square matrix A is **idempotent** if $AA = A$. Hence, the identity matrix is an example of an idempotent matrix.

$$A = \begin{bmatrix} 0.4 & 0.8 \\ 0.3 & 0.6 \end{bmatrix}$$

Addition and Subtraction of Matrices: Addition (or subtraction) of matrices, $A + B$ (or $A - B$) requires that the matrices A and B be of equal dimension. Each element of matrix B is then added to (or subtracted from) the corresponding element of A.

Properties:

- Additions and subtractions are performed on each elements of the matrix.

- The order of the matrices must be the same (they are conformable).
- A + B = B +A (**Commutative**)
- A + B + C = (A + B) + C = A + (B + C) (**Associative**)
- A + 0 = 0 + A = A
- A – B = A + (–B)
- $(A + B)^T = A^T + B^T$
- $(A - B)^T = A^T - B^T$

Matrix Multiplication

Multiplication With Scalar: To multiply a matrix by a number (or 'scalar'), we multiply **every element** in the matrix by the number.

Multiplication with a Matrix: Matrix multiplication is simply an extension of vector multiplication. Let ***A*** be an m x n matrix, and ***B*** be a j × k matrix. Then the matrix product ***AB*** exists **only if** the dimensions of ***A*** and ***B*** allow for multiplication, i.e. only if j = n. If the dimensions do indeed allow for multiplication, then the matrix product, ***AB***, will be an m x k matrix. The product of matrices ***A*** and ***B***, denoted ***AB***, is itself a matrix. We obtain ***AB*** by using the rules of vector multiplication described below. Let ***AB = C***. Then c_{11} = the vector product of row 1 in ***A*** with column 1 in ***B***. Similarly, c_{ij} = the vector product of row i in *A* with column j in *B*.

$$\boldsymbol{A} = (\boldsymbol{m} \times \boldsymbol{p}) \quad \boldsymbol{B} = (\boldsymbol{p} \times \boldsymbol{n})$$
$$\boldsymbol{C} = \boldsymbol{AB} = (\boldsymbol{m} \times \boldsymbol{n})$$
$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$$

Matrix Multiplications properties

- $\boldsymbol{AB} \neq \boldsymbol{BA}$ (in general)
- **k*A* =*A*k** (for **scalar** multiplication)
- If ***(XY)Z*** exists, then ***(XY)Z = X(YZ)***
- If ***A(B + C)*** exists, then ***A(B + C) = AB + AC***
- $(A^T)^T = A$
- $(A + B)^T = A^T + B^T$
- $(kA)^T = k\,A^T$, where k is some scalar constant
- $(AB)^T = B^T A^T$
- For $A = B$ both matrices must be of the same size and $a_{ij} = b_{ij}$

Two forms of matrix multiplication are given as: (assuming A and B are two conformable matrices).

C = AB	***A*** premultiplies ***B***
C = BA	*A* postmultiplies ***B***

In general

$AB \neq BA$

except when

- Both matrices are diagonal
- Scalar multiplication
- When one matrix is an identity matrix

For multiplication, the DISTRIBUTIVE law holds

$$\boldsymbol{A(B + C)} = \boldsymbol{AB + AC}$$

As does the associative law

$$\boldsymbol{A(BC)} = \boldsymbol{(AB)C}$$

The CANCELLATION law does not hold

If $AB = 0$, then it does not follow that A or B or both are 0

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix}$$
$$B = \begin{bmatrix} 2 & -4 \\ -1 & 2 \end{bmatrix}$$

The Transpose of a Product

$$\boldsymbol{(AB)^T} = \boldsymbol{B^T A^T}$$

If $\boldsymbol{A} = \boldsymbol{A^T}$ and $\boldsymbol{B} = \boldsymbol{B^T}$ then $\boldsymbol{(AB)^T} = \boldsymbol{B^T A^T} = \boldsymbol{BA}$ does not necessarily = ***AB***

If ***A*** is symmetric but ***B*** is not note that $\boldsymbol{B^T AB}$ is symmetric since $\boldsymbol{(B^T AB)^T} = \boldsymbol{B^T A^T B} = \boldsymbol{B^T AB}$

If $\boldsymbol{A} = \boldsymbol{I}$ then $\boldsymbol{B^T B}$ is symmetric.

Let $A_{r \times c}$ be a $r \times c$ matrix. Then, both AA^t and A^tA are symmetric since

$$(AA^t)^t = (A^t)^t A^t = AA^t \text{ and } (A^tA)^t = A^t (A^t)^t = A^tA.$$

A^t is a $r \times r$ symmetric matrix while $A^t A$ is a $c \times c$ symmetric matrix.

For instance, let

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 3 & 0 & 1 \end{bmatrix} \text{ and } A^t = \begin{bmatrix} 1 & 3 \\ 2 & 0 \\ -1 & 1 \end{bmatrix}.$$

Then,

$$AA^t = [\, col_1(A)\ col_2(A)\ col_3(A)] \begin{bmatrix} row_1(A^t) \\ row_2(A^t) \\ row_3(A^t) \end{bmatrix}$$

$$[\, col_1(A)\ col_2(A)\ col_3(A)] \begin{bmatrix} col\,^t_1(A) \\ col\,^t_2(A) \\ col\,^t_3(A) \end{bmatrix}$$

$$col_1(A)col\,^t_1(A) + col_2(A)col\,^t_2(A) + col_3(A)col\,^t_3(A)$$

$$= \begin{bmatrix} 1 \\ 3 \end{bmatrix}\begin{bmatrix} 1 & 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \end{bmatrix}\begin{bmatrix} 2 & 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix}\begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 3 \\ 3 & 9 \end{bmatrix} + \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 2 \\ 2 & 10 \end{bmatrix}$$

In addition,

$$A^tA = row_1^t(A)row_1(A) + row_2^t(A)row_2(A)$$

$$= \begin{bmatrix}1\\2\\-1\end{bmatrix}[1 \quad 2 \quad -1] + \begin{bmatrix}3\\0\\1\end{bmatrix}[3 \quad 0 \quad 1]$$

$$= \begin{bmatrix}1 & 2 & -1\\2 & 4 & -2\\-1 & -2 & 1\end{bmatrix} + \begin{bmatrix}9 & 0 & 3\\0 & 0 & 0\\3 & 0 & 1\end{bmatrix} = \begin{bmatrix}10 & 2 & 2\\2 & 4 & -2\\2 & -2 & 2\end{bmatrix}$$

Note: If A and B are symmetric matrices, then AB is not necessarily equal to $(AB) = (BA)^T$. That is, AB might not be a symmetric matrix.

■ **Example**

$$A = \begin{bmatrix}1 & 2\\2 & 3\end{bmatrix} \text{ and } B = \begin{bmatrix}3 & 7\\7 & 6\end{bmatrix}.$$

Then,

$$AB = \begin{bmatrix}17 & 19\\27 & 32\end{bmatrix} \neq BA = \begin{bmatrix}17 & 27\\19 & 32\end{bmatrix}.$$

Properties of A^t and A^tA:

(a) $A^tA = 0 \Rightarrow A = 0$

$tr(A^tA) = 0 \Rightarrow A = 0$

(b) $PAA^t = QAA^t \Rightarrow PA = QA$

Properties of idempotent matrices:

1. $K^r = K$ for r being a positive integer.
2. $I - K$ is idempotent.
3. If K_1 and K_2 are idempotent matrices and $K_1K_2 = K_2K_1$. Then, K_1K_2 is idempotent.

■ **Example**

Let $A_{r\times c}$ be an $r \times c$ matrix. Then,

$K = A(A^tA)^{-1}A$ is an idempotent matrix since

$KK = A(A^tA)^{-1}A^tA(A^tA)^{-1}A = AI(A^tA)^{-1}A^t = A(A^tA)^{-1}A = K.$

Note

A matrix A satisfying $A^2 = 0$ is called nilpotent, and that for which $A^2 = I$ could be called unipotent.

■ **Example**

$$A = \begin{bmatrix}1 & 2 & 5\\2 & 4 & 10\\-1 & -2 & -5\end{bmatrix} \Rightarrow A^2 = 0 \Rightarrow \text{A is nilpotent.}$$

$$B = \begin{bmatrix}1 & 3\\0 & -1\end{bmatrix} \Rightarrow B^2 = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \Rightarrow \text{B is unipotent.}$$

Note

K is a idempotent matrix. Then, $K - I$ might not be idempotent.

Matrix Division: For simple numbers, division can be reduced to multiplication by the reciprocal of the divider: 32 divided by 4 is the same as 32 multiplied by 1/4 , or multiplied by 4^{-1}, where 4^{-1} is defined by the general equality a^{-1} a = 1. When working with matrices, we shall adopt the latter idea, and therefore not use the term division at all; instead we take the multiplication by an inverse matrix as the equivalent of division. That is, A % B = AB^{-1}

Trace of a Matrix : The trace of a matrix is the sum of the diagonal elements. That is, Trace of a matrix $tr(\boldsymbol{A}) = a_{11} + a_{22} + \ldots + a_{nn}$. It is defined for square matrices only.

- **Useful Points**
  - $tr(\boldsymbol{A}^T) = tr(\boldsymbol{A})$
  - $tr(\mathbf{k}\boldsymbol{A}) = k\,tr(\boldsymbol{A})$
  - $tr(\boldsymbol{I}_n) = n$
  - If $\boldsymbol{A}$ and B are square $tr(\boldsymbol{A} + \boldsymbol{B}) = tr(\boldsymbol{A}) + tr(\boldsymbol{B})$
  - $tr(\boldsymbol{AB}) = tr(\boldsymbol{BA})$
  - $tr(\boldsymbol{ABC}) = tr(\boldsymbol{BCA}) = tr(\boldsymbol{CAB})$

9.5.1 Determinant of a Matrix

The Determinant of a matrix is scalar and is referred as det A or $|A|$. This is also defined for square matrices. For example, determinant of 2×2 and 3×3 matrices are given as:

$$A = \begin{bmatrix}a_{11} & a_{12}\\a_{21} & a_{22}\end{bmatrix}$$

$$|A| = a_{11}a_{22} - a_{12}a_{21}$$

$$A = \begin{bmatrix}a_{11} & a_{12} & a_{13}\\a_{21} & a_{22} & a_{23}\\a_{31} & a_{32} & a_{33}\end{bmatrix}$$

$$|A| = +a_{11}\begin{vmatrix}a_{22} & a_{23}\\a_{32} & a_{33}\end{vmatrix} - a_{12}\begin{vmatrix}a_{21} & a_{23}\\a_{31} & a_{33}\end{vmatrix} + \begin{vmatrix}a_{21} & a_{22}\\a_{31} & a_{32}\end{vmatrix}$$

That is, $\boldsymbol{a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})}$

Where, $(a_{22}a_{33} - a_{23}a_{32})$ is called the minor of a_{11} and is usually denoted $|A_{ij}|$ – in this case $|A_{11}|$

$$A = \begin{bmatrix}3 & 5 & 4\\6 & 9 & 7\\2 & 8 & 1\end{bmatrix}$$

$$|A| = +3\begin{vmatrix}9 & 7\\8 & 1\end{vmatrix} - 5\begin{vmatrix}6 & 7\\2 & 1\end{vmatrix} + 4\begin{vmatrix}6 & 9\\2 & 8\end{vmatrix}$$

$$= 3(9 - 56) - 5(6 - 14) + 4(48 - 18)$$
$$= -141 + 40 + 120 = 19$$

The COFACTOR of the elements of a_{ij} denoted by c_{ij} is

$$c_{ij} = (-1)^{i+j}|A_{ij}|$$

Where A_{ij} is the sub-matrix of size n–1 by n–1 which is formed by removing ith row and jth column of matrix **A.** Determinant is calculated by selecting any row (any value for i) or any column (any value for j) using the following generalised equation.

$$|A| = \sum_{j=1}^{n} a_{ij}c_{ij} = \sum_{i}^{n} a_{ij}c_{ij}$$

Note:

- Minor: Any square sub-matrix $\underline{A}$ is called a minor of $\underline{A}$.
  - Principal minors:

$$\left|\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\right| \left|\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\right| \left|\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\right|$$

  - Leading principal minors:

$$a_{11}\left|\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\right| \left|\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\right|$$

Properties of Determinants

1. $|A^T| = |A|$
2. If we multiply all the elements of a column with k, then the determinant of the matrix gets multiplied by k. For example:

$$\begin{vmatrix} a_{11} & ka_{12} \\ a_{21} & ka_{22} \end{vmatrix} = ka = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$$

3. If A is (n x n) then $|kA| = k^n |A|$
4. If a square matrix has two equal rows or columns its determinant is zero
5. If any row (or column) is the multiple of any other row (or column) then its determinant is zero
6. The value of a determinant is unchanged if a multiple of one row (or column) is added to another row (or column)
7. If A is a diagonal matrix of order n then its determinant is $a_{11}a_{22} \ldots a_{nn}$
8. If A is a triangular matrix of order n then its determinant is $a_{11}a_{22} \ldots a_{nn}$
9. If B is the matrix obtained from a square matrix A by interchanging any two rows (or columns) then det B = -det A
10. If A and B are square matrices of the same order then $|AB| = |A|\,|B|$
11. If $A_1, A_2, \ldots, A_s$ are square matrices then $|\text{diag}(A_1, A_2, \ldots, A_s)| = |A_1|\,|A_2| \ldots |A_s|$
12. In general $|A + B|$ does not equal $|A| + |B|$
13. Determinant of Product of Matrices is product of individual matrices determinants. That is; $|ABC| = |A||B||C|$.

For covariance and correlation matrices, the determinant is a number that is sometimes used to express the "generalised variance" of the matrix. That is, covariance matrices with small determinants denote variables that are redundant or highly correlated (this is something that is used in factor analysis, or regression analysis). Matrices with large determinants denote variables that are independent of one another. The determinant has several very important properties for some multivariate stats (e.g., change in R^2 in multiple regression can be expressed as a ratio of determinants). It is obvious that the computation of the determinant is a tedious business, so only fools calculate the determinant of a large matrix by hand. We will try to avoid that, and have the computer do it for us.

9.5.2 Rank of a Matrix

The **Rank** of a matrix is equal to the highest order non-zero determinant that can be formed from its sub-matrices

$$A = \begin{bmatrix} 4 & 5 & 2 & 14 \\ 3 & 9 & 6 & 21 \\ 8 & 10 & 7 & 28 \\ 1 & 2 & 9 & 5 \end{bmatrix}$$

$\det A = 0$

$$\begin{vmatrix} 4 & 5 & 2 \\ 3 & 9 & 6 \\ 8 & 10 & 7 \end{vmatrix} = 63$$

Rank of $A = 3$

The rank of a matrix can also be measured by the maximum number of linearly independent columns of A. This also equals the maximum number of linearly independent rows

$$1\begin{bmatrix} 4 \\ 3 \\ 8 \\ 1 \end{bmatrix} + 1\begin{bmatrix} 5 \\ 9 \\ 10 \\ 2 \end{bmatrix} + 0\begin{bmatrix} 2 \\ 6 \\ 7 \\ 9 \end{bmatrix} + (-1)\begin{bmatrix} 14 \\ 21 \\ 28 \\ 5 \end{bmatrix} = 0$$

$c_1\underline{a}_1 + c_2\underline{a}_2 + c_3\underline{a}_3 + c_4\underline{a}_4 = 0$

- Definition: A set of vectors $\{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_m\}$ in $\Re^n$ is linearly independent iff
  $\alpha_1\underline{x}_1 + \alpha_2\underline{x}_2 + \ldots + \alpha_m\underline{x}_m = 0 \Leftrightarrow \alpha_1 = \alpha_2 = \ldots = \alpha_m = 0$
- **Examples**:
  - $\underline{x}_1 = [1\ 0]^T$ and $\underline{x}_2 = [1\ 2]^T$
    $\alpha_1(1) + \alpha_2(1) = 0$ and $\alpha_1(0) + \alpha_2(2) = 0$
    $\Rightarrow \alpha_1 = \alpha_2 = 0$

$\Rightarrow$ they are linearly independent

- $\underline{x}_1 = [1\ 1]^T$ and $\underline{x}_2 = [2\ 2]^T$

  $\alpha_1(1) + \alpha_2(2) = 0$ and $\alpha_1(1) + \alpha_2(2) = 0$

$\Rightarrow \alpha_1 = -2\alpha_2$

$\Rightarrow$ they are linearly dependent

A full column rank matrix has the same number of linearly independent columns (rows) equal to the number of columns

A full row rank matrix has the same number of linearly independent rows (columns) equal to the number of rows

If $\boldsymbol{A}$ does not have full row and column rank it is singular

If $\boldsymbol{A}$ does have full row and column rank it is non-singular

rank $(I_n) = n$

rank $(\mathbf{k}\boldsymbol{A})$ = rank $(\boldsymbol{A})$

rank $(\boldsymbol{A}\mathbf{T})$ = rank $(\boldsymbol{A})$

If $\boldsymbol{A}$ is (m x n) then rank $(\boldsymbol{A})$ is $\leq$ min {m, n}

rank $\boldsymbol{AB} \leq$ **min{rank** $(\boldsymbol{A})$**, rank** $(\boldsymbol{B})$**}**

9.5.3 Inverse of a Matrix

In scalar algebra, the inverse of a number is that number which, when multiplied by the original number, gives a product of 1. Hence, the inverse of x is simple 1/x. or, in slightly different notation, x^{-1}. In matrix algebra, the inverse of a matrix is that matrix which, when multiplied by the original matrix, gives an identity matrix. The inverse of a matrix is denoted by the superscript "–1". Hence,

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

A matrix must be square to have an inverse, but not all square matrices have an inverse. In some cases, when the **determinant** of the matrix is zero, the inverse does not exist. For covariance and correlation matrices, an inverse will always exist, provided that there are more subjects than there are variables and that every variable has a variance greater than 0.

If $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices of order n such that $\boldsymbol{AB} = \boldsymbol{BA} = \boldsymbol{I}_n$ then $\boldsymbol{B}$ is called the inverse of $\boldsymbol{A}$.

$\boldsymbol{A}$ has an inverse iff it is of full column and row rank – non-singular.

$\boldsymbol{A^{-1}} = \boldsymbol{C}^T / |\boldsymbol{A}|$

C^T is the transpose of the matrix of co-factors which is also called adjoint matrix.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$|A| = 4 - 6 = -2$$

$$c_{11} = 4\ c_{12} = -3\ c_{21} = -2\ c_{22} = 1$$

$$A^{-1} = \frac{-1}{2}\begin{bmatrix} 4 & -3 \\ -2 & 1 \end{bmatrix}^T$$

$$A^{-1} = \begin{bmatrix} 2 & 1 \\ 3/2 & -1/2 \end{bmatrix}$$

Properties of Inverses

1. $\boldsymbol{I^{-1}} = \boldsymbol{I}$
2. $(\boldsymbol{A^{-1}})^{-1} = \boldsymbol{A}$
3. $\boldsymbol{AB} = \boldsymbol{I} \qquad \boldsymbol{BA} = \boldsymbol{I}$
4. *If* $\boldsymbol{A}$ is non-singular then $\boldsymbol{A^{-1}}$ non-singular
5. *If* $\boldsymbol{A}$ and $\boldsymbol{B}$ non-singular $\boldsymbol{(AB)^{-1} = B^{-1}A^{-1}}$

The inverse of a product of two matrices is the swapped product of the individual inverse matrices. Thus: $(\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$. *Where it is assumed that* $\boldsymbol{A}$ *and* $\boldsymbol{B}$ *are square and that* $\boldsymbol{A}^{-1}$ *and* $\boldsymbol{B}^{-1}$ *exist. The proof is straightforward. Let* $\boldsymbol{AB}$ *be given, then we have*

$$(\boldsymbol{AB})\ (\boldsymbol{AB})^{-1} = (\boldsymbol{AB})\ (\boldsymbol{B}^{-1}\boldsymbol{A}^{-1}) = \boldsymbol{A}\ (\boldsymbol{B}\ \boldsymbol{B}^{-1})\ \boldsymbol{A}^{-1} = \boldsymbol{A}\ \boldsymbol{I}\ \boldsymbol{A}^{-1}$$

$$= \boldsymbol{A}\ \boldsymbol{A}^{-1} = \boldsymbol{I}.$$

Left Inverse of a (m x n) matrix $\boldsymbol{A}$ is the (n x m) matrix $\boldsymbol{B}$ such that $\boldsymbol{BA} = \boldsymbol{I}_n$

Right Inverse of a (m × n) matrix $\boldsymbol{A}$ is the (n × m) matrix $\boldsymbol{C}$ such that $\boldsymbol{AC} = \boldsymbol{I}_m$

(T × k) design matrix $\boldsymbol{X}$ which has rank k < T has an infinite number of left inverses including $\boldsymbol{(X^TX)^{-1}X^T}$

Orthogonal Matrices: Two nx1 vectors u and v are said to be orthogonal if

$$u^t v = v^t u = 0$$

A set of nx1 vectors $\{x_1, x_2, \ldots x_n\}$ is said to be orthonormal if

$$x_i^t x_i = 1,\ x_i^t x_j = 0\ ,\ i \neq j,\ i, j = 1, 2, ..., n$$

Definition of orthogonal matrix:

A n x n square matrix $\boldsymbol{P}$ is said to be orthogonal if

$$PP^t \setminus P^t P = I_{n \times m}$$

Positive Definite Matrices

A symmetric $n \times n$ matrix $\boldsymbol{A}$ satisfying

$$x^t_{1\times n} A_{n\times n} x_{n\times 1} > 0 \text{ for all } x \neq 0,$$

is referred to as a positive definite (p.d.) matrix.

Intuition:

If $ax^2 > 0$ for all real numbers x, $x \neq 0$ then the real number a is positive. Similarly, as x is a $n \times 1$ vector, $\boldsymbol{A}$ is a $n \times n$ matrix and $x^t Ax > 0$, then the matrix $\boldsymbol{A}$ is "positive".

Note:

A symmetric $n \times n$ matrix $\boldsymbol{A}$ satisfying

$$x^t_{1\times n} A_{n\times n} x_{n\times 1} \geq 0 \text{ for all } x \neq 0,$$

is referred to as a positive semi definite (p.d.) matrix.

Kronecker Product

$$\boldsymbol{A} \otimes \boldsymbol{B}$$

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 0 \end{bmatrix}$$

$$\mathrm{B} = \begin{bmatrix} 2 & 2 & 0 \\ 1 & 0 & 3 \end{bmatrix}$$

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} 1\begin{bmatrix} 2 & 2 & 0 \\ 1 & 0 & 3 \end{bmatrix} & 3\begin{bmatrix} 2 & 2 & 0 \\ 1 & 0 & 3 \end{bmatrix} \\ 2\begin{bmatrix} 2 & 2 & 0 \\ 1 & 0 & 3 \end{bmatrix} & 0\begin{bmatrix} 2 & 2 & 0 \\ 1 & 0 & 3 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 2 & 0 & 6 & 6 & 0 \\ 1 & 0 & 3 & 3 & 0 & 9 \\ 4 & 4 & 0 & 0 & 0 & 0 \\ 2 & 0 & 6 & 0 & 0 & 0 \end{bmatrix}$$

9.5.4 Eigen Values and Vectors

Let A be a $n \times n$ matrix. The real number λ is called an eigen value of A if there exists a *nonzero* vector x in R^n such that

$$Ax = \lambda x.$$

The nonzero vector x is called an eigenvector of A associated with the eigenvalue λ.

■ **Example**

Let

$$A = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}.$$

As $x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, then $Ax = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \end{bmatrix} = 3x.$

Thus, $x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is the eigenvector of A associated with the eigenvalue $\lambda = 3$.

Similarly,

As $x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, then

$$Ax = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} = 2x.$$

Thus, $x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ is the eigenvector of A associated with the eigenvalue $\lambda = 2$.

Note: Let x be the eigenvector of $\boldsymbol{A}$ associated with some eigenvalue λ. Then, $cx, x \in R, c \neq 0$, is also the eigenvector of $\boldsymbol{A}$ associated with the same eigenvalue λ since

$$A(cx) = cAx = c\lambda x = \lambda(cx).$$

Computation of Eigenvalues and Eigenvectors:

■ **Example**

Let

$$A = \begin{bmatrix} 1 & 1 \\ -2 & 4 \end{bmatrix}$$

Let $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ be the eigenvector associated with the eigenvalue λ. Then,

$$Ax = \begin{bmatrix} 1 & 1 \\ -2 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda x = (\lambda I)x \Leftrightarrow (\lambda I)x - Ax$$

$$= (\lambda I - A)\, x = 0.$$

Thus,

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ is the nonzero (nontrivial) solution of the homogeneous linear system $(\lambda I - A)\, x = 0$. $\Leftrightarrow \lambda I - A$ is singular $\Leftrightarrow \det(\lambda I - A)\, x = 0$.

Therefore,

$$\det(\lambda I - A)\, x = \begin{vmatrix} \lambda - 1 & -1 \\ 2 & \lambda - 4 \end{vmatrix} = (\lambda - 3)(\lambda - 2) = 0$$

$\Leftrightarrow \quad \lambda = 2$ or 3.

1. As $\lambda = 2$,

   $Ax = 2x = 2Ix \Leftrightarrow 2Ix - 2x$

   $= (2I - A)\, x = \begin{bmatrix} 1 & -1 \\ 2 & -2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0.$

   $\Leftrightarrow x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} t, \quad t \in R.$

   $\Leftrightarrow \begin{bmatrix} 1 \\ 1 \end{bmatrix} t$, $t \neq 0$, $t \in R$, are the eigenvecto rs associated with $\lambda = 2$

2 As $\lambda = 3$,

   $Ax = 3x = 3Ix \Leftrightarrow 3Ix - \mathrm{A}x$

   $= (3I - A)\, x = \begin{bmatrix} 2 & -1 \\ 2 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0.$

   $\Leftrightarrow x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1 \end{bmatrix} r, \quad r \in R.$

   $\Leftrightarrow \begin{bmatrix} 1/2 \\ 1 \end{bmatrix} r$, $r \neq 0$, $r \in R$, are the eigenvectors associated with $\lambda = 3$

Note

In the above example, the eigenvalues of A satisfy the following equation

$$\det(\lambda I - A) = 0.$$

After finding the eigenvalues, we can further solve the associated homogeneous system to find the eigenvectors.

Definition of the characteristic polynomial

Let $A_{n\times n} = \lfloor a_{ij} \rfloor$. The determinant

$$f(\lambda) = \det(\lambda I - A) = \begin{vmatrix} \lambda - a_{11} & -a_{21} & \cdots & -a_{1n} \\ -a_{21} & \lambda - a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & \lambda - a_{nn} \end{vmatrix},$$

is called the characteristic polynomial of A.

$f(\lambda) = \det(\lambda I - A) = 0$,

is called the characteristic equation of A.

Theorem:

A is singular if and only if 0 is an eigen value of A.

$\Rightarrow$: .

$\boldsymbol{A}$ is singular $\Rightarrow Ax = 0$ has non-trivial solution $\Rightarrow$ There exists a nonzero vector $\boldsymbol{x}$ such that

$$Ax = 0 = 0x.$$

$\Rightarrow$ $\boldsymbol{x}$ is the eigenvector of $\boldsymbol{A}$ associated with eigen value 0.

$\Leftarrow$:

0 is an eigen value of $\boldsymbol{A} \Rightarrow$ There exists a nonzero vector $\boldsymbol{x}$ such that

$$Ax = 0 = 0x.$$

$\Rightarrow$ The homogeneous system $Ax = 0$ has nontrivial (nonzero) solution.

$\Rightarrow$ $\boldsymbol{A}$ is singular.

Theorem:

The eigen values of A are the real roots of the characteristic polynomial of A.

$\Rightarrow$:

Let λ^* be an eigen value of $\boldsymbol{A}$ associated with eigenvector $\boldsymbol{u}$. Also, let $f(\lambda)$ be the characteristic polynomial of $\boldsymbol{A}$. Then,

$Au = \lambda^* u \Rightarrow \lambda^* u - Au = \lambda^* Iu - Au = (\lambda^* I - A)\, u = 0 \Rightarrow$

The homogeneous system has nontrivial (nonzero) solution $x \Rightarrow \lambda^* I - A$ is singular $\Rightarrow$

$$\det(\lambda^* I - A) = f(\lambda^*) = 0.$$

$\Rightarrow \lambda^*$ is a real root of $f(\lambda) = 0$.

$\Leftarrow$:

Let λ_r be a real root of $f(\lambda) = 0 \Rightarrow f(\lambda_r) = \det(\lambda_r I - A) = 0 \Rightarrow$ $\lambda_r I - A$ is a singular matrix $\Rightarrow$ There exists a nonzero vector (nontrivial solution) v such that $(\lambda_r I - A)\, v = 0 \Rightarrow Av = \lambda_r v$.

$\Rightarrow v$ is the eigenvector of A associated with the eigenvalue λ_r.

Procedure of finding the eigenvalues and eigenvectors of A:

1. Solve for the real roots of the characteristic equation $f(\lambda) = 0$. These real roots $\lambda_1, \lambda_2, \ldots$ are the eigenvalues of A.
2. Solve for the homogeneous system $(A - \lambda_i I)x = 0$ or $(\lambda_i I - A)\, x = 0$, $i = 1, 2, \ldots$. The nontrivial (nonzero) solutions are the eigenvectors associated with the eigenvalues λ_i.

Theorem:

Let $u_1, u_2, \ldots, u_k$ be the eigenvectors of a $n \times n$ matrix A associated with distinct eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_k$, respectively, $k \le n$. Then, $u_1, u_2, \ldots, u_k$ are linearly independent.

Proof:

Assume $u_1, u_2, \ldots, u_k$ are linearly dependent. Then, suppose the dimension of the vector space V generated by $u_1, u_2, \ldots, u_k$ is $j < k$

(i.e. the dimension of $V\left\{u \mid u = \sum_{i=1}^{k} c_i u_k,\ c_i \in R,\ i = 1, 2, \ldots, k\right\} \equiv$

the vector space generated by $u_1, u_2, \ldots, u_k$). There exists $\boldsymbol{j}$ linearly independent vectors of $u_1, u_2, \ldots, u_k$ which also generate V. Without loss of generality, let $u_1, u_2, \ldots, u_j$ be the j linearly independent vectors which generate V (i.e., $u_1, u_2, \ldots, u_j$ is a basis of V). Thus,

$$u_{j+1} = \sum_{i=1}^{j} a_i\, u_i\ ,$$

$a'_i s$ are some real numbers. Then,

$$Au_{j+1} = A\left(\sum_{i=1}^{k} a_i\, u_i\right) = \sum_{i=1}^{j} a_i\, Au_i = \sum_{i=1}^{j} a_i\, \lambda_i\, u_i$$

Also,

$$Au_{j+1} = \lambda_{j+1} u_{j+1} = \lambda_{j+1} \sum_{i=1}^{j} a_i\, u_i = \sum_{i=1}^{j} a_i\, \lambda_{j+1}\, u_i$$

Thus,

$$\sum_{i=1}^{j} a_i\, \lambda_i\, u_i = \sum_{i=1}^{j} a_i\, \lambda_{j+1}\, u_i \Leftrightarrow \sum_{i=1}^{j} a_i \left(\lambda_i - \lambda_{j+1}\right) u_i = 0$$

Since $u_1, u_2, \ldots, u_k$ are linearly independent,

$$a_1(\lambda_{j+1} - \lambda_1) = a_2(\lambda_{j+1} - \lambda_2) = \ldots . a_j(\lambda_{j+1} - \lambda_1) = 0.$$

Furthermore,

$\lambda_1, \lambda_2, \ldots, \lambda_j$ are distinct, $\lambda_{j+1} - \lambda_1 \ne 0$, $\lambda_{j+1} - \lambda_2 \ne 0$, ..., $\lambda_{j+1} - \lambda_1 \ne 0$

$$\Rightarrow \quad a_1 = a_2 = \ldots . = a_j = 0 \Rightarrow u_{j+1} - \sum_{i=1}^{j} a_i u_i = 0.$$

It is contradictory.

Corollary:

If a $n \times n$ matrix A has n distinct eigenvalues, then A has n linearly independent eigenvectors.

Properties of Eigenvalues and Eisssgenvectors:

(a) Let u be the eigenvector of $A_{n\times n}$ associated with the eigen value λ. Then, the eigen value of $a_k A^k + a_{k-1} A^{k-1} + ... + a_1 A + a_0 I$, associated with the eigenvector u is $a_k \lambda^k + a_{k-1} \lambda^{k-1} + ... + a_1 \lambda + a_0$, where a_k, a_{k-1}, ..., a_1, a_0 are real numbers and k is a positive integer.

Proof:

$$(a_k A^k + a_{k-1} A^{k-1} + ... + a_1 A + a_0 I)$$
$$u = a_k A^k u + a_{k-1} A^{k-1} u + ...+ a_1 Au + a_0 u$$
$$= a_k \lambda^k u + a_{k-1} \lambda^{k-1} u + ...+ a_1 \lambda u + a_0 u$$
$$= (a_k \lambda^k + a_{k-1} \lambda^{k-1} u + ...+ a_1 \lambda + a_0) u$$

since

$$A^j u = A^{j-1} (Au) = A^{j-1} \lambda u = lA^{j-1} u = \lambda A^{j-2} (Au)$$
$$= \lambda^2 A^{j-2} u = ... = \lambda^{j-1} Au = \lambda^j u$$

■ **Example**

$$\mathbf{A} = \begin{bmatrix} 1 & 4 \\ 9 & 1 \end{bmatrix},$$

what is the eigen values of $2A^{100} + 4A - 12I$.

■ **Solution** The eigenvalues of ***A*** are –5 and 7. Thus, the eigenvalues of ***A*** are

$$2(-5)^{100} + 4(-5) - 12 = 2.5^{100} - 32$$

and

$$2(7)^{100} + 4(7) - 12 = 2.7^{100} + 16$$

■ **Example** Let λ be the eigenvalue of ***A***. Then, we denote

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + ... + \frac{A^n}{n!} + ... = \frac{\sum_{i=0}^{\infty} A^i}{i!}$$

Then, e^A has eigenvalue

$$e^A = I + \lambda + \frac{\lambda^2}{2!} + \frac{\lambda^3}{3!} + ... + \frac{\lambda^n}{n!} + ... = \frac{\sum_{i=0}^{\infty} \lambda^i}{i!}.$$

Note:

Let u be the eigenvector of ***A*** associated with the eigenvalue λ. Then, u is the eigenvector of A^{-1}associated with the eigen value $\lambda^{-1} = \frac{1}{\lambda}$.

Proof:

$$A^{-1} u = A^{-1}(\lambda u)\frac{1}{\lambda} = A^{-1}(Au)\frac{1}{\lambda} = I\frac{1}{\lambda}u = \frac{1}{\lambda}u.$$

Therefore, u is the eigenvector of A^{-1} associated with the eigenvalue $\lambda^{-1} = \frac{1}{\lambda}$.

(b) Let $\lambda_1, \lambda_2, ... , \lambda_n$ be the eigen values of A ($\lambda_1, \lambda_2, ... , \lambda_n$ are not necessary to be distinct). Then,

$$\sum_{i=1}^{n} \lambda_i = tr(A) \text{ and } \prod_{i=1}^{n} \lambda_i = \det(A) = |A|.$$

Proof:

$$f(\lambda) = \det(\lambda I - A) = (\lambda - \lambda_1)(\lambda - \lambda_2) ... (\lambda - \lambda_n)$$

Thus,

$$f(0) = \det(-A) = (-1)^n \det(A) = (0 - \lambda_1)(0 - \lambda_2) ... (0 - \lambda_n)$$
$$= (-1)^n \lambda_1 \lambda_2 ... \lambda_n = (-1)^n \prod_{i=1}^{n} \lambda_i$$

Therefore,

$$\det(A) = \prod_{i=1}^{n} \lambda_i .$$

Also, by diagonal expansion on the following determinant

$$f(\lambda) = \begin{vmatrix} -a_{11} + \lambda & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & -a_{22} + \lambda & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & -a_{nn} + \lambda \end{vmatrix}$$
$$= \lambda^n - \left(\sum_{i=1}^{n} a_{ii}\right)\lambda^{n+1} + \cdots + (-1)^n \prod_{i=1}^{n} \lambda_i$$

and by the expansion of $f(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2) ... (\lambda - \lambda_n)$

$$= \lambda^n - \left(\sum_{i=1}^{n} \lambda_i\right)\lambda^{n+1} + \cdots + (-1)^n \prod_{i=1}^{n} \lambda_i$$

therefore,

$$\sum_{i=1}^{n} \lambda_i = \sum_{i=1}^{n} a_{ii} = tr(A).$$

Some important points about eigenvectors and eigenvalues are:

1. The eigenvectors are scaled so that **A** is an orthogonal matrix. Thus, $\mathbf{A}^t = \mathbf{A}^{-1}$, and $\mathbf{A}\mathbf{A}^t = \mathbf{I}$. Thus, each eigenvector is said to be orthogonal to all the other eigenvectors.
2. The eigenvalues will all be greater than 0.0, providing that the four conditions outlined above for **C** are true.
3. For a covariance matrix, the sum of the diagonal elements of the covariance matrix equals the sum of the eigenvalues, or in math terms, tr(**C**) = tr(**D**). For a correlation matrix, all the eigenvalues sum to n, the number of variables. Furthermore, in case you have a burning passion to know about it, the determinant of **C** equals the product of the eigenvalues of **C**.
4. The decomposition of a matrix into its eigenvalues and eigenvectors is a mathematical/geometric decom-

position. The decomposition literally rearranges the dimensions in an n dimensional space (n being the number of variables) in such a way that the axis (e.g., North-South, East-West) are all perpendicular. This rearrangement may but is not guaranteed to uncover an important psychological construct or even to have a psychologically meaningful interpretation.

5. An eigenvalue tells us the proportion of total variability in a matrix associated with its corresponding eigenvector. Consequently, the eigenvector that corresponds to the highest eigenvalue tells us the dimension (axis) that generates the maximum amount of individual variability in the variables. The next eigenvector is a dimension perpendicular to the first that accounts for the second largest amount of variability, and so on.

9.5.5 Diagonalisation of a Matrix

A matrix A is diagonalisable if there exists a nonsingular matrix P and a diagonal matrix D such that

$$D = P^{-1} AP.$$

■ **Example**

Let

$$A = \begin{bmatrix} -4 & -6 \\ 3 & 5 \end{bmatrix}.$$

Then,

$$A = \begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & -2 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -4 & -6 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} -1 & -2 \\ 1 & 1 \end{bmatrix} = P^{-1} AP,$$

where

$$D = \begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix}, \; P = \begin{bmatrix} -1 & -2 \\ 1 & 1 \end{bmatrix}.$$

Theorem:

An $n \times n$ matrix A is diagonalisable if and only if it has n linearly independent eigenvector.

Proof:

$\Rightarrow$ **:**

A is diagonalisable. Then, there exists a nonsingular matrix ***P*** and a diagonal matrix

$$D = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix},$$

such that

$$D = P^{-1} AP \Leftrightarrow AP = PD$$

$$\Leftrightarrow \quad A = [col_1 (P)\ col_2 (P) \ldots col_n (P)]$$

$$= [col_1 (P) \ldots col_n(P)] \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}.$$

Then,

$$Acol_j (P) = \lambda_j col_j (P), j = 1, 2, ..., n.$$

That is,

$$col_1, col_2 (P), ..., col_n(P)$$

are eigenvectors associated with the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$.

Since ***P*** is nonsingular, thus $col_1 (P)\ col_2 (P) \ldots col_n (P)$ are linearly independent.

$\Leftarrow$ **:**

Let $x_1, x_2, ..., x_n$ be ***n*** linearly independent eigenvectors of ***A*** associated with the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$. That is,

$$Ax_j = \lambda_j x_j, j = 1, 2, ..., n.$$

Thus, let

$$P = [x_1\ x_2 .... x_n] \text{ (i.e., } col_j (P) = x_j)$$

and

$$D = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}.$$

Since $Ax_j = \lambda_j x_j$,

$$AP = A[x_1 x_2 ..... x_n]$$

$$= [x_1 x_2 ...x_n] \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} = PD.$$

Thus,

$$P^{-1} AP = P^{-1} AP = D,$$

P^{-1} exists because $x_1 x_2 ..... x_n$ are linearly independent and thus ***P*** is nonsingular.

Important result:

An $n \times n$ matrix A is diagonalisable if all the roots of its characteristic equation are real and distinct.

■ **Example**

Let

$$A = \begin{bmatrix} -4 & -6 \\ 3 & 5 \end{bmatrix}.$$

Find the nonsingular matrix ***P*** and the diagonal matrix ***D*** such that

$$D = P^{-1} AP$$

and find A^n, n is any positive integer.

■ **Solution**

We need to find the eigenvalues and eigenvectors of $\boldsymbol{A}$ first. The characteristic equation of $\boldsymbol{A}$ is

$$\det(\lambda I - A) = \begin{vmatrix} \lambda+4 & 6 \\ -3 & \lambda-5 \end{vmatrix} = (\lambda+1)(\lambda-2) = 0.$$

$$\Rightarrow \quad \lambda = -1 \text{ or } 2.$$

By the above important result, $\boldsymbol{A}$ is diagonalisable. Then,

1. As $\lambda = 2$,

$$Ax = 2x \Leftrightarrow (2I - A)\,x = 0 \Leftrightarrow x = r\begin{bmatrix} -1 \\ 1 \end{bmatrix},\ r \in R.$$

2. As $\lambda = -1$,

$$Ax = -x \Leftrightarrow (-I - A)\,x = 0 \Leftrightarrow x = t\begin{bmatrix} -2 \\ 1 \end{bmatrix},\ t \in R.$$

Thus,

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

are two linearly independent eigenvectors of $\boldsymbol{A}$.

Let

$$P = \begin{bmatrix} -1 & -2 \\ 1 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix}.$$

Then, by the above theorem,

$$D = P^{-1}AP.$$

To find A^n,

$$D^n = \begin{bmatrix} 2^n & 0 \\ 0 & (-1)^n \end{bmatrix} = (P^{-1}AP)(P^{-1}AP) \ldots (P^{-1}AP) = P^{-1}A^nP$$

n times

Multiplied by P and P^{-1} on the both sides,

$$PD^nP^{-1} = PP^{-1}A^nPP^{-1}$$

$$= A^n = \begin{bmatrix} -1 & -2 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 2^n & 0 \\ 0 & (-1)^n \end{bmatrix}\begin{bmatrix} -1 & -2 \\ 1 & 1 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} -\left[2^n + 2\cdot(-1)^{n+1}\right] & -\left[2^{n+1} + 2\cdot(-1)^{n+1}\right] \\ 2^n + (-1)^{n+1} & 2^{n+1} + (-1)^{n+1} \end{bmatrix}$$

Note:

For any $n \times n$ diagonalisable matrix A, $D = P^{-1}AP$ **then**

$$A^k = PD^kP^{-1},\ k = 1, 2, \ldots.$$

where

$$D^k = \begin{bmatrix} \lambda_1^k & 0 & \cdots & 0 \\ 0 & \lambda_2^k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n^k \end{bmatrix}.$$

■ **Example**

Is $A = \begin{bmatrix} 5 & -3 \\ 3 & -1 \end{bmatrix}$ diagonalisable?

■ **Solution:**

$$\det(\lambda I - A) = \begin{vmatrix} \lambda-5 & 3 \\ -3 & \lambda+1 \end{vmatrix} = (\lambda-2)^2 = 0.$$

Then, $\lambda = 2, 2.$

As $\lambda = 2,$

$$(2I - A)\,x = 0 \Leftrightarrow x = t\begin{bmatrix} 1 \\ 1 \end{bmatrix},\ t \in R.$$

Therefore, all the eigenvectors are spanned by $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$. There *does not exist* two linearly independent eigenvectors. By the previous theorem, *A is not diagonalisable.*

Note:

An $n \times n$ matrix may fail to be diagonalisable since

- ***Not*** all roots of its characteristic equation are real numbers.
- It does not have ***n*** linearly independent eigenvectors.

Note:

The set S_j consisting of both all eigenvectors of an $n \times n$ matrix $\boldsymbol{A}$ associated with eigenvalue λ_j and zero vector 0 is a subspace of R^n. S_j is called the ***eigenspace*** associated with λ_j.

(b) *Diagonalisation of symmetric matrix*

Theorem:

If A is an $n \times n$ symmetric matrix, then the eigenvectors of A associated with distinct eigenvalues are orthogonal.

Proof:

Let $x_1 = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$ and $x_2 = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ be eigenvectors of $\boldsymbol{A}$ associated with distinct eigenvalues λ_1 and λ_2, respectively, i.e.,

$$Ax_1 = \lambda_1 x_1 \text{ and } Ax_2 = \lambda_2 x_2.$$

Thus,

$$x_1^tAx_2 = x_1^t(Ax_2) = x_1^t\lambda_2x_2 = \lambda_2x_1^tx_2$$

and

$$x_1^tAx_2 = x_1^tA^tx_2 = (x_1^tA^t)x_2 = (Ax_1)^t x_2$$

$$= (\lambda_1x_1)^t x_2 = \lambda_1x_1^tx_2$$

Therefore,

$$x_1^tAx_2 = \lambda_2x_1^tx_2 = \lambda_1x_1^tx_2.$$

Since $\lambda_1 = \lambda_2$, $x_1^t x_2 = 0$.

■ **Example**

Let

$$A = \begin{bmatrix} 0 & 0 & -2 \\ 0 & -2 & 0 \\ -2 & 0 & 3 \end{bmatrix}.$$

A is a symmetric matrix. The characteristic equation is

$$\det(\lambda I - A) = \begin{vmatrix} \lambda & 0 & 2 \\ 0 & \lambda+2 & 0 \\ 2 & 0 & \lambda-3 \end{vmatrix}$$

$$= (\lambda + 2)(\lambda - 4)(\lambda + 1) = 0.$$

The eigenvalues of ***A*** are –2, 4, –1. The eigenvectors associated with these eigenvalues are

$$x_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} (\lambda = 2),\ x_2 = \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix} (\lambda = 4),\ x_3 = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} (\lambda = -1).$$

Thus,
x_1, x_2, x_3 are orthogonal.

Important Result:

If A is an n × n symmetric matrix, then there exists an orthogonal matrix P such that

$$D = P^{-1} AP = P^t AP,$$

where $col_1(P)$, $col_2(P)$, ..., $col_n(P)$ are n linearly independent eigenvectors of A and the diagonal elements of D are the eigenvalues of A associated with these eigenvectors.

■ **Example**

Let

$$A = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}.$$

Find an orthogonal matrix ***P*** and a diagonal matrix ***D*** such that $D = P^t AP$.

■ **Solution**

We need to find the orthonormal eigenvectors of ***A*** and the associated eigenvalues first. The characteristic equation is

$$f(\lambda) = \det(\lambda I - A) = \begin{vmatrix} \lambda & -2 & -2 \\ -2 & \lambda & -2 \\ -2 & -2 & \lambda \end{vmatrix}$$

$$= (\lambda + 2)^2 (\lambda - 4) = 0$$

Thus, λ = –2, –2, 4.

1. As λ = –2, solve for the homogeneous system $(-2I - A)\,x = 0$.

   The eigenvectors are

$$t\begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} + s\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix},\ t, s \in R,\ t \neq 0 \text{ or } s \neq 0$$

$\begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ are two eigenvectors of ***A***. However, the two eigenvectors are not orthogonal. We can obtain two orthonormal eigenvectors via ***Gram-Schmidt process***. The orthogonal eigenvectors are

$$v_1^* = v_1 = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$

$$v_2^* = v_2 - \frac{v_2 \cdot v_1^*}{v_1^* \cdot v_1^*} v_1^* = \begin{bmatrix} -1/2 \\ -1/2 \\ 1 \end{bmatrix}$$

Standardising these two eigenvectors results in

$$w_1 = \frac{v_1^*}{\|v_1^*\|} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}$$

$$w_2 = \frac{v_2^*}{\|v_2^*\|} = \begin{bmatrix} -1/\sqrt{6} \\ 1/\sqrt{6} \\ 2/\sqrt{6} \end{bmatrix}$$

2. As λ = 4, solve for the homogeneous system $(4I - A)\,x = 0$.

   The eigenvectors are

$$r\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},\ r \in R,\ r \neq 0.$$

$$\Rightarrow v_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$ is an eigenvector of ***A***. Standardising the eigenvector results in $w_3 = \frac{v_3}{\|v_3\|} = \begin{bmatrix} 1/\sqrt{3} \\ 1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix}$.

Thus,

$$P = [w_1\ w_2\ w_3] = \begin{bmatrix} -1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 0 & 2/\sqrt{6} & 1/\sqrt{3} \end{bmatrix}$$

$$D = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

and $D = P^t AP$.

Note

For a set of vectors $v_1, v_2, ..., v_n$ we can find a set of orthogonal vectors $v_1^*, v_2^*, \ldots, v_n^*$ via **Gram-Schmidt process:**

$$v_1^* = v_1$$

$$v_2^* = v_2 - \frac{v_2 \cdot v_1^*}{v_1^* \cdot v_1^*} v_1^*$$

$$\vdots$$

$$v_1^* = v_1 - \frac{v_i \cdot v_{i-1}^*}{v_{i-1}^* \cdot v_{i-1}^*} v_{i-1}^* - \frac{v_i \cdot v_{i-2}^*}{v_{i-2}^* \cdot v_{i-2}^*} v_{i-2}^* - \cdots - \frac{v_i \cdot v_2^*}{v_2^* \cdot v_2^*} v_2^* - \frac{v_n \cdot v_1^*}{v_1^* \cdot v_1^*} v_1^*$$

$$\vdots$$

$$v_n^* = v_n - \frac{v_n \cdot v_{n-1}^*}{v_{n-1}^* \cdot v_{n-1}^*} v_{n-1}^* - \frac{v_n \cdot v_{n-2}^*}{v_{n-2}^* \cdot v_{n-2}^*} v_{n-2}^* - \cdots - \frac{v_n \cdot v_2^*}{v_2^* \cdot v_2^*} v_2^* - \frac{v_n \cdot v_1^*}{v_1^* \cdot v_1^*} v_1^*$$

- **Diagonal form and Jordan form:** A square matrix $\underline{A}$ can be transformed into a diagonal or block diagonal form:

  $\underline{\breve{A}} = \underline{Q}^{-1}\, \underline{A}\, \underline{Q}$ where $\underline{Q} = \left[\underline{q}_1\, \underline{q}_2 \cdots \underline{q}_n\right]$ and $\underline{q}_1\, \underline{q}_2 \cdots \underline{q}_n$ are eigenvectors of $\underline{A}$.
  - Distinct real λs: All real diagonal elements, every element corresponds to a *l*.
  - Distinct complex λs: All real/complex elements, every element corresponds to a *l*. Additional transform can remove the imaginary parts but the transformed matrix becomes block diagonal (modal form).
  - Repeated λs: If the λs are not all distinct, the resulting matrix may comprise upper triangular blocks along the diagonal (Jordan form).
- **Norm of a matrix :** magnification capability $\underline{A}_{m\times n}$
  - $||\underline{A}||_1$ = Largest column absolute sum
  - $||\underline{A}||_2$ = Largest singular value
  - $||\underline{A}||_\infty$ = Largest row absolute sum
  - Matlab usages: norm (a, 1), norm (a, 2) = norm (a), and norm (a, inf)
- **Singular-value Decomposition (SVD):**
  - Eigenvalues/vectors are defined for square matrices only.
  - Non-square matrices are important in linear system analysis (controllability/observability)
  - Foe an $m \times n$ matrix $\underline{H}$, we can define a symmetric matrix $(n \times n)$ $\underline{M} = \underline{H}^{\mathrm{T}}\, H$ and the eigenvalues of $\underline{M}$ are real and nonnegative (positive semi definite).
  - The eigenvalues of $\underline{M}$ are called the singular values of $\underline{H}$.
  - SVD: $\underline{H}$ can be decomposed into a product of 3 matrices.
    - $\underline{H} = \underline{R}\, \underline{S}\, \underline{Q}^{\mathrm{T}}$
    - $\underline{R}\, \underline{R}^{\mathrm{T}} = \underline{R}^{\mathrm{T}}\, \underline{R} = \underline{I}_{\mathrm{m}}$ $\underline{Q}\, \underline{Q}^{\mathrm{T}} = \underline{Q}^{\mathrm{T}}\, \underline{Q} = \underline{I}_{\mathrm{n}}$
    - S is an $m \times n$ matrix with the singular values of $\underline{H}$ on the diagonal.
  - Applications of SVD:
    - Norm of a matrix: $||\underline{A}||_2 = \sigma_1$ (Largest singular value)
    - Rank of a matrix: equal to the number of non-zero singular values.
    - Condition number = $\sigma_{\max} / \sigma_{\min}$: Indicates how close a matrix is to rank deficiency (How much numerical error is likely to be introduced by computations involving the matrix). Large condition numbers imply ill-conditioned systems and small (close to 1) condition numbers imply well-conditioned systems.
- **Lyapunov theorem**
  - The Lyapunov theorem provides an alternate means to check the asymptotic stability of a system.
  - Lyapunov equation:

    $\underline{\boldsymbol{A}}(\underline{\mathrm{M}}) = \underline{\mathrm{A}}\, \underline{\mathrm{M}} + \underline{\mathrm{M}}\, \underline{\mathrm{B}} = \underline{\mathrm{C}}$
  - $\underline{\boldsymbol{A}}(\underline{\mathrm{M}}) = \eta\, \underline{\mathrm{M}}$, where η's are the eigenvalues of $\underline{\boldsymbol{A}}$ and they represent all possible sums of the eigenvalues of $\underline{\mathrm{A}}$ and $\underline{\mathrm{B}}$.
  - A symmetric matrix $\underline{\mathrm{M}}$ is said to be positive definite (denoted by $\underline{\mathrm{M}} > 0$) if $\underline{\mathrm{x}}^{\mathrm{T}}\underline{\mathrm{M}}\, \underline{\mathrm{x}} > 0$ for nonzero x.
  - If $\underline{\mathrm{M}} > 0$, then $\underline{\mathrm{x}}^{\mathrm{T}}\underline{\mathrm{M}}\, \underline{\mathrm{x}} = 0$ iff $\underline{\mathrm{x}} = 0$.
  - $\underline{\mathrm{M}} > 0$ iff any one of the following conditions holds:
    - Every eigenvalue of $\underline{\mathrm{M}}$ is positive.
    - All leading principal minors of $\underline{\mathrm{M}}$ are positive.
    - There exists an $n \times n$ nonsingular matrix such that $\underline{\mathrm{M}} = \underline{\mathrm{N}}^{\mathrm{T}}\underline{\mathrm{N}}$.

9.5.6 Linear Independence

Spanning Sets

Definition of linear combination:

Let $v_1, v_2, ..., v_k$ be vectors in a real vector space V. A vector v in V is called a *linear combination* of $v_1, v_2, ..., v_k$ if

$$v = c_1 v_1 + c_2 v_2 + ... + c_k v_k,$$

where $c_1 c_2,...,c_k$.

■ **Example**

Let

$$e_1 = \begin{bmatrix}1\\0\\0\end{bmatrix}, e_2 = \begin{bmatrix}0\\1\\0\end{bmatrix}, e_3 = \begin{bmatrix}0\\0\\1\end{bmatrix}, v = \begin{bmatrix}1\\2\\3\end{bmatrix}$$

Since

$$v = \begin{bmatrix}1\\2\\3\end{bmatrix} = 1\cdot\begin{bmatrix}1\\0\\0\end{bmatrix} + 2\cdot\begin{bmatrix}0\\1\\0\end{bmatrix} + 3\cdot\begin{bmatrix}0\\0\\1\end{bmatrix} = 1e_1 + 2e_2 + 3e_3$$

v is a linear combination of e_1, e_2, e_3.

■ **Example**

Let

$$v_1 = \begin{bmatrix}0 & 2\\1 & 0\end{bmatrix} = v_2 = \begin{bmatrix}0 & 8\\2 & 1\end{bmatrix},$$

$$v_3 = \begin{bmatrix}-2 & 0\\1 & 3\end{bmatrix}, v = \begin{bmatrix}0 & 8\\2 & 1\end{bmatrix}$$

be vectors in the vector space consisting of all 2×2 matrices. Then,

$$v = \begin{bmatrix}0 & 8\\2 & 1\end{bmatrix}$$

$$= 1\cdot\begin{bmatrix}0 & 2\\1 & 0\end{bmatrix} + 2\cdot\begin{bmatrix}-1 & 3\\1 & 2\end{bmatrix} + (-1)\cdot\begin{bmatrix}-2 & 0\\1 & 3\end{bmatrix} = v_1 + 2v_2 - v_3.$$

That is, $\boldsymbol{v}$ is a linear combination of v_1, v_2, v_3.

■ **Example**

For linear system

$$Ax = \begin{bmatrix}1 & 0 & -1\\2 & 1 & 0\\3 & 2 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} = b$$

$\Rightarrow$ $d = \begin{bmatrix}2\\-3\\1\end{bmatrix}$ is a solution for the above linear system. Thus,

$$A\begin{bmatrix}2\\-3\\1\end{bmatrix} = \begin{bmatrix}1 & 0 & -1\\2 & 1 & 0\\3 & 2 & 1\end{bmatrix}\begin{bmatrix}2\\-3\\1\end{bmatrix} = 2\cdot\begin{bmatrix}1\\2\\3\end{bmatrix} - 3\cdot\begin{bmatrix}0\\1\\2\end{bmatrix} + 1\begin{bmatrix}-1\\0\\1\end{bmatrix}$$

$$= 2\,col_1(A) - 3\,col_2(A) + col_1(A) = \begin{bmatrix}1\\1\\1\end{bmatrix} = b$$

That is, $\boldsymbol{b}$ is a linear combination of the column vectors of $\boldsymbol{A}$,

$$col_1(A), col_2(A) + col_3(A).$$

Note:

For a linear system $A_{m\times n}\, x_{n\times 1} = b_{m\times 1}$, the linear system has solution or solutions $\Leftrightarrow \boldsymbol{b}$ is a linear combination of the column vectors of $\boldsymbol{A}$,

$$col_1(A), col_2(A), ..., col_3(A).$$

For example, if $c = \begin{bmatrix}c_1\\c_2\\\vdots\\c_n\end{bmatrix}$ is a solution of $A_{m\times n}\, x_{n\times 1} = b_{m\times 1}$,

then $c_1\, col_1(A) + c_2\, col_2(A) + ... + c_n\, col_n(A) = b$

On the other hand, the linear system has no solution $\Leftrightarrow \boldsymbol{b}$ is not a linear combination of the column vectors of $\boldsymbol{A}$

■ **Example**

Is the vector $v = \begin{bmatrix}4\\5\\5\end{bmatrix}$ a linear combination of the vectors

$$v_1 = \begin{bmatrix}1\\2\\3\end{bmatrix}, v_2 = \begin{bmatrix}-1\\1\\4\end{bmatrix}, v_3 = \begin{bmatrix}3\\3\\2\end{bmatrix}.$$

■ **Solution**

We need to find the constants c_1, c_2, c_3 such that

$$v = \begin{bmatrix}4\\5\\5\end{bmatrix} = c_1\begin{bmatrix}1\\2\\3\end{bmatrix} + c_2\begin{bmatrix}-1\\1\\4\end{bmatrix} + c_3\begin{bmatrix}3\\3\\2\end{bmatrix}$$

$$= c_1 v_1 + c_2 v_2 + c_3 v_3$$

$\Leftrightarrow$ we need to solve for the linear system

$$A\begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix} = \begin{bmatrix}1 & -1 & 3\\2 & 1 & 3\\3 & 4 & 2\end{bmatrix}\begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix} = \begin{bmatrix}4\\5\\5\end{bmatrix}.$$

The solutions are

$$c_1 = 2t + 3, c_2 = t - 1, c_3 = t, t \in R.$$

Thus,

$$v = (-2t + 3)\, v_1 + (t - 1)\, v_2 + tv_3, t \in R.$$

$\boldsymbol{v}$ is a linear combination of v_1, v_2, v_3 with infinite number of expressions.

■ **Example**

Is the vector $v = \begin{bmatrix} 3 \\ -4 \\ -6 \end{bmatrix}$ a linear combination of the vectors

$$v_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, v_2 = \begin{bmatrix} -1 \\ -1 \\ -2 \end{bmatrix}, v_3 = \begin{bmatrix} 1 \\ 4 \\ 5 \end{bmatrix}.$$

■ **Solution**

We need to find the constants c_1, c_2, c_3 such that

$$v = \begin{bmatrix} 3 \\ -4 \\ -6 \end{bmatrix} = c_1 \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + c_2 \begin{bmatrix} -1 \\ -1 \\ -2 \end{bmatrix} + c_3 \begin{bmatrix} 1 \\ 4 \\ 5 \end{bmatrix}$$

$$= c_1 v_1 + c_2 v_2 + c_3 v_3$$

$\Leftrightarrow$ we need to solve for the linear system

$$\mathrm{A} = A \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 \\ 2 & -1 & 4 \\ 3 & -2 & 5 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -4 \\ -6 \end{bmatrix}.$$

$\Leftrightarrow$ The linear system has ***no*** solution.

$\Leftrightarrow$ v is ***not*** a linear combination of v_1, v_2, v_3

Note:

Let $v_1, v_2, \ldots, v_m$ and $\boldsymbol{v}$ be vectors in R^n and let $\boldsymbol{A}$ be the matrix with column vectors $col_j(A) = v_j$, $j = 1, 2, \ldots, m$. Thus, $Ax = v$ has solution or solutions $\Leftrightarrow$ v is a linear combination of $v_1, v_2, \ldots, v_m$.

$Ax = v$ has no solution $\Leftrightarrow$ v is not a linear combination of $v_1, v_2, \ldots, v_m$.

Definition of spanning set:

Let $S = \{v_1, v_2, \ldots, v_k\}$ be a set of vectors in a real vector space V. Then, the *span of S,* denoted by *span* (S), is the set consisting of *all* the vectors that are linear combinations of $v_1, v_2, \ldots, v_k$. That is,

$$span\ (S) = \{c_1 v_1 + c_2 v_2 + \ldots + c_k v_k \mid c_1, c_2, \ldots, c_k \in R\}.$$

If *span* $(S) = V$, it is said that V is spanned by S or S spans V.

■ **Example**

Let

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \text{ and } S = \{e_1, e_2, e_3\}$$

Then,

$$span\ (S) = \left\{ c_1 e_1 + c_2 e_2 + c_3 e_3 = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \middle| c_1, c_2\ c_3 \in R \right\} = \mathrm{R}^3.$$

■ **Example**

$$v_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, v_2 = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, v_3 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \text{ and } S = \{v_1, v_2, v_3\}.$$

Does *span* $(S) = R^3$?

■ **Solution**

span $(S) = R^3 \Leftrightarrow$ For any vector $v = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \in R^3$, there exist real numbers c_1, c_2, c_3 such that

$$v = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = c_1 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} + c_2 \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} + c_3 \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$= c_1 v_1 + c_2 v_2 + c_3 v_3$$

$\Leftrightarrow$ we need to solve for the linear system

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}.$$

The solution is

$$c_1 = \frac{-2a + 2b + c}{3}, c_2 = \frac{a - b + c}{3},$$

$$c_3 = \frac{4a - b - 2c}{3}.$$

Thus,

$$v = \left(\frac{-2a + 2b + c}{3}\right) v_1 + \left(\frac{a + b + c}{3}\right) v_2 + \left(\frac{4a - b - 2c}{3}\right) v_3.$$

That is, every vector in R^3 can be a linear combination of v_1, v_2, v_3.

Linear Independence

Motivation:

Let $S = \{v_1, v_2, \ldots, v_k\}$ and *span* $(S) = W$**.** Is it possible to find a smaller (or even smallest) set, for example, $S = \{v_1, v_2, \ldots, v_{k-1}\}$**,** such that

$$span\ (S) = W = span\ (S^*)?$$

To answer this question, we need to introduce the concept of *linear independence and linear dependence.*

Definition of linear dependence and linear independence:

The vectors $v_1, v_2, \ldots, v_k$ in a vector space V are said to linearly dependent if there exist constants, $c_1, c_2, \ldots, c_k$, not all 0, such that

$$c_1 v_1 + c_2 v_2 + \ldots + c_k v_k = 0$$

$v_1, v_2, \ldots, v_k$ are linearly independent if

$$c_1 v_1 + c_2 v_2 + \ldots + c_k v_k = 0 \Rightarrow c_1 = c_2 = \ldots + c_k = 0.$$

The procedure to determine if $v_1, v_2, ..., v_k$ are linearly dependent or linearly independent:

1. Form equation $c_1v_1 + c_2v_2 + ... + c_kv_k = 0$, which lead to a homogeneous system.
2. If the homogeneous system has only the trivial solution, then the given vectors are linearly independent; if it has a nontrivial solution, then the vectors are linearly dependent.

■ **Example**

$$e_1 = \begin{bmatrix}1\\0\\0\end{bmatrix}, e_2 = \begin{bmatrix}0\\1\\0\end{bmatrix}, e_3 = \begin{bmatrix}0\\0\\1\end{bmatrix} \text{ and } S = \{e_1, e_2, e_3\}$$

Are e_1, e_2 and e_3 linearly independent?

■ **Solution**

$$c_1 e_1 + c_2 e_2 + ... + c_3 e_3$$

$$= c_1\begin{bmatrix}1\\0\\0\end{bmatrix} + c_2\begin{bmatrix}0\\1\\0\end{bmatrix} + c_3\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix} = 0$$

$$\Rightarrow \begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}.$$

Therefore, e_1 e_2 *and* e_3 *are linearly independent.*

■ **Example**

$$v_1 = \begin{bmatrix}1\\2\\3\end{bmatrix}, v_2 = \begin{bmatrix}-2\\1\\1\end{bmatrix}, v_3 = \begin{bmatrix}8\\6\\10\end{bmatrix}.$$ Are v_1, v_2 and v_3 linearly independent?

■ **Solution**

$$c_1 v_1 + c_2 v_2 + ... + c_3 v_3$$

$$= c_1\begin{bmatrix}1\\2\\3\end{bmatrix} + c_2\begin{bmatrix}-2\\1\\1\end{bmatrix} + c_3\begin{bmatrix}8\\6\\10\end{bmatrix} = \begin{bmatrix}1&-2&8\\2&1&6\\3&1&10\end{bmatrix}\begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix} = 0$$

$$\Rightarrow \begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix} = t\begin{bmatrix}4\\-2\\-1\end{bmatrix}, t \in R.$$

Therefore, v_1 v_2 *and* v_3 *are linearly dependent.*

■ **Example**

Determine whether the following set of vectors in the vector space consisting of all 2×2 matrices is linearly independent or linearly dependent.

$$S = \{v_1, v_2, v_3\} \left\{\begin{bmatrix}2&1\\0&1\end{bmatrix}, \begin{bmatrix}3&0\\2&1\end{bmatrix}, \begin{bmatrix}1&0\\2&0\end{bmatrix}\right\}.$$

■ **Solution:**

$$c_1 v_1 + c_2 v_2 + ... + c_3 v_3$$

$$= c_1\begin{bmatrix}2&1\\0&1\end{bmatrix} + c_2\begin{bmatrix}3&0\\2&1\end{bmatrix} + c_3\begin{bmatrix}1&0\\2&0\end{bmatrix} = \begin{bmatrix}0&0\\0&0\end{bmatrix}.$$

Thus,

$$\begin{aligned} 2c_1 + 3c_2 + c_3 &= 0\\ c_1 &= 0\\ 2c_2 + 2c_3 &= 0\\ c_1 + c_2 &= 0 \end{aligned} \Leftrightarrow c_1\begin{bmatrix}2\\1\\0\\1\end{bmatrix} + c_2\begin{bmatrix}3\\0\\2\\1\end{bmatrix} + c_3\begin{bmatrix}1\\0\\2\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\\0\end{bmatrix}.$$

The homogeneous system is

$$\begin{bmatrix}2&3&1\\1&0&0\\0&2&2\\1&1&0\end{bmatrix}\begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix} = \begin{bmatrix}0\\0\\0\\0\end{bmatrix}.$$

The associated homogeneous system has only the trivial solution

$$\begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}.$$

Therefore, v_1 v_2 *and* v_3 *are linearly independent.*

Important result:

The nonzero vectors $v_1, v_2, ..., v_k$ in a vector space V are linearly dependent if and only if one of the vectors v_j, $j \geq 2$, is a linear combination of the preceding vectors $v_1, v_2, ..., v_{j-1}$.

Note

Every set of vectors containing the zero vector is linearly dependent. That is, $v_1, v_2, ..., v_k$ are ***k*** vectors in any vector space and v_i is the zero vector, then $v_1, v_2, ..., v_k$ are linearly dependent.

9.5.7 Zero-One Matrices

e.g., $\mathbf{A} = \begin{bmatrix}1&0&1\\0&0&1\\1&1&1\end{bmatrix}$ a zero-one matrix

- **Boolean operations on zero-one matrices**
- "$\vee$" **join** $\mathbf{A} \vee \mathbf{B} = [a_{ij} \vee b_{ij}]$ **A**: m×n, **B**: m×n
- "$\wedge$" **meet** $\mathbf{A} \wedge \mathbf{B} = [a_{ij} \wedge b_{ij}]$ **A**: m×n, **B**: m×n
- "$\odot$" **Boolean product** $\mathbf{A} \odot \mathbf{B} = \left[\bigvee_{q=1}^{k} a_{iq}(b_{qi})\right]$ **A**: m×**k**, **B**: **k**×n
- "[r]" **Boolean power** $\mathbf{A}^{[r]} = \mathbf{A} \odot \mathbf{A} \odot \mathbf{A} \odot \ldots \odot \mathbf{A}$ r times A (r–1 $\odot$'s)

Algorithm : The Boolean Product
procedure *Boolean product* (**A**, **B**: zero-one matrices)
for i := 1 **to** m
for j := 1 **to** n
begin
$c_{ij} := 0$
for q := 1 **to** k
$c_{ij} := c_{ij} \vee (a_{iq} \wedge b_{qj})$
end
{C = [c_{ij}] is the Boolean product of **A** and **B**}

9.6 Numerical Methods

9.6.1 Linear Algebraic Equations

We will be given coefficients a11, …and b1, b2,…. Then we are supposed to calculate x1, x2, such that the following equalities are satisfied.

$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1$

$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2$

$\vdots$

$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_nx_n = b_n$

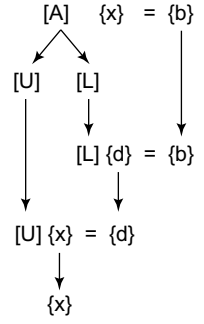
The same thing in matrix form can be represented as:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ & & \vdots & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{Bmatrix}$$

n x n **n x 1** **n x 1**

or simply **[A]{x} = {b}**

LU Decomposition
In essence mathematically LU decomposition is explained as:



(a) decomposition (A) ⇒ [L] [U]

(b) forward substitution [L | U] ⇒ {d}

(c) backward substitution

U is just the upper triangular matrix from Gaussian elimination

[A | b] → [U | b′]

[L] has one's on the diagonal (i.e., it is a "unit lower triangular matrix" and therefore can be denoted [L_1]), and elements below diagonal are just the factors used to scale rows when doing Gaussian elimination, e.g., $\ell_{i1} = a_{i1}/a_{11}$ for i = 2, 3, …, n

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \ell_{21} & 1 & 0 & \cdots & 0 \\ \ell_{31} & \ell_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

then [L1] {d} = {b} ⇒ [U] {x} = {d} in which {d} is synonymous with {b′}

9.6.2 Root Finding Methods

The following methods are used to find root of an algebraic function f.

9.6.2.1 Bisection Method

Given lower and upper bounds, x_l and x_u which bracket the root:

$f(x_l)\, f(x_u) < 0$

(1) Estimate the Root by midpoint: $x_r = \dfrac{x_l + x_u}{2}$

(2) Revise the bracket:

$f(x_l)\, f(x_r) < 0$, $x_r \rightarrow x_u$,

$f(x_l)\, f(x_r) > 0$, $x_r \rightarrow x_l$

(3) Repeat steps 1-2 until:

(a) $|\, f(x_r)\, | \le \delta$

(b) $\varepsilon_a < \varepsilon_s$, with ε_a

$$= \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| \times 100\%$$

(c) $|\, x_u - x_l\, | \le \delta$

(d) maximum # of iterations is reached

Figure 9.4 illustrates the adjustment of root in the given domain.

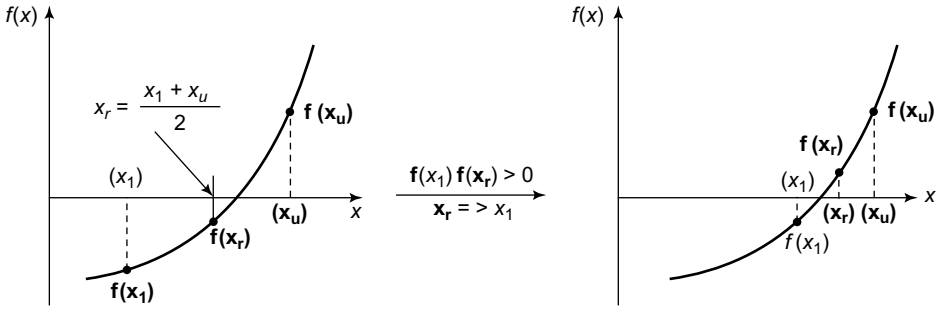


Figure 9.4

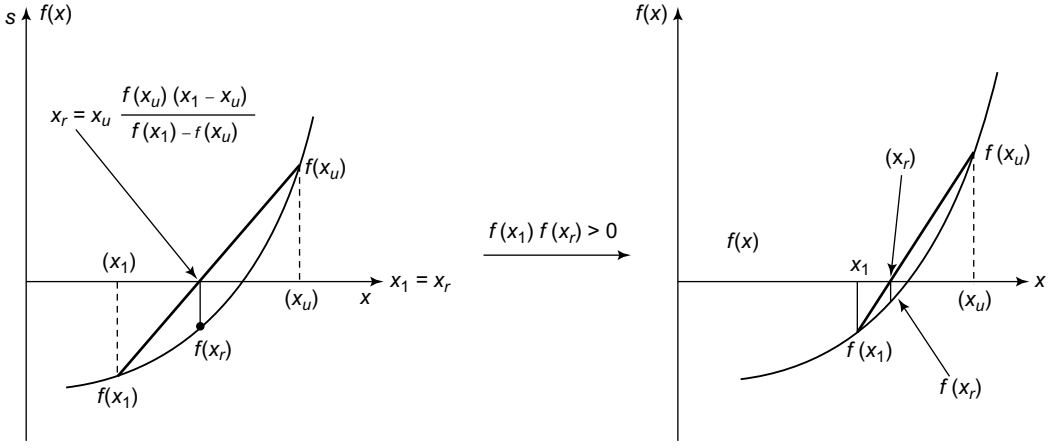


Figure 9.5

Advantages:

1. Simple
2. Good estimate of maximum error

$$|E_{\max}| \le \left|\frac{x_1 - x_u}{2}\right|$$

3. Convergence guaranteed

$$\left|E_{\max}^{i+1}\right| \le 0.5\ 0.5\left|E_{\max}^{i}\right|$$

Disadvantages:

1. Slow
2. Requires initial interval around root:
   - Use graph of function,
   - Incremental search, or
   - Trial & error

9.6.2.2 False-position Method

This is similar to bisection; but uses linear interpolation to approximate root x_r

(1) $x_r = x_u - \dfrac{f(x_u)(x_1 - x_u)}{f(x_1) - f(x_u)}$

(2) Revise the bracket:

$f(x_1)\, f(x_r) < 0, \quad x_r \rightarrow x_u,$

$f(x_1)\, f(x_r) > 0, \quad x_r \rightarrow x_1$

(3) Repeat steps 1-2 until:

(a) $|\, f(x_r)\,| \le \delta$

(b) $\varepsilon_a < \varepsilon_s$, with ε_a

$$= \left|\frac{x_r^{new} - x_r^{old}}{x_r^{new}}\right| \times 100\%$$

(c) $|x_u - x_1| \le \delta$

(d) maximum # of iterations is reached

Figure 9.5 illustrates the adjustment of root in the given domain.

Advantages

1. Simple
2. Brackets root
3. Gives maximum error

Disadvantages

1. Can be very slow
2. Like Bisection, needs an initial interval around the root

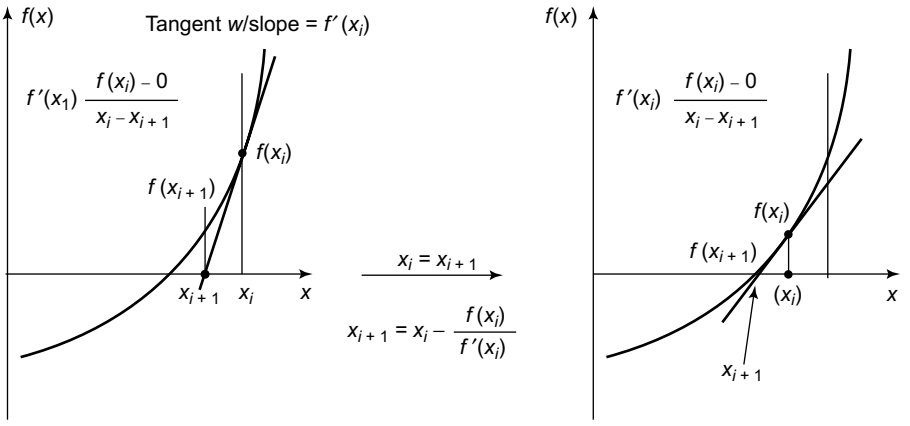


Figure 9.6

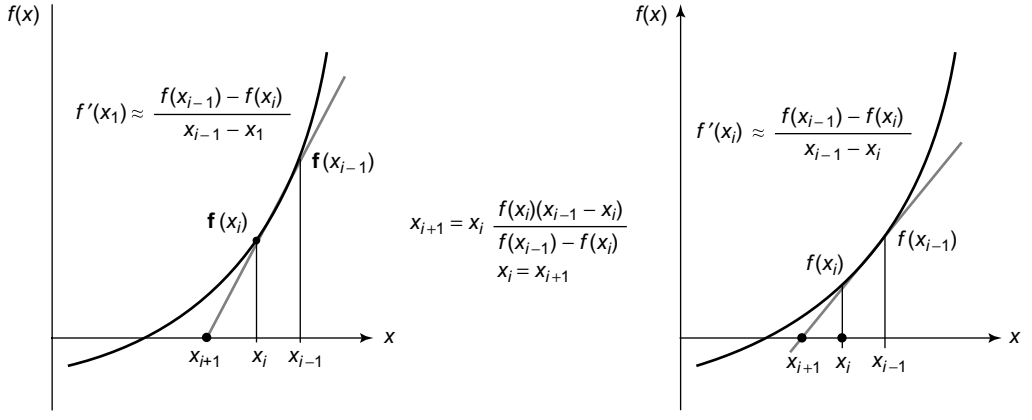


Figure 9.7

9.6.2.3 Newton-Raphson Method

Here, we solve for x_r (root) to yield next guess x_{i+1}

$$x_r \approx x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Newton-Raphson iteration:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \rightarrow x_i^{new} = x_{i+1}^{old}$$

This iteration is repeated until:

1. $f(x) \approx 0$, i.e., $|f(x_{i+1})| \le \kappa$
2. $\varepsilon_a = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \times 100\% \le \varepsilon_s$
3. Max. # iterations is reached

Figure 9.6 illustrates the root adjustment in this method.

9.6.2.4 Secant Method

Secant method solution: Approx. f′(x) with backward FDD:

$$f'(x_i) = \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}$$

Substitute this into the Newton-Raphson equation: $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ to obtain the iterative expression:

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

(1) Requires two initial estimates:
x_{i-1} and x_i These do NOT have to bracket root !

(2) Maintains a strict sequence:

$$x_{i+1} = x_1 - x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

Repeated until:

a. $|f(x_{i+1})| \le \kappa$ with κ = small number

b. $\varepsilon_a = \left|\frac{x_{i-1} - x_i}{x_{i+1}}\right| \times 100\% \le \varepsilon_s$

c. Max. # iterations reached (note no δ)

(3) If x_i and x_{i+1} were chosen to bracket the root, this would be the same as the False-Position Method.

9.6.3 Numerical Integration by Trapezoidal Rule

If y0,y1,y2,…..,yn are the function values of a function f (curve) at x0, x1, …xn then area under the curve using trapezoidal is given as: h/2*(y0+yn+2(y1+y2+…+yn-1), where h is the spacing between the x co-ordinates.

■ **Example** Use the trapezium rule to obtain estimate for the value of $\int_2^6 \frac{x^2}{x-1}dx$ using eight strips.

■ **Answer:** Calculate function value at 9 points of the given domain [2,6].

x	2	2.5	3	3.5	4	4.5	5	5.5	6
$y=\frac{x^2}{x-1}dx$	4	4.167	4.5	4.9	5.333	5.786	6.25	6.722	7.2

Substitute into the trapezium rule formula:

$$A \approx \frac{h}{2}[y_0 + y_n + 2(y_1 + y_2 + \cdots + y_{n-1})]$$

$$\text{Area} \approx \frac{\left(\frac{6-2}{8}\right)}{2}\ [4 + 7.2 + 2(4.167 + 4.5 + 4.9 + 5.333 + 5.786 + 6.25 + 6.722)] = 21.6 \text{ units}^2$$

Numerical Integration by Simpson's Rules

If y0,y1,y2,…..,yn are the function values of a function f (curve) at x0, x1, …xn then area under the curve using Simpson rule is given as: h/2*(y0+yn+4(y1+y3+…)+2(y2+ y4+…), where h is the spacing between the x co-ordinates.

■ **Example** Use Simpson's Rule to approximate the area under the curve $y = \frac{6}{x-2}$ between $x = 3$ and $x - 7$ using four strips.

■ **Answer:** Calculate function y value at five points in the domain [3,7] which makes four strips.

x	3	4	5	6	7
$y=\frac{6}{x-2}$	6	3	2	1.5	1.2

$$A \approx \frac{\left(\frac{7-3}{4}\right)}{3}[6+1.2+4(3+1.5)+2(2)] = 9.73 \text{ units}^2$$

9.7 Introduction to Calculus

9.7.1 Theorems on Limits

Let f **and** g **be functions of a variable** x. Then, if the following limits exist:

$$\lim_{x\to l} f = A, \text{ and } \lim_{x\to l} g = B,$$

- $\lim_{x\to l}(f+g) = A + B.$
- $\lim_{x\to l}(f\,g) = AB.$
- $\lim_{x\to l}\frac{f}{g} = \frac{A}{B}$, **if** B **is not 0.**

In other words:

(1) The limit of a sum is equal to the sum of the limits.

(2) The limit of a product is equal to the product of the limits.

(3) The limit of a quotient is equal to the quotient of the limits, provided the limit of the denominator is not 0.

Also, if c does not depend on x, if c is a constant, then

(4) $\lim_{x\to l} c = c.$

For example, $\lim_{x\to l} 5 = 5$.

To see that, let x approach 4: e.g., $4\frac{1}{2}4\frac{1}{4}4\frac{1}{8}4\frac{1}{16}4\frac{1}{32}$ ..., then the value of 5 or any constant does not change. It is constant.

When c is a constant factor, but f depends on x, then

(5) $\lim_{x\to l} cf = c\lim_{x\to l} f$

A constant factor may pass through the limit sign. (This follows from Theorems 2 and 4.) For example,

$$\lim_{x\to l} 8x^3 = 8\lim_{x\to l} x^3.$$

9.7.2 Continuous Function

We say that a function $f(x)$ that is defined at $x = c$ is continuous at $x = c$ if the limit of $f(x)$ as x approaches c is equal to the value of $f(x)$ at $x = c$. In symbols, if

$$\lim_{x\to l} f(x) = f(c)$$

then, $f(x)$ is continuous at $x = c$.

And so for a function to be continuous at $x = c$, the limit must exist as x **approaches** c, that is, the left- and right-hand limits must be equal. If a function is continuous at every value in an interval, then we say that the function is continuous in that interval. And if a function is continuous in any interval, then we simply call it a continuous function.

9.7.3 Properties of Definite Integrals

Certain properties of the definite integral are useful in solving problems. Some of the often used properties are given below. It is assumed throughout that $\phi'(x) = f(x)$.

(1) $\int_a^a f(x)dx = 0$

(2) $\int_a^b f(x)dx = -\int_a^a f(x)dx$

(3) $\int_a^b k\,dx = k(b-a)$ where k is constant.

(4) $k\int_a^b f(x)dx = k\int_a^b f(x)dx$

(5) $\int_a^b [f(x) \pm g(x)]dx = \int_a^b f(x)dx \pm \int_a^b g(x)dx$

(6) If $f(x) \geq 0$ on $[a, b]$, then $\int_a^b f(x)dx \geq 0$

(7) If $f(x) \leq 0$ on $[a, b]$, then $\int_a^b f(x)dx \leq 0$

(8) If $f(x) \geq g(x)$ on $[a, b]$, then $\int_a^b f(x)dx \pm \int_a^b g(x)dx$

(9) If $a < c < b$ in $[a, b]$, then $\int_a^b f(x)dx$

$= \int_a^c f(x)dx + \int_c^b f(x)dx$

(10) $\int_a^b f(x)dx = \int_a^b f(t)dt$ i.e. value of the integral is independant of the variable of integration.

(11) $\int_0^a f(x)dx = \int_0^a f(a-x)dx$

(12) $\int_a^b f(x)dx = \int_a^b f(a+b-x)dx$

(13) $\int_{-a}^{+a} f(x)dx = 2\int_0^a f(x)dx$ if 'f' is even = 0, if 'f' is odd.

(14) $\int_0^{2a} f(x)dx = \int_0^a f(x)dx + \int_0^a f(2a-x)dx$

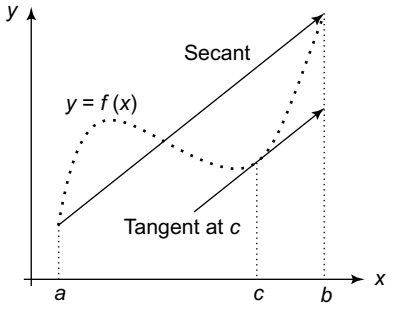
Corollary :

If $f(2a - x) = f(x)$, then $\int_0^{2a} f(x)dx = 2\int_0^a f(x)dx$

and $f(2a - x) = f(x)$, then $\int_0^{2a} f(x)dx$

9.7.4 Mean Value Theorem

Given a planar arc between two endpoints, there is at least one point at which the tangent to the arc is parallel to the secant through its endpoints.



The theorem is used to prove global statements about a function on an interval starting from local hypotheses about derivatives at points of the interval.

More precisely, if a function f **is** continuous **on the** closed interval $[a, b]$, where $a < b$, and differentiable on the open interval (a, b), then there exists a point c **in** (a, b) such that

$$f'(c) = \frac{f(b) - f(a)}{b - a}$$

This theorem is the basis for finding maxima and minima of functions.

■ **Example:** Suppose that we know that $f(x)$ is continuous and differentiable in [6, 15]. Also, that $f(6) = -2$ and that we know that $f(x) \leq 10$. What is the largest possible value for $f(15)$?

■ **Answer:** From Mean Value Theorem,

$f(15) - f(6) = f'(c)\,(15 - 6)$

Plugging in for the known quantities and rewriting this a little gives,

$$f(15) = f(6) + f'(c)\,(15 - 6) = -2 + 9f'(c)$$

Now we know that $f(x) \leq 10$ so in particular we know that $f(x) \leq 10$. This gives us the following,

$$f(15) = -2 + 9f'(c)$$

$$= -2 + 9\,(9)\,10 = 88$$

This we have achieved by replacing $f'(c)$ with its largest possible value.

This means that the largest possible value for $f(15)$ is 88.

This theorem can be also visualised like this. If y = f (x) is given a function defined on [a, b], then the mean value of the function denoted by y = f (c) is defined as y or $f(c) = \frac{1}{b-a}\int_a^b f(x)dx$ for some c ∈ (a, b)

■ **Example** Find the mean value of $y = 3x^2 + 2x$ over [0, 1]

■ **Answer: By applying mean value theorem**

Mean value $\bar{y}$ or $f(c) = \frac{1}{b-a}\int_a^b f(x)dx$

$$= \frac{1}{2-0}\int_0^2 (3x^2 + 2x)dx$$

$$= \frac{1}{2}\left[x^3 + x^2\right]_0^2 = 6$$

Now $f(c) = 3c^2 + 2c = 6 \Rightarrow 3c^2 + 2c - 6 = 0$

$$\therefore \quad c = \frac{-4 \pm \sqrt{4+72}}{6}$$

$$= \frac{-4 \pm \sqrt{76}}{6}$$

$$c = \frac{-4 \pm 2\sqrt{19}}{6}$$

$$\therefore \quad c = \frac{-2 \pm \sqrt{19}}{3}$$

Since $c = \frac{-2 \pm \sqrt{19}}{3} \approx 0.79$ is in (0, 2)

The mean value theorem is also satisfied.

■ **Example** Given that $\int_1^6 (x^3 - 1)\,dx = 318$. Find the mean value of $x^3 - 1$ and also find all '*c*' values that satisfies the mean value theorem for this function on the closed interval.

■ **Solution**

Mean value y or $f(c)$ $= \frac{1}{b-a}\int_a^b f(x)\,dx$

$$= \frac{1}{6-1}(318)$$

$$= \frac{318}{5}$$

Now $f(c) = c^3 - 1 = \frac{318}{5}$

$$\therefore \quad c^3 = \frac{318}{5} + 1 = \frac{323}{5}$$

$\therefore$ $c \approx 4$ is in the closed interval (1, 6)

$\therefore$ The mean-value theorem is satisfied.

9.7.5 Maxima and Minima Calculation

A value of x at which the function has either a maximum or a minimum is called a critical value. Both at maximum and minimum points derivative of a function f(x) known as f′(x) changes its sign; at a maximum, f′(x) changes sign from + to –, while at a minimum, f′(x) changes sign from – to + .

Theorem:

The function has a minimum value at x = a if f ′(a) = 0 and f″(a) = a positive number.

The function has a maximum value at x = a if f′(a) = 0 and f″ (a) = a negative number.

■ **Example** Calculate the critical values for the function $f(x) = x^2 - 6x + 5$ if exists any.

■ **Answer:** f′ (*x*)=2x-6.Therefore, at x=3 the given function will be having extreme value. To find whether it is maximum or minimum, we calculate f″ (x)=2. As f″ (x) value is positive, the extreme value can be said as minimum.

9.7.6 Partial Derivatives

Partial derivatives are defined as derivatives of a function of multiple variables when all but the variable of interest are held fixed during the differentiation.

$$\frac{\partial f}{\partial x_m} = \lim_{x \to \infty} \frac{f(x_1, \ldots, x_m + h, \ldots, x_n) - f(x_1, \ldots, x_m, \ldots, x_n)}{h} \tag{1}$$

The above partial derivative is sometimes denoted f_{x_m} for brevity.

Partial derivatives can also be taken with respect to multiple variables, as denoted for examples

$$\frac{\partial^2 f}{\partial x^2} = f_{x_m} \tag{2}$$

$$\frac{\partial^2 f}{\partial x \partial y} = f_{x_y} \tag{3}$$

$$\frac{\partial^2 f}{\partial x^2 \partial y} = f_{xxy}\,. \tag{4}$$

Such partial derivatives involving more than one variable are called mixed partial derivatives.

For a two-dimensional function $f(x, y)$ **(i.e., one for** which f, f_x, f_y, f_{xy}, f_{yx} **exist and are continuous in a** neighbourhood (a, b), then

$$f_{x_y}(a, b) = f_{yx}(a, b)$$

More generally, for functions, mixed partial derivatives must be equal regardless of the order in which the differentiation is performed, so it is also true that

$$f_{xxy} = f_{xyx} = f_{yxx}$$

9.7.7 Total Derivatives

The total derivative is the derivative **with respect to** t **of the function** $y = f(t, u_1, \ldots, u_m)$ **that depends on the variable** t **not only directly but also via the intermediate** variables $u_1 = u_1(t, u_1, \ldots, u_m)$, ..., $u_m = u_m(t, u_1, \ldots, u_m)$. It can be calculated using the formula

$$\frac{\partial y}{\partial t} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial u_1}\frac{\partial u_1}{\partial t} + \cdots + \frac{\partial f}{\partial u_m}\frac{\partial u_m}{\partial t}.$$

9.8 Solved Questions

1. How many numbers are there between 1 and 10000, which are either even, end in 0, or have the sum of its digits divisible by 9?

Answer: We will use inclusion/exclusion principle.

Let E, Z and N denote the numbers between 1 and 10,000 which are even, end in 0, or have the sum of their digits divisible by 9, respectively (not that N consists of all multiples of 9. Then, (including the number 10,000 in our calculations), $|E| = 5000$, $|Z| = 1000$, $|N| = 1111$, $|E \cap Z| = 1000$, $|E \cap N| = 555$, $|Z \cap N| = 111$, $|E \cap Z \cap N| = 111$. Therefore, by the Inclusion/Exclusion principle:

$|E \cup Z \cup N| = 5000 + 1000 + 1111 - 1000 - 555 - 111 + 111 = 5556$

2. Given ten points in the plane with no three collinear, how many different segments joining two points are there?

Answer: $10C_2=45$

3. Given ten points in the plane with no three collinear, how many ways are there to choose a directed path of length two through three distinct points?

Answer: We can choose a directed path of length two uniquely by choosing the starting point, the middle point, and the end point. There are $10 \cdot 9 \cdot 8 = 720$ such paths

4. Given ten points in the plane with no three collinear, how many different triangles are there?

Answer: $10C_3=120$.

5. Given ten points in the plane with no three collinear, how many ways are there to choose 4 segments?

Answer: Number of segments joining two points = $10C_2 = 45$. Number of ways of selecting four segments from 45 is = $45C_4 = 148995$.

6. Given ten points in the plane with no three collinear, if you choose 4 segments at random, what is the chance that some three form a triangle?

Answer: Number of segments joining two points = $10C_2 = 45$. Number of ways of selecting four segments from 45 is = $45C_4 = 148995$. Let us assume that first we select three vertices that will form a triangle and then we choose any one segment out of the 42. Thus, the number of ways to choose four segments that include a triangle is: 10C3*42=5040. The probability that we have picked such an arrangement when picking four segments at random is therefore=5040/148995=3.38%.

7. Forty equally skilled teams play a tournament in which every team plays every other team exactly once, and there are no ties.

a. How many different games were played?

b. How many different possible outcomes for these games are there?

c. How many different ways are there for each team to win a different number of games?

Answer:

a. $40C_2=780$ b. 2^{780} c. 40

8. How many passwords can be created in the form $[A–Z][a–z]^9[0,1]^6$? (That is, a capital letter followed by 9 lowercase letters followed by 6 bits).

Answer: $26 \cdot 26^9 \cdot 2^6$

9. How many ways are there to distribute eight balls into six distinct boxes with at least one ball in each box if:

a. Balls are identical?

b. Balls are distinct?

Answer:

(a) The balls are identical?

Because the balls are identical, we can begin by putting one ball in each box. Then, we can distribute the remaining two balls into the six boxes arbitrarily. There are $(6–1+2)C_2=21$ ways.

(b) The balls are distinct?

Since we must have at least one ball in each box, there are only two possibilities: either there are two boxes with two balls and four boxes with one ball, or there is one box with three balls and five boxes with two balls. In the former case, we must choose which two boxes have two balls ($6C_2$ ways), we must choose which balls go in each of these boxes ($8C_26C_2$ ways), and we must arrange the remaining four balls into the other four boxes (4 ways). In the latter case, we must choose which box has three balls ($6C_1$), choose which three balls go into this box ($8C_3$), and arrange the remaining five balls into the remaining five boxes (5!). The total number of arrangements is therefore:

$6C_28C_26C_24! + 6C_18C_35 = 191,520$

10. There's a new screen saver that displays a random rectangular piece of an *n* by *n* checkerboard.

a. How many rectangles are there in a checkerboard of size 1? 2? 3? 4?

b. How many squares are there in a checkerboard of size 1? 2? 3? 4?

c. Guess a general formula for the number of squares and rectangles. Put each in closed form in terms of *n*.

Answer:

(a) Size 1? 1. Size 2? 9. Size 3? 36. Size 4? 100.

(b) Size 1? 1. Size 2? 5. Size 3? 14. Size 4? 30.

(c) Number of squares in a checkerboard of size n:

$$1^2 + 2^2 + 3^2 + ... + n^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Number of rectangles in a checkerboard of size n:

$$\binom{n+1}{2}^2 = \frac{(n+1)^2 n^2}{4}$$

11. An oil tank is to be drained for cleaning. There are V gallons of oil left in the tank after t minutes of draining, where $V = 50(40 - t)^2$.

(a) What is the average rate at which oil drains out of the tank during the first 20 minutes?

Answer :

The average rate of draining AR is given by

$$AR = \frac{V(20) - V(0)}{20 - V} = \frac{50(40-20)^2 - 50(40-0)^2}{20}$$

$$= -3000$$

The average rate of drainage during the first 20 minutes is 3,000 gallons per minute.

(b) What is the rate at which oil is flowing out of the tank 20 minutes after draining begins?

Answer :

The instantaneous change in the amount of oil in the tank at time t (for 0<= t <= 40) is $\frac{dV}{dt} = -100\ (40 - t)$

At t = 20, the rate of change is –2,000 gallons per minute; 2,000 gallons per minute are flowing out of the tank.

12. What values of C the following matrix can not be LU decomposed?

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 2 & c & 7 \\ 0 & 1 & 3 \end{bmatrix}$$

Answer: If we subtract two times of first row from second row, the resulting matrix becomes:

$$\begin{bmatrix} 1 & 2 & 4 \\ 0 & c-4 & -1 \\ 0 & 1 & 3 \end{bmatrix}$$

If $c = 4$ then we cannot transform this matrix to an upper triangular one without permuting last two rows. So, there is not LU decomposition for this if $c = 4$.

13. If we have a matrix A of the following form, then its determinant is:

$$A = \begin{bmatrix} B & 0 \\ 0 & C \end{bmatrix}.$$

Answer: determinant(B)*determinant(C)

14. Explain about pivot rows(columns) and free rows(columns).

Answer: The columns that contain a leading entry are called *pivot columns*. The columns that do not contain leading entries are called *free columns*. The rows that are not entirely zeroes are pivot rows.

15. Define rank of a matrix?

Answer: The rank of a matrix A is the number of pivot columns (or rows) that it has when it is transformed into reduced row echelon form.

16. Compute the ranks of the following matrices.

$$E = \begin{bmatrix} 1 & 2 & 0 & 5 \\ 2 & 3 & 1 & 4 \\ -1 & -1 & -1 & 1 \end{bmatrix} \qquad F = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 3 & 4 \\ 2 & -1 & -3 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 3 & 1 \\ -2 & 1 & 4 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 3 \\ 2 & -1 \\ -1 & -3 \end{bmatrix}$$

Answer:

(a) Two pivots so E has rank 2

$$\begin{bmatrix} 1 & 2 & 0 & 5 \\ 2 & 3 & 1 & 4 \\ -1 & -1 & -1 & 1 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 0 & 5 \\ 0 & -1 & 1 & -6 \\ 0 & 1 & -1 & 6 \end{bmatrix}$$

$$\to \begin{bmatrix} 1 & 2 & 0 & 5 \\ 0 & -1 & 1 & -6 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) Two pivots so F has rank 2

$$\begin{bmatrix} 1 & 2 & 1 \\ -1 & 3 & 4 \\ 2 & -1 & -3 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 1 \\ 0 & 5 & 5 \\ 0 & -5 & -5 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 1 \\ 0 & 5 & 5 \\ 0 & 0 & 0 \end{bmatrix}$$

(c) Three pivots, so G has rank 3

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 3 & 1 \\ -2 & 1 & 4 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 1 \\ 0 & 3 & 1 \\ 0 & 5 & 6 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 1 \\ 0 & 3 & 1 \\ 0 & 0 & 13 \end{bmatrix}$$

(d) Two pivots, so H has rank 2

$$\begin{bmatrix} 1 & 3 \\ 2 & -1 \\ -1 & -3 \end{bmatrix} \to \begin{bmatrix} 1 & 3 \\ 0 & -7 \\ 0 & 0 \end{bmatrix}$$

17. Suppose A is a 3×3 matrix with eigenvalue 1, 2 and 3. If v_1 is an eigenvector for the eigenvalue 1, v_2 for 2, and v_3 for 3, then what is $A(v_1 + v_2 - v_3)$?

Answer:

$$A\left(\overrightarrow{v_1}+\overrightarrow{v_2}-\overrightarrow{v_3}\right)= A\overrightarrow{v_1}+A\overrightarrow{v_2}-A\overrightarrow{v_3}= 1\overrightarrow{v_1}+2\overrightarrow{v_2}-3\overrightarrow{v_3}$$

18. Explain about Row-Echelon Form

1. If a row does not consist entirely of zeros, then the first nonzero element in the row is a 1 (called a **leading 1**).
2. For any two successive nonzero rows, the leading 1 in the lower row is farther to the right than the leading 1 in the higher row.
3. All the rows consisting entirely of zeros are at the bottom of the matrix.

If a fourth property is also satisfied, a matrix is said to be in **reduced row-echelon form**:

4. Each column that contains a leading 1 has zeros everywhere else.

19. If the product of the matrices BA is defined and A is invertible, then rank (BA)= rank(B). Is it valid?

Answer : Yes.

20. Prove that the sum of the degrees of the vertices of any finite graph is even.

Answer: Each edge ends at two vertices. If we begin with just the vertices and no edges, every vertex has degree zero, so the sum of those degrees is zero, an even number. Now add edges one at a time, each of which connects one vertex to another, or connects a vertex to itself (if you allow that). Either the degree of two vertices is increased by one (for a total of two) or one vertex's degree is increased by two. In either case, the sum of the degrees is increased by two, so the sum remains even.

21. Every simple finite graph has two vertices of the same degree.

Answer: Yes

22. If G is a graph of order n, what is the maximum number of edges in G?

Answer: n(n-1)/2

23. Is it true that finite graphs having exactly two vertices of odd degree must contain a path from one to the other?

Answer: Yes

24. If graphs G and H are isomorphic, then their complements G' and H' are also isomorphic. Is it valid?

Answer: Yes.

25. For a directed graph, the absence of back edges with respect to a BFS tree implies that the graph is acyclic.

Answer: False. It is true that the absence of back edges with respect to a DFS tree implies that the graph is acyclic. However, the same is not true for a BFS tree. There may be cross edges which go from one branch of the BFS tree to a lower level of another branch of the BFS tree. It is possible to construct a cycle using such cross edges (which decrease the level) and using forward edges (which increase the level).

26. Does the depth of any DFS tree rooted at a vertex is at least as much as the depth of any BFS tree rooted at the same vertex?

Answer: Yes. Since BFS finds paths using the fewest number of edges, the BFS depth of any vertex is at least as small as the DFS depth of the same vertex. Thus, the DFS tree has a greater or equal depth.

27. Is there any edge in an undirected graph that jumps more than one level of any BFS tree of the graph?

Answer: No. If such an edge existed, it would provide a shorter path to some node than the path found by BFS (in terms in the number of edges). This cannot happen, as BFS always finds the path with the fewest edges.

28. In an unweighted graph where the distance between any two vertices is at most T, any BFS tree has depth at most T, but a DFS tree might have larger depth.

Answer: True. Since all vertices are connected by a path with at most T edges, and since BFS always finds the path with the fewest edges, the BFS tree will have depth at most T. A DFS tree may have depth up to V –1 (for example, in a complete graph).

29. If a graph contains Hamiltonian cycle then DFS traversal tree height will be V–1, where V is the number of vertexes.

Answer: Yes.

30. Suppose $\det = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ 3. Find $\det \begin{bmatrix} a & d & g \\ 2b & 2e & 2h \\ c & f & i \end{bmatrix}$.

Answer : $\det \begin{bmatrix} a & d & g \\ 2b & 2e & 2h \\ c & f & i \end{bmatrix} = 2.\det \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$

$$= 2.\det \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}^T = 2.\det \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = 6.$$

31. Consider the following matrix A. If we assume a=bor a=c, then what is its determinant? Does it have inverse if a=b or a=c?

$$A = \begin{bmatrix} 1 & a & b+c \\ 1 & b & a+c \\ 1 & c & a+b \end{bmatrix}$$

Answer: No. If a = b or a = c then A will have two identical rows and thus is row equivalent to a matrix that has a row of zeros. In such cases, det(A) is 0 and thus A is not invertible.

32. Is determinant of a matrix A is same as its transpose?

Answer: Yes

33. Multiplying a row (whole) of matrix A with a scalar x will gives a matrix whose determinant is x times of A.

Answer: Y

34. Multiplying a matrix A(of size nxn) with a scalar x will gives a matrix whose determinant is n*x times of A.

Answer: Y

35. The following operations on a matrix A is carried out such that resulting matrix U is observed to be upper triangular with its diagonal elements as 4,3, and 2. What will be the determinant of A?

1. rows r1 and r2 are swapped
2. ten times of r2 is added to row r3.
3. row 1 is multiplied with 4.

Answer: We know if rows are exchanged, determinant gets opposite sign, that is gets multiplied by -1.

Also, there will not be any change in determinant value of row operations.

If row is multiplied by x, the determinant value also gets multiplied by x.

Thus, 4x1x-1x $|A| = |U|$

$-4|A| = (4x3x2)$.

Therefore, $|A| = -6$.

36. Assume determinant of the following matrix A is 600, calculate value of w.

$$A = \begin{bmatrix} 5 & y & 2 & 4 \\ 0 & 0 & w & 0 \\ 0 & 10 & 9 & 7 \\ 0 & 0 & 6 & 3 \end{bmatrix}$$

Answer: Consider sub matrix that is formed by removing row and columns having w.

$$\begin{bmatrix} 5 & y & 4 \\ 0 & 10 & 7 \\ 0 & 0 & 3 \end{bmatrix}$$

The above matrix is upper triangular matrix. Thus, determinant of this sub matrix becomes 5x10x3=150.

Now, determinant of A= –w*determinant of sub matrix.

Therefore, w= –600/150 = –4.

37. Is there any way to find value of y given determinant of the following matrix A as 600?.

$$A = \begin{bmatrix} 5 & y & 2 & 4 \\ 0 & 0 & w & 0 \\ 0 & 10 & 9 & 7 \\ 0 & 0 & 6 & 3 \end{bmatrix}$$

Answer: No

38. If A is a matrix and determinant of AA is |A|x|A|. Is it valid?

Answer: Yes

39. If A, B are two matrices and |A| is 4 and |AB| is 8 then what is the value of |B|, $|B^T|$?

Answer: 8/4=2. Also, $|B^T|$ value is 2.

40. If A, B are two matrices and |A| is 4 and |AB| is 8 then what is the value of |A+B|?

Answer: We cannot calculate.

41. Suppose A and B are both nxn matrices, |B |= 5, and $BAB^T = I^n$. Determine the value of |A|.

Answer :

$$BAB = I_n \Rightarrow |BAB^T| = |I_n|$$
$$\Rightarrow |B|\,|A|\,|B^T| = 1$$
$$\Rightarrow (-5)|A|(-5) = 1$$
$$\Rightarrow |A| = \frac{1}{25}$$

OBJECTIVE TYPE QUESTIONS

1. Find odd one

A Multigraph with zero odd vertex is traversable

B. A multigraph with two odd vertices is traversable

C. A multigraph with more than two odd vertices are not traversable

D. None

2. Hamiltonian circuit

A. Closed path

B. Visits every vertex exactly once

C. May repeat edges

D. No. of vertices should be >=3

E. All

3. Spanning tree

A. Connected graph

B. Does not contain cycles

C. Includes all vertices of the graph

D. Subgraph of graph

E. All

4. A map
A. Planar graph B. May be multigraph
C. Is connected if underlying mutligraph is connected
D. All

5. The sum of degrees equal to double the edges is valid for
A. Graph B. Multigraph
C. Isolated graph D. All

6. Trivial graph
A. Multigraph B. A single graph
C. Finit graph D. Connected O-regular
E. All

7. A connected graph
A. Contains only one connected component
B. Contains path between any two of its vertices
C. May contain bridge
D. Will have path matrix with all 1's
E. All

8. Traversible trail (find odd one out)
A. A path without using edges twice
B. A path with out cycles
C. A path includes all vertices
D. A & C

9. A Eulierian graph
A. Contains closed traversible trail
B. Finite connected graph
C. Each vertex has even degree
D. All

10. Find odd one out
A. If a graph contains two vertices of odd degree then there must be a path joining them
B. The max degree of any vertex in a simple graph with n vertices is n-1
C. The max no. of edges in a simple graph will n vertices is $n(n-1)/2$
D. A simple graph with n vertices and k components can have atmost $(n-k)(n-k+1)/2$ edges
E. None

11. Solve the traveling salesman problem

Station	A	B	C	D	E
A	∞	13	19	16	15
B	14	∞	18	15	16
C	14	18	∞	14	13
D	13	17	16	∞	18
E	19	17	16	17	∞

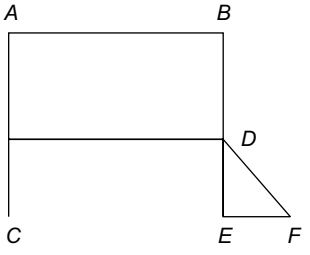
A. C→A→B→D→E
B. C→B→A→E→D
C. B→E→A→C→D
D. None

12. Find odd one regarding the following graph
A. A,B,C,D,E,F B. A,B,C,D,F,E
C. B,A,C,D,E,F D. B,A,C,D,F,E
E. None

13. Find odd one out
A. The sum of the degrees of nodes equal to twice the no. of edges
B. The no. of vertices odd degree in a undirected graph is always even
C. Trivial graph contains 1 edge
D. A & C

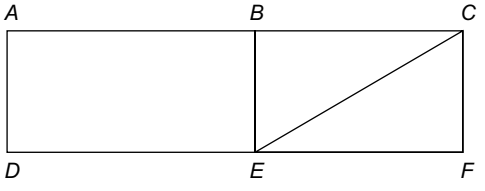
14. Biconnected graph
A. Connected
B. Undirected
C. Contains no vertices whose removal disconnects the rest of the graph
D. All

15. Articulation points in the following graph are



A. C B. A
C. D D. A & C
E. None

16. Find odd man with respect to the following graph
A. ADEBCEF B. ADECBEF
C. ABECF D. None



17. Are the following equavalent of graph G?
A. G is 2-colourable
B. G is bipartite
C. Every cycle of G has even length
D. Contains two connected components

18. A planar graph is ___ colourable.

A. 2 B. 4 C. 3 D. 5

E. None

19. If path matrix contains all 1's then graph is

A. Strongly connected

B. Unilaterally connected

C. Weakly connected

D. None

20. If a column in an adjacency matrix contains zero's then

A. Respective column in path matrix also contains all zeros

B. Respective vertex is non reachable

C. Respective vertex is a source

D. Respective column in path matrix contains all 1's.

21. The number of different ways for n people to arrange themselves in a straight line is n! (Y/N).

22. Power set P(A) will be having cardinality of $2^{|A|}$. (Y/N)

23. A graph's edges can be covered by n edge-disjoint paths, but not n-1, if and only if the graph has n pairs of odd-degree vertices. (Y/N)

24. A Hamiltonian cycle in a Hamiltonian graph of order has

A. 12 edges B. 24 edges

C. 23 edges D. None

25. A simple graph with 13 vertices out of which 4 vertices has degree 3, 3 vertices of degree 4 and 6 vertices of degree 1. The graph G must be a tree

A. True

B. False

26. A spanning tree for a simple graph of order 24 has

A. 12 edges

B. 6 edges

C. 23 edges

D. None

27. The order of a forest, F, with17 vertices and 4 components is

A. 17 B. 4 c. 16 D. None

28. The size of a forest, F, with17 vertices and 4 components is

A. 17 B. 4 C. 16 D. 13

29. The number of different labeled trees of order n is

A. n^n B. $(n-2)^n$ C. n(n–2) D. None

30. Consider a simple graph G with n vertices and n edges(n>2). Then which of the following statements are valid?

A. G has atleast one cycle.

B. G has no cycles

C. The graph obtained by removing any edge from G is not connected.

D. G has atleast one cycle and the graph obtained by removing any edge from G is not connected.

31. The number of distinct simple graphs with up to three nodes is

A. 9 B. 7 C. 10 D. 15

32. Maximum number of edges in an n-node undirected graph without self loops

A. n^2 B. n/2 C. n! D. None

33. A graph which all nodes are of equal degree is known as

A. Complete graph

B. Multi graph

C. Non regular graph

D. Regular graph

34. The minimum number of spanning trees in a connected graph with n nodes is

A. n–1 B. n/2 C. 2 D. 1

35. The minimum number of edges in a connected cyclic graph with n vertices is

A. n–1 B. n C. n+1 D. 2

36. Every directed acyclic graph has exactly one topological ordering. (Y/N)

37. If a directed graph G is cyclic but can be made acyclic by removing one edge, then a depth-first search in G will encounter exactly one back edge. (Y/N)

38. If all edges in a graph have distinct weights, then the shortest path between two vertices is unique(Y/N).

39. On matrix A, the following elementary row operations are applied one after another.

$R2:=R2-R1$, $R2\leftrightarrow R3$(row exchange), $R3:=R3-3R2$,

$$\begin{bmatrix} 1 & -1 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 6 \end{bmatrix}$$

What is the determinant of A after the operations?

A. 6 B. 16 C. –6 D. None

Answer: Original determinant of A=6. Because of simple row operations (1st and last), determinant will not change. However, because of the row exchange, determinant gets multiplied by -1. Thus, determinant at the end becomes –6.

40. The matrix A is given below. The following elementary row operations are applied on it in sequence.

$R2:=R2-3R1$,$R3:=R3+4R1$,$R2\leftrightarrow R3$,$R2:=12R2$,$R4:=R4-2R2$,

What is the determinant of A after these row operations?

$$\begin{bmatrix} 2 & 1 & 0 & 3 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A. −3 B. 3 C. 12 D. None

41. After the following elementary row operations in the sequence, a matrix A became the following matrix.

$R2:=R2-3R1, R3:=R3+4R1, R2\leftrightarrow R3, R2:=12R2, R4:=R4-2R2,$

What is the determinant of *A(original matrix)?*

$$\begin{bmatrix} 2 & 1 & 0 & 3 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A. −3 B. 3 C. 12 D. 12

42. Find correct statement out of the following.

A. $\det A^{-1} = -\det A$

B. $\det A^{-1} = -\dfrac{1}{\det A}$

C. $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$

D. $\begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix} = 2\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

Answer: C. Determinant of a matrix and its transpose are same.

ANSWER KEY

1. C	**2.** E	**3.** E	**4.** D
5. D	**6.** E	**7.** E	**8.** B
9. D	**10.** E	**11.** A	**12.** E
13. C	**14.** D	**15.** D	**16.** C
17. D	**18.** D	**19.** A	**20.** D
21. Y	**22.** Y	**23.** Y	**24.** B
25. False	**26.** C	**27.** A	**28.** D
29. C	**30.** A	**31.** D	**32.** D
33. D	**34.** D	**35.** B	**36.** N
37. N	**38.** N	**39.** C	**40.** A
41. A&D	**42.** C		

Previous Years' GATE Questions

1. A binary operation $\overset{\square}{\oplus}\square$ on a set of integers is defined as $\overset{\square}{\oplus}\square\, xy=x^2+y^2$. Which one of the following statements is TRUE about $\overset{\square}{\oplus}\square$? **(GATE 2013)**

A. Commutative but not associative

B. Both commutative and associative

C. Associative but not commutative

D. Neither commutative not associative

2. Suppose p is the number of cars per minute passing through a certain road junction between 5PM and 7PM, and p has the poisson distribution with mean 3. What is the probability of observing fewer than three cars during any given minute in this interval? **(GATE 2013)**

A. $8/(2e^3)$ B. $9/(2e^3)$

C. $17/(2e^3)$ D. $26/(2e^3)$

Explanation:

X is a Poisson variable with pdf:

$$P(X=x) = e^{-\lambda}\frac{\lambda^x}{x!},\ x = 0, 1, ..., \infty$$

Here, λ is given as 3. We need to calculate probability of observing fewer than 3 cars. That is, we need to calculate $P(0)+P(1)+P(2)=e^{-3}+e^{-3}\ 3+e^{-3}3^2/2! = e^{-3}(1+3+3/2) =17/(2\ e^3)$

3. Which one of the following is NOT equal to

$$\begin{vmatrix} 1 & x & \boxed{x^2} \\ 1 & y & \boxed{y^2} \\ 1 & z & \boxed{z^2} \end{vmatrix}$$ **(GATE 2013)**

A. $\begin{vmatrix} 1 & x(x+1) & x+1 \\ 1 & y(y+1) & y+1 \\ 1 & z(z+1) & z+1 \end{vmatrix}$ B. $\begin{vmatrix} 1 & x+1 & x^2+1 \\ 1 & y+1 & y^2+1 \\ 1 & z+1 & z^2+1 \end{vmatrix}$

C. $\begin{vmatrix} 0 & x-y & x^2-y^2 \\ 0 & y-z & y^2-z^2 \\ 1 & z & z^2 \end{vmatrix}$ D. $\begin{vmatrix} 2 & x+y & x^2+y^2 \\ 2 & y+z & y^2+z^2 \\ 1 & z & z^2 \end{vmatrix}$

4. Which one of the following statements is true? **(GATE 2013)**

1 The problem of determining whether there exists a cycle in an undirected graph is in P.

2 The problem of determining whether there exists a cycle in an undirected graph is in NP.

3 If a problem is NP-Complete, there exists a non-deterministic polynomial time algorithm to solve A.

A. 1,2 and 3 B. 1 and 2 only

C. 2 and 3 only D. 1 and 3 only

5. Which one of the following functions is continuous at x=3? **(GATE 2013)**

A. $f(x) = \begin{cases} 2, & \text{if } x = 3 \\ x-1, & \text{if } x > 3 \\ \frac{x+3}{3}, & \text{if } x < 3 \end{cases}$

B. $f(x) = \begin{cases} 4, & \text{if } x = 3 \\ 8-x, & \text{if } x \neq 3 \end{cases}$

C. $f(x) = \begin{cases} x+3 & \text{if } x \leq 3 \\ x-4, & \text{if } x > 3 \end{cases}$

D. $f(x) = \frac{1}{x^3 - 27}$, if x ≠ 3

6. The function f is known at the following points. **(GATE 2013)**

	x	$f(x)$
Y_0	0	0
Y_1	0.3	0.09
Y_2	0.6	0.36
Y_3	0.9	0.81
Y_4	1.2	1.44
Y_5	1.5	2.25
Y_6	1.8	3.24
Y_7	2.1	4.41
Y_8	2.4	5.76
Y_9	2.7	7.29
Y_{10}	3.0	9.00

The value of $\int_0^x x dx$ computed using trapezoidal rule is

A. 8.983 B. 9.003

C. 9.017 D. 9.045

Explanation: h value = 0.3. Therefore, area = 0.3/2(0 + 9 + 2(0.09 + 0.36 + 0.81 + 1.44 + 2.25 + 3.24 + 4.41 + 5.76 + 7.29)] = 9.045

7. Consider an undirected random graph of eight vertices. The probability that there is an edge between two vertices is 1/2, what is the expected number of unordered cycles of length three? **(GATE 2013)**

A. 1/8 B. 1 C. 7 D. 8

Explanation: We can have total 8C3 ways to have a unordered cycle of three vertices. The probability that there is an edge between two vertices is 1/2. So, expected number of unordered cycles of length three = $(8C3)*(1/2)^3 = 7$

8. Which of the following statements is/are TRUE for undirected graphs? **(GATE 2013)**

P: Number of odd degree vertices is even.

Q: Sum of degrees of all vertices is even

A. P only B. Q only

C. Both P and Q D. Neither P nor Q

9. What is the logical translation of the following statement? **(GATE 2013)**

"None of my friends are perfect".

A. $\exists x\,(F(x) \wedge \neg P(x))$ B. $\exists x\,(\neg F(x) \wedge P(x))$

C. $\exists x\,(\neg F(x) \wedge \neg P(x))$ D. $\neg \exists x\,(F(x) \wedge P(x))$

Answer: D

10. Which one of the following is logical equivalent to $\neg \exists x\,(\forall y\,(\alpha) \wedge \forall z\,(\beta))$? **(GATE 2013)**

A. $\forall x\,(\exists z\ (\neg \beta) \rightarrow \forall y\,(\alpha))$

B. $\forall x\,(\exists z\ (\beta) \rightarrow \exists y\,(\neg \alpha))$

C. $\forall x\,(\forall y\ (\alpha) \rightarrow \exists z\,(\neg\beta))$

D. $\forall x\,(\exists y\,(\neg \alpha) \rightarrow \exists z\,(\neg\beta))$

Answer: A, D

11. Consider the following logical inferences. **(GATE 2012)**

1. If it rains then the cricket match will not be played.
   The cricket match was played.
   Inference: There was no rain.
2. If it rains then the cricket match will not be played.
   It did not rain.
   Inference: The cricket match was played.

Which of the following is TRUE?

A. Both 1 and 2 are correct inferences

B. 1 is correct but 2 is not a correct inference

C. 1 is not correct but 2 is a correct inference

D. Both 1 and 2 are not correct inferences

12. Consider the function f(x) = sin(x) in the interval $x \in [\pi/4, 7\pi/4]$. The number and location(s) of the local minima of this function are **(GATE 2012)**

(A) One, at π/2

(B) One, at 3π/2

(C) Two, at π/2 and 3π/2

(D) Two, at π/4 and 3π/2

Explanation: We can verify by calculating second derivative, f"(x) at the given points.

13. Let A be the 2 × 2 matrix with elements a11 = a12 = a21 = +1 and a22 = −1. Then the eigenvalues of the matrix A^{19} are **(GATE 2012)**

(A) 1024 and −1024

(B) $1024\sqrt{2}$ and $-1024\sqrt{2}$

(C) $4\sqrt{2}$ and $-4\sqrt{2}$

(D) $512\sqrt{2}$ and $-512\sqrt{2}$

Explanation: Given matrix is: $\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$. To calculate eigenvalues, $\begin{vmatrix}\lambda-1 & -1\\ -1 & \lambda+1\end{vmatrix}=0$. That is, eigenvalues becomes $\sqrt{2}$ and $-\sqrt{2}$. Therefore, eigenvalues of A19 becomes $(\sqrt{2})^{19}$ and $(\sqrt{2})^{19}$. Thus, D is the valid option.

14. What is the correct translation of the following statement into mathematical logic?

"Some real numbers are rational" **(GATE 2012)**

A. $\exists x(\text{real}(x) \vee \text{rational}(x))$

B. $\forall x(\text{real}(x) \rightarrow \text{rational}(x))$

C. $\exists x(\text{real}(x) \wedge \text{rational}(x))$

D. $\exists x(\text{rational}(x) \rightarrow \text{real}(x))$

15. Let G be a simple undirected planar graph on 10 vertices with 15 edges. If G is a connected graph,then the number of bounded faces in any embedding of G on the plane is equal to **(GATE 2012)**

(A) 3 (B) 4 (C) 5 (D) 6

16. Consider a random variable X that takes values +1 and −1 with probability 0.5 each. The values of the cumulative distribution function F(x) at x = −1 and +1 are **(GATE 2012)**

A. 0 and 0.5

B. 0 and 1

C. 0.5 and 1

D. 0.25 and 0.75

Explanation: Only two possibilities with equal probability of 0.5. Therefore, cdf becomes 0.5 and 1.

17. Which of the following problems are decidable? **(GATE 2013)**

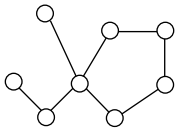
1. Does a given program ever produce an output?
2. If L is a context-free language then, is $\overline{L}$ also context-free?
3. If L is a regular-free language then, is $\overline{L}$ also regular?
4. If L is a recursive language then, is $\overline{L}$ also recursive?

(A) 1, 2, 3, 4 (B) 1, 2

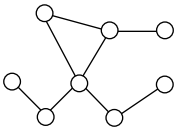
(C) 2, 3, 4 (D) 3, 4

18. Which of the following graphs is isomorphic to **(GATE 2012)**

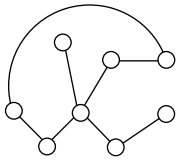
A.



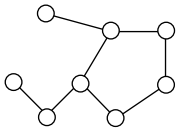
B.

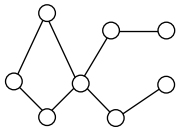


C.



D.





19. The bisection method is applied to compute a zero of the function $f(x) = x^4 - x^3 - x^2 - 4$ in the interval [1,9]. The method converges to a solution after ––––– iterations. **(GATE 2012)**

A. 1 B. 3

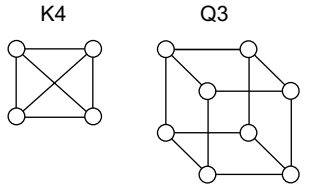
C. 5 D. 7

20. How many onto (or surjective) functions are there from an n-element ($n \geq 2$) set to a 2-element set? **(GATE 2012)**

A. 2^n B. $2^n - 1$

C. $2^n - 2$ D. $2(2^n - 2)$

21. K4 and Q3 are graphs with the following structures. **(GATE 2011)**



Which one of the following statements is True in relation to these graphs?

A. K4 is planar while Q3 is not

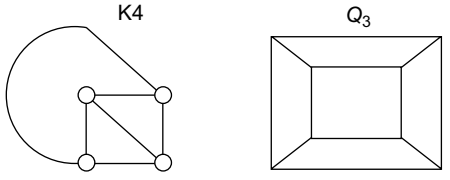
B. Both K4 and Q3 are planar

C. Q3 is planar while K4 is not

D. Neither K4 nor Q3 is planar

Explanation:

A graph can be said as planar if the nodes of the graph can be re-arranged (without breaking or adding the edges) such that no edge of the graph cross each other. Given graphs also can be drawn in such mannar. Thus, both are planar.



22. If the difference between the expectation of the square of random variable(E $[X^2]$)and the square of the expectation of the random variable (E$[X^2]$) is denoted by R then **(GATE 2011)**

A. R = 0 B. R<0

C. R ≥ 0 D. R>0

Explanation:

Assuming x1, x2,……..xn, then expectation of the series = mean=(x1+x2+….+xn)/n

Expectation of squares of the values=($x1^2$+$x2^2$+…… xn^2)/n

R= ($x1^2$+$x2^2$+……xn^2)/n - ((x1+x2+….+xn)/n)^2

= ($x1^2$+$x2^2$+……xn^2)/n - (($x1^2$+$x2^2$+….+xn^2 + 2x1x2+2x2x3+…..+xn-1xn)/n^2)

=($x1^2$+$x2^2$+……xn^2)/n*(1–1/n) - (2x1x2+2x2x3+…..+xn–1xn)/n^2

We can verify that the above value will be >=0.

23. Consider the matrix as given below. **(GATE 2011)**

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 7 \\ 0 & 0 & 3 \end{bmatrix}$$

Which one of the following provides the correct values of eigenvalues of the matrix?

A. 1,4,3 B. 3,7,3

C. 7,3,2 D. 1,2,3

Explanation: Given matrix is upper triangular matrix and thus its diagonal elements are its eigenvalues.

24. Given i = $\sqrt{-1}$, what will be the evaluation of the definite integral $\int_0^{\pi/2} \frac{\cos x + i \sin x}{\cos x - i \sin x} dx$? **(GATE 2011)**

A. 0 B. 2 C. –1 D. i

Explanation: $\int_0^{\pi/2} \frac{e^{ix}}{e^{-ix}} dx = \int_0^{\pi/2} e^{2ix}\, dx$

$$= \left(\frac{e^{2ix}}{2i}\right)_0^{\pi/2} = \frac{1}{2i}\left[e^{i\pi} - 1\right]$$

$$= \frac{1}{2i}[\cos\pi - i\sin\pi - 1]$$

$$= \frac{1}{2i}[-1+0-1] = \frac{-2}{2i} = \frac{-1}{i} \times \frac{i}{i} = \frac{-i}{-1} = i$$

25. Consider a finite sequence of random values X = [x1,x2,...xn]. Let μ_x be the mean and σx be the standard deviation of X. Let another finite sequence Y of equal length be derived from this as $y_i = a^* x_i + b$, where a and b are positive constants. Let μ_y be the mean and σ_y be the standard deviation of this sequence. Which one of the following statements is Incorrect? **(GATE 2011)**

A. Index position of mode of X in X is the same as the index position of mode of Y in Y.

B. Index position of median of X in X is the same as the index position of median of Y in Y.

C. $\mu_y = a\mu_x + b$

D. σ_y s = $a\sigma_x + b$

Explanation:

Assuming x1,x2,……xn are the values.

μ_x =(x1+x2+….+xn)/n

μ_y =(a*x1+b + a*x2+b + …..+ a*xn+b)/n

= a*(x1+x2+…+xn)/n +b = a μ_x + b

Similarly,

σx = sqrt(($x1^2$+$x2^2$+….xn^2)/n – μ_x^2)

Similary,

σ_y= sqrt(($(a^*x1+b)^2$+$(a^*x2+b)^2$+….+$(a^*xn+b)^2$)/n–$\mu_y 2$)

If one verifies the above equation, we can find that option D is not valid.

26. Let G=(V, E) be a graph. Define ξ (G) = $\sum_d id \times d$, where id is the number of vertices of degree d in G. If

S and T are two different trees with ξ (S) = ξ (T), then **(GATE 2010)**

A. |S| = 2 |T| B. |S| = |T| –1

C. |S| = |T| D. |S| = |T| + 1

27. Newton-Raphson method is used to compute a root of the equation $x^2 - 13 = 0$ with 3.5 as the initial value. The approximation after one iteration is **(GATE 2010)**

A. 3.575 B. 3.676

C. 3.667 D. 3.607

Explanation: $f(3.5) = 3.5^2 - 13 = -0.75$

$f'(3.5) = 2*3.5 = 7$

Therefore, $x_{pred} = 3.5 - \frac{-0.75}{7} = 3.607$

As, 3.5 is different from 3.607, initial value will be taken as this.

28. What is the possible number of reflexive relations on a set of 5 elements? **(GATE 2010)**

A. 2^{10} B. 2^{15}

C. 2^{20} D. 2^{25}

Explanation: The total no. of reflexive relations on a set A having n elements is $2^{n(n-1)}$. Thus, the required no. is 2^{20}.

29. Consider the set S = {1, ω, ω2}, where ω and ω2 are cube roots of unity. If *denotes the multiplication operation, the structure (S, *) forms **(GATE 2010)**

A. A group

B. A ring

C. An integral domain

D. A field

30. Consider the following matrix **(GATE 2010)**

$$A = \begin{bmatrix} 2 & 3 \\ x & y \end{bmatrix}$$

If the eigenvalues of A are 4 and 8, then

A. x = 4, y = 10

B. x = 5, y = 8

C. x = –3, y = 9

D. x = –4, y = 10

Explanation: We know trace of a matrix is same as the sum of the eigenvalues. Thus,

2+y=12

Therefore, y=10.

Similarly, product of eigen values will be same as the determinant. That is,

2y – 3x = 4*8

20 – 3x = 32

x = –4

Thus, option D is valid.

31. Suppose the predicate F(x, y, t) is used to represent the statement that person x can fool person y at time t. which one of the statements below expresses best the meaning of the formula $\forall x \exists y \exists t \neg F(x, y, t)$? **(GATE 2009)**

A. Everyone can fool some person at some time

B. No one can fool everyone all the time

C. Everyone cannot fool some all the time

D. No one can fool person at same time

32. Which one of the following in NOT necessarily a property of a Group? **(GATE 2009)**

A. Commutativity

B. Associativity

C. Existence of inverse for every element

D. Existence of identity

Explanation: It follows directly from the definition of group.

33. What is the chromatic number of an n-vertex simple connected graph which does not contain any odd length cycle? Assume n>=2. **(GATE 2009)**

(A) 2 (B) 3

(C) n–1 (D) n

Explanation: This is directly verified from the theory.

34. Which one of the following is TRUE for any simple connected undirected graph with more than 2 vertices? **(GATE 2009)**

A. No two vertices have the same degree.

B. At least two vertices have the same degree.

C. At least three vertices have the same degree.

D. All vertices have the same degree.

35. Consider the binary relation R = {(x,y), (x,z), (z,x), (z,y)} on the set {x,y,z}. **(GATE 2009)**

Which one of the following is true?

A. R is symmetric but not antisymmetric

B. R is not symmetric but antisymmetric

C. R is both symmetric and antisymmetric

D. R is neither symmetric nor antisymmetric

36. The binary operation T is defined as follows **(GATE 2009)**

P	Q	$P \square Q$
T	T	T
T	F	T
F	T	T
F	F	F

Which one of the following is equivalent to P v Q ?

A. $\neg Q \square \neg P$ B. $P \square \neg Q$

C. $\neg P \square Q$ D $\neg P \square \neg Q$

Explanation: See the following table.

P	Q	P ν Q	¬Q	P□¬Q
T	T	T	F	T
T	F	T	T	T
F	T	T	F	T
F	F	F	T	F

37. $\int_0^{\pi/4} (1-\tan x)/(1+\tan x)dx$ evaluates to **(GATE 2009)**

A. 0 B. 1

C. ln 2 D. $\frac{1}{2}$ ln 2

38. Consider the following well-formed formulae: **(GATE 2009)**

I. ¬∀x (P (x)) II. ¬∃x (P (x))

III. ¬∃x (¬P (x)) IV. ¬∃x (¬P (x))

Which of the above are equivalent?

A. I and III B. I and IV

C. II and III D. II and IV

39. Let X, Y, Z be sets of sizes x, y and z respectively. Let W = X × Y and E be the set of all subsets of W. The number of functions form Z to E is : **(GATE 2006)**

A. Z^{2xy} B. $Z \times 2^{xy}$

C. Z^{2x+y} D. 2^{xyz}

40. A relation R is defined on ordered pairs of integers as follows :

(x, y) R (u, v) if $x < u$ and $y > v$. Then R is **(GATE 2006)**

A. Neither a Partial Order nor an Equivalence Relation

B. A Partial Order but not a Total Order

C. A Total Order

D. An Equivalence Relation

41. We are given a set $X = \{x_1,.....,x_n\}$ where $x_i = 2^i$. A sample $S \subseteq X$ is drawn by selecting each xi independently with probability $p^i = \frac{1}{2}$. The expected value of the smallest number in sample S is : **(GATE 2006)**

A. $\frac{1}{n}$ B. 2

C. $\sqrt{n}$ D. n

ANSWER KEY

1. A	**2.** C	**3.** A	**4.** A
5. A	**6.** D	**7.** C	**8.** C
9. D	**10.** A & D	**11.** B	**12.** D
13. D	**14.** C	**15.** D	**16.** C
17. D	**18.** B	**19.** C	**20.** C
21. B	**22.** C	**23.** A	**24.** D
25. D	**26.** C	**27.** D	**28.** C
29. A	**30.** D	**31.** D	**32.** A
33. A	**34.** B	**35.** D	**36.** B
37. D	**38.** Incomplete question		**39.** D
40. A	**41.** B		

CHAPTER TEN 10

Verbal Ability and Numerical Reasoning

10.1 Verbal Ability (English Language Tips)

Noun: A noun is a word which is used to name items, things, ideas, people, places, entities etc.

- Countable nouns (have singular and plural forms): e.g. thoughts, children, etc.
- Uncountable nouns (do not have different singular and plural forms): e.g. information, equipment, etc.
- Either countable or uncountable for different purposes: e.g. university, climate, communication

Nouns usually come with **'determiner'** words **(articles)** which help to specify which particular items are being referred to:

- **'a'** (indefinite): all, some, any, another, each, every, either, neither, no, etc.
- **'the'** (definite): this, that, these, those, my, your, his, her, its, out, their, etc.

There can be only one determiner in a noun group.

- **predeterminers :**
  - all, both, half, double, treble, quadruple, twice, (before definite determiners)
  - such and what (before indefinite determiners)

Articles

Type of noun	Countable	Type of article	Uncountable	Type of article
singular	a/an chair the	indefinite definite (collective)	some money the	

Type of noun	Countable		Type of article	Uncountable		Type of article
Concrete						
plural	some chairs the		indefinite generalised definite	No plural (but 'bags of money')		
Singular	a/an idea the		indefinite definite (collective)	some clarity the		indefinite generalised definite
Abstract						
plural	some ideas the		indefinite generalised definite		No plural (but degrees of clarity)	

Tips

- Use **'the'** when the reader already knows what is referred to, or if an explanation comes immediately. It can also be used in a collective sense—for example, 'the chair' represents all chairs.
- Use **'a/an'** only with countable nouns.
- If it's countable and singular, it must use either 'a/an' or 'the'. Deliberate omission of any article is only possible when the noun is being generalised—for example, 'chairs' (in general) or 'money' (in general).

Singular	**Countable**
indefinite	A bucket of water was left on the road.
definite	The bucket of water which was left on the road could have been a hazard.
Plural	
generalised	Buckets of water left on the road are usually hazardous
indefinite	Some buckets of water were left on the road
discriminating definite	The buckets of water which prove most hazardous are the ones that are left on the road.
Singular	**Uncountable**
Indefinite	Some clarity in the argument would be welcomed.
generalised	Clarity in argument is very desirable.
definite	The clarity of your argument is impressive.

Adjectives

Adjectives often accompany nouns, giving further descriptions of the item:
Single word adjective: green, hot, difficult, random etc.
Adjective *phrase:* (man) with no name, (sale) of the century
Adjective *clause:* which was left on the table, that I held in my hand

Pronouns

Pronouns are used to represent items and to avoid repetition of nouns:

	Personal		**Objective possessive**	**Reflexive determiner**		
Singular	I		me	mine	myself	my
	You	You	yours	yourself	your	
	he/she/it	him/her/it	his/hers	himself/ herself/ itself	his/her/its	
Plural	we	us	ours	ourselves	our	
	you	you	yours	yourselves	your	
	they	them	theirs	themselves	their	

Pronouns for reference

	Personal pronouns		**Object pronouns**		**Possessive pronouns**		**Reflexive pronouns**		**Possessive determinants**	
	Singular	***Plural***	***Singular***	***Plural***	***Singular***	***Plural***	***Singular***	***Plural***	***Singular***	***Plural***
1st person	I	we	me	us	mine	ours	myself	ourselves	my	our
2nd person	you	you	you	you	yours	yours	yourself	yourselves	your	
3rd person	he, she, it	they	him, her, it	theirs	his, hers	theirs	himself, herself, itself	themselves	his, her, its	their

Verbs: to act, to be, to do, to have

A verb represents an action or state; it also shows who or what is associated with that action and when it occurred.

who/what *noun subject*		**did what to** *verb = action/state*	**whom/what** *noun/adjective = object*
Active	the cat	bit	the dog
Passive	the dog	was bitten by	the cat

The important thing for good sentence construction is to **know** which noun did which verb, and when. This gives you the correct agreement and tense.

Singular–plural agreement

It is absolutely vital that the number of subject(s) matches the correct form of the verb:

- Blue and green **are** colours.
- It seem**s** that Smith et al.(1988) **were** aware of the fact.
- The equipment **was** prepared.

Subject–verb partnership

Each statement is built around the subject/verb partnership. A sentence has various features according to the main verb, and the associated noun, which can determine the best word order and syntax for the sentence.

Verb forms

Each verb has three versions for use in forming tenses:

infinitive	past simple	past participle
to be	I was	I have been
to look	I looked	I have looked
to forget	I forgot	I have forgotten
to cut	I cut	I have cut

Verb tenses

simple present	I am
present perfect	I have been
simple past	I was
simple future	I will be
future perfect	I will have been

Conditionality

Conditional			
Active	I would have done if ...	I would do it ... I would be doing if ...	I would have been doing if ...
passive	I would have heard if ... I would have been	I would be heard if ...	

Verb control

Aim to practice and familiarity with a variety of verb tenses.

Verb tenses

Verb	*Past perfect*	*Simple past*	*Present perfect*	*Simple present*	*Future perfect*	*Simple future*
To do	I had done	**I did**	**I have done**	**I do**	I will have done	**I will do**
	I had been doing	**I was doing**	I have been doing	**I am doing**	I will have been doing	**I will be doing**
		I did do		**I do**		**I am going to do**
		I used to do				
To see	I had seen	**I saw**	I have seen	**I see**	I will have seen	**I will see**
	I had been seeing	**I was seeing**	I have been seeing	**I am seeing**	I will have been seeing	**I will be seeing**
		I did see		**I do see**		**I am going to see**
		I used to see				
To have	I had had	**I had**	I have had	**I have**	I will have had	**I will have**
	I had been having	**I was having**	I have been having	**I am having**	I will have been having	**I will be having**
		I did have		**I do have**		**I am going to have**
		I used to have				
To go	I had gone	**I went**	I have gone	**I go**		
		I did go	I have been going	**I am going**	I will have gone	**I will go**
	I had been going	**I was going**		**I do go**	I will have been going	**I will be going**
		I used to go				**I am going to go**

Verb	*Past perfect*	*Simple past*	*Present perfect*	*Simple present*	*Future perfect*	*Simple future*
To plan it	I had planned it I had been planning it	**I planned it** **I did plan it** **I was planning it** **I used to plan it**	I have planned it I have been planning it	**I plan it** **I am planning it** **I do plan it**	I will have planned it I will have been planning it	**I will plan it** **I will be planning it** **I am going to plan it**
To be heard	I had been heard	**I was heard** **I was being heard** **I used to be heard**	I have been heard	**I am heard** **I am being heard**	I will have been heard	**I will be heard** **I am going to be heard**

Adverbs

Adverbs often accompany verbs, giving further information about the action or state (e.g. when, where, how):

Single word adverb: wisely, overnight, adequately, seldom

Adverb phrase: (he spoke) with difficulty,
(she drove me) round the bend

Adverb clause: (call me) when you get there
(you will see) when the time comes

10.1.1 Vocabulary Builder

Word	Meaning and sample usage
Abase	Behave in a way that causes others to think less of one E.g.: He abased himself before the king.
Abate	(of something bad)Become less severe or widespread E.g.: The doctor gave him some medicine to abate the pain which he was suffering from morning.
Abdicate	1. Give up the role of king or queen 2. Fail to carry out (a duty) E.g.: Our king is abdicated by his brother.
Aberrant	Not normal or acceptable E.g.: *Aberrant* behaviour can be a sign of rabies in a wild animal
Abeyance	Temporarily not occurring or not in use E.g.: The imposition of the new tax has been held in abeyance, as the government wants some more time to study its consequences.
Abjure	Swear to give up (a belief or a claim) E.g.: For nearly 21 years after his resignation as Prime Minister in 1963, he abjured all titles, preferring to remain just plain 'Mr.' (Time).

Word	Meaning and sample usage
Abscond	Leave quickly and secretly to escape from custody or avoid arrest E.g.: A rigorous and wide search is under way for the manager who absconded with company funds.
Abstain	Stop oneself from(doing something enjoyable) E.g.: Members were also instructed to completely abstain from all drugs.
Abstemious	Taking care to limit one's intake of food or alcohol E.g.: A Catholic Nun lives a very abstemious lifestyle.
Abyss	A very deep hole E.g.: He fell into the dark abyss.
Accretion	Growth or increase by a gradual build-up of layers E.g.: The phenomenon of "coastal squeeze" of saltmarsh may be causing some net vertical accretion of mudflat surfaces.
Acidulous	1. Sharp-tasting or sour 2. (of person's remarks)bitter or sour E.g.: The drama critic made some acidulous comments on the play.
Acme	The highest point of achievement or excellence E.g.: A baseball player usually reaches the acme of his skill before he is thirty.

Word	Meaning and sample usage
Acquiesce	To comply or submit E.g.: Though I wasn't enthusiastic about Tom's plan to go fishing, I acquiesced in it because there seemed nothing else to do.
Acrid	Unpleasantly bitter or sharp E.g.: The acrid odor of burnt gun powder filled the air after the pistol had been fired.
Acrimonious	Bitter and sharp in language or tone; rancorous E.g.: I am watching an acrimonious debate between the two gubernatorial candidates.
Acute	Keenly perceptive; sharp or severe E.g.: I had an acute pain in my knees
Adamant	Refusing to be persuaded or to change one's mind E.g.: The student was adamant in his decision to pursue a career in music.
Adequate	Satisfactory or acceptable E.g.: The student who arrived ten minutes late did not have adequate time to finish the test.
Adulate	Excessive admiration E.g.: The singer raised his arms and basked in the adulation of the audience.
Adulterate	Make poorer in quality by adding another substance E.g.: Hospitals take strict precautions to assure that's nothing adulterates the blood supply.
Advocate	Publicly recommend or support E.g.: Health authorities continue to advocate immunisation for children.
Aerie	A large nest of a bird of prey, esp. an eagle, typically built high in a tree or on a cliff. E.g.: There in the tree an aerie is approached up there.
Aesthetic	1. Concerned with beauty or appreciation of beauty 2. Having a pleasant appearance E.g.: Marjorie, having studied modernist painters, appreciated the aesthetic of the painting.
Affected	False and designed to impress E.g.: He was affected with the view that he stopped and took out his camera.
Aggrandize	Increase the power, status or wealth of E.g.: Julius Caesar worked hard during his period of ruling to aggrandize his empire.
Aggregate	Formed or calculated by combining many separate items E.g.:The pore of the allies' aggregated together was great, though individually some were quite weak.

Word	Meaning and sample usage
Alacrity	Brisk eagerness or enthusiasm E.g.: The elves in Santa's workshop moved about with obvious alacrity.
Alleviate	Make (pain or difficulty) less severe E.g.: I took an advil to alleviate my back pain.
Amalgamate	Combine to form one organisation or structure E.g.: In order to create a single entity from two parts you must amalgamate them.
Ambiguous	1. Having more than meaning 2. Not clear or decided E.g.: The results of the experiment were ambiguous, and no conclusions could be made.
Ambivalence	Having mixed feelings about something or someone E.g.: The girl was feeling some ambivalence when she had to choose between two very close answers.
Ameliorate	Make(something bad or unsatisfactory) better E.g.: By expanding our non polluting energy choices, we can ameliorate a variety of risks.
Amortize	Gradually pay off(a debt) E.g.: In order to pay off his large debts, John is going to amortize the payments.
Amulet	An ornament or small piece worn as protection against evil E.g.: Charlie the unicorn and his friends found the amulet of wonder and took it to the banana king.
Anachronism	A thing belonging to a period other than the one in which it exists E.g.: It is an anachronism to say that William Shakespeare typed his manuscripts.
Analgesia	The inability to feel pain E.g.: If you have severe pain you may be prescribed stronger analgesics such as codeine.
Analogous	Alike or comparable in some ways E.g.: The attack on the world trade center was exactly analogous to pearl harbor.
Anodyne	1. Unlikely to cause offence or disagreement; bland 2. A pain killing drug or medicine E.g.: An aspirin is an anodyne for a headache.
Anomalous	Differing from what is standard or normal E.g.: Sleep walking is an anomalous behavior who's causes are not well understood
Anomaly	Something that departs what is standard or normal E.g.: A bird that cannot fly is an anomaly.

Word	Meaning and sample usage
Antagonize	Make(someone) hostile E.g.: No candidate will want to antagonize organised labour in an election campaign.
Antipathy	A strong feeling of dislike E.g.: A guardian leader warns of growing antipathy in politics.
Apathy	Lack of interest or enthusiasm E.g.: Only an apathy person can see suffering without trying to relieve it.
Aperture	An opening, hole or gap E.g.: The audience enter a booth one at a time to watch a three minute performance through a small rectangular aperture.
Apocryphal	Widely circulated but unlikely to be true E.g.: She told an apocryphal story about the pendant, but later the truth was discovered.
Apostate	A person who abandons a belief or a principle E.g.: Julian the Emperor of the Eastern Roman Empire was titled 'the apostate' as he renounced Christianity and attempted to revitalise Paganism.
Approbation	Approval E.g.: I was cheerfully received, and rejoiced that I had merited the approbation of so many worthy individuals.
Appropriate	1. Take for one's own use without permission 2. Set aside (money) for special purpose E.g.: I always appropriate my income so as to meet emergency health problems.
Arbitrary	Based on random choice or impulse
Arbitrate	Act as an arbitrator to settle a dispute
Archaic	Very old or old-fashioned Belonging to former or ancient times
Ardor	Enthusiasm or passion
Arduous	Difficult and tiring
Arrogance	Having too great a sense of one's own importance or abilities
Arrogate	Take or claim for oneself without justification
Articulate	Fluent and clear in speech E.g.: A baby cries and gurgles but does not use articulate speech (or speak distinctly).
Ascetic	Strictly self-disciplined and avoiding any pleasures or luxuries
Assail	Attack violently E.g.: The senators assailed the President on the subject of the treaty between the two countries.
Assiduous	Showing great care and thoroughness

Word	Meaning and sample usage
Assuage	Make(an unpleasant feeling) less intense
Attenuate	Make weaker
Audacious	Willing to take bold risks E.g.: Risking serious injury, the outfielder made an audacious leap against the concrete wall and caught the powerfully hit ball.
Augment	Make greater by adding E.g.: The king augmented his pore by taking over rights that had belonged to the nobles.
Augury	A sign that shows what will happen in the future
August	Inspiring respect and admiration
Austere	Severe or strict in appearance or manner E.g.: Grandfather was an austere man; he used to be silent and very strict to us.
Autonomous	Self-governing or independent E.g.: The Alumni Association is not under the control of the school. It is a completely autonomous group.
Avarice	Extreme greed for wealth or material things E.g.: People who suffer from avarice spend much less and save much more than they should.
Axiom	A statement regarded as obviously true E.g.: Armstrong axioms are the basis for database queries.
Banal	Boring because not new or unusual E.g.: Today, my lecture became too banal because of my quarrel with traffic police during my arrival from home.
Belfry	The place in a bell tower E.g.: My child is still fond of belfry.
Belie	Fail to give a true idea of E.g.: The date of the bridge - 2000 AD - belies the fact that the canal itself is almost 200 years old!.
Beneficent	Having a good effect E.g.: Always expert lectures are beneficent to young interns.
Bevy	A large group E.g.: Being a teacher I never get disturbed by a bevy of students during ceremonial functions.
Bifurcate	Divide into two branches or forks E.g.: Government of AP bifurcated electricity board to improve its performance.
Bilk	Cheat or defraud E.g.: The realtor tried to bilk the homeowner out of his deposit.

Word	Meaning and sample usage
Blight	A plant disease especially one caused by fungi E.g.: The overflowing trash can was a blight on the otherwise beautiful park setting.
Blithe	Without thought or care E.g.: Independent of numerous obstacles from her business opponents, Roja maintains a blithe and bubbly attitude.
Bolster	Support or strengthen E.g.: I observed that a trail run of the game Predator to be bolstering children's intelligence.
Bombastic	Ostentatiously lofty in style E.g.: Since my childhood I hated my father's bombastic life style.
Bonhomie	A good-natured friendliness E.g.: My teachers bonhomie approachable manners made him as this year's best teacher.
Boor	A rough and bad mannered person E.g.: The professor was popular in his subject classes but a terribly boor at social gatherings.
Burgeon	Grow or increase rapidly E.g.: My IT business burgeoned rapidly this year though net profit has diminished.
Burnish	Polish by rubbing E.g.: My company hired me after seeing my high-profile public relations in hopes of burnishing recently tainted company image.
Cabal	A secret political group E.g.: The journalist uncovered evidence that a cabal of power brokers was plotting to overthrow the popular government.
Cacophony	A harsh discordant mixture of sounds E.g.: Even at the age of eighty I visit my village every year only to hear cacophony of horns of my fellow tribal childhood friends.
Cajole	Persuade (someone) to do something using flattery E.g.: My first boy became too naughty because of my mother's consistent cajoling since his birth.
Calumny	The making of false and damaging statements about someone E.g.: During board meeting, my calumny against the Chairman forced me to resign.
Canard	An unfounded rumor or story E.g.: The canard made everybody believe that I and Roja are in a relation.
Candid	Truthful and straightforward; frank E.g.: The candid member of the meeting frequently interrupted to state his bigoted opinion.

Word	Meaning and sample usage
Capitulate	Give in to an opponent or an unwelcome demand E.g.: During my childhood I used to dislike liver and onions but after my marriage my wife forced me to eat it so I had no choice but to capitulate.
Capricious	Prone to sudden changes of mood or behavior E.g.: My visit to USA is completely capricious, yet my opponents followed me.
Cartography	The science or practice of drawing maps E.g.: Because of poor availability of cartography maps of my native town, I cannot set automatic navigation mode in my car.
Castigate	Reprimand severely E.g.: Our old warden reprimanded me many a times for my naughty actions.
Catalyst	A substance that increases the rate of a chemical reaction while remaining unchanged itself; a person or thing that triggers an event
Cathartic	Providing psychological relief through the expression of strong emotions; causing catharsis E.g.: Our graduation ceremony was a cathartic experience which I cannot forget in my life.
Catholic	Including a wide variety of things: catholic tastes. E.g.: Her library collection was catholic; books on many topics are available.
Caustic	Able to burn through or wear away by chemical action. E.g.: My client's caustic warning regarding delivery deadline made me work during nights to finish the job.
Chaos	Complete disorder and confusion E.g.: When he is elected as a Prime Minister country is in utter chaos.
Chauvinist	A person displaying extreme or unreasonable support for their own country, cause, group or sex E.g.: Male chauvinism that is dominating in India lead to poor female, male ratio.
Chicanery	The use of trickery to achieve one's aims E.g.: The peddler often used chicanery to convince people that his goods were worth buying.
Circumspect	Cautious or sensible E.g.: It is better to be bold than to be too circumspect.
Clamor	A loud and confused noise, esp. that of people shouting vehemently E.g.: They got frustrated and decided to clamor for attention.

Word	Meaning and sample usage
Cloy	Disgusting or sickening because of excessive sweet or sentiment E.g.: I enjoy learning new things, but too much studying cloys my desire to even go to school anymore.
Coagulate	Change to a solid or a semi solid state E.g.: Allow the gelatin to almost coagulate before adding banana slices.
Coalesce	Come or bring together to form a mass or whole E.g.: Only after interrogating house maid, the various clues available with the detective began to coalesce into a complete sequence of events.
Coffer	A small chest for holding valuables E.g.: I got a coffer as a prize during my tenth class.
Cogent	Clear, logical and convincing E.g.: My solicitors cogent arguments, got me a bail on the same day of my arrest.
Collusion	Secret cooperation in order to cheat or to deceive E.g.: My wife's collusion with my opponent made to frustrate.
Compliant	A reason for dissatisfaction E.g.:My complaint to Goverment against the municipality made them to release water regularly.
Compunction	A feeling of guilt that prevents or follows wrongdoing E.g.: Mohammad Ghazani did not show any compunction for looting India several times.
Conciliatory	Make calm and content E.g.: After recent students unrest, Home Ministry concentrated on conciliatory measures to continue peace.
Condone	Accept or forgive (an offence or wrongdoing). E.g.: How can the teachers condone such a bad behavior of the minister's son?
Connoisseur	An expert judge in matters of taste E.g.: My frequent visits to the best restaurants in France, made me a connoisseur of fine wines that are available all over the globe.
Consensus	General agreement E.g.: I became successful in bringing consensus in our apartment's general body meeting.
Conspicuous	Clearly visible; attracting notice E.g.: Because of conspicuous errors in my report, I did not get nomination.
Contentious	Causing or likely to cause disagreement or controversy E.g.: That contentious child in the grocery store gave me a headache.

Word	Meaning and sample usage
Contrite	Very sorry for having done wrong. E.g.: His guilty face and apologetic tone showed that he felt contrite.
Contumacious	Stubbornly or willfully disobedient to authority E.g.: Her contumacious attitude gave her detention for a week.
Conundrum	A confusing or a difficult problem or a question E.g.: I'm in a bit of a conundrum- I locked my keys in the car.
Convoluted	Extremely complex E.g.: It is impossible to get any new laws passed in that convoluted government bureaucracy.
Corpulent	Fat E.g.: Because of my corpulent body, I could not join the army though I love it.
Cqorroborate	Confirm or give support to a statement or a theory E.g.: I am always forced to corroborate my wife because of her convincing explanations.
Cosset	Care for and protect in an excessively soft-hearted way E.g.: The poor child wanted to feel cosset.
Coterie	A small exclusive group of people with shared interests or tastes E.g.: Recent unrest in the capitol can be attributed to Prime Minister's coterie.
Craven	Cowardly E.g.: A hero risks his life to help others while a craven runs away from the scene.
Credulous	Too ready to believe things E.g.: A credulous rumor is spreading in the college about joining a new, strict principal who is from military.
Crescendo	A gradual increase in loudness in a piece of music; a climax E.g.: The crescendo in the orchestra excited the audience at the end.
Crestfallen	Sad and disappointed E.g.: My wife was crestfallen when she saw the bad rank of our daughter in JEE examination.
Cupidity	Greed for money and possessions E.g.: My father's cupidity led him to problems and in the end resulted in a prison sentence.
Curmudgeon	A bad-tempered person E.g.: My neighbor is widely viewed as a curmudgeon who never had anything good to say about his neighbors.
Daunt	Cause to feel nervous or discouraged E.g.: Because of my daunting answers during the Interview, I was not selected for the coveted post.

Word	Meaning and sample usage
Debase	Lower the value, quality or character of E.g.: Recent debasing rules that are introduced in the cricket game, many people lost interest in it.
Debutante	A young upper class woman making her first appearance in the society E.g.: The debutante's photograph was at the head of the society page.
Declivity	A downward slope
Decorous	In good taste; polite and restrained E.g.: Members of the diplomatic corps are expected to behave in a decorous manner.
Decorum	Polite and socially acceptable behavior
Deface	Spoil the appearance of E.g.: The ecliptic lover of my neighbors daughter defaced her with acid.
Deference	Humble respect E.g.: My beloved parents do not drink alcoholic beverages, so in deference to their beliefs I have not habituated to drink wine with my meal.
Deleterious	Causing harm or damage E.g.: Pollution has deleterious effects on our environment.
Delineate	Describe or indicate precisely E.g.: At last, we have delineated the responsibilities of new incumbent to our office.
Demagogue	A political leader who appears to the desires and prejudices of the public E.g.: Hitler's type of political leadership made him something of a demagogue.
Demean	Cause to suffer a loss of dignity or respect E.g.: My father in law demeaned me by listening to the words of my wife.
Demur	Show reluctance E.g.: She thought the home was beautiful, but she demurred at the price.
Demure	(of a woman) reserved, modest and shy
Denunciation	The action of denouncing E.g.: The king's denunciation leads to anarchy in our country.
Deprecate	Express disapproval of E.g.: Do not deprecate their actions until you know the whole story.
Deride	Ridicule E.g.: The teacher derided the student in front of the entire class.
Derision	Scornful ridicule or mockery Eg: The matrix revolutions found itself on the wrong end of more criticalderision than any decent film in quite some time.

Word	Meaning and sample usage
Derivative	Something which is derived from other source E.g.: Copycats are never as good as the original and always feel derivative.
Desiccate	Remove the moisture from E.g.: Salts desiccate plants and can become toxic to many plant species.
Desultory	Lacking purpose or enthusiasm E.g.: After some time his output of talk became more desultory as he continued to fail to make progress.
Detached	Separate or disconnected E.g.: He brought an essential element of sadness to the role of a man always slightly detached from the action.
Deterrent	A thing that deters or intend to deter E.g.: A border of at least a meter, filled with thorny shrubs is a great burglar deterrent.
Detraction	A petty disparagement E.g.: The candidate responded sharply to the long list of detractions concocted by his opponent
Diaphanous	Light, Delicate and semi-transparent E.g.: The stain glass window was very diaphanous I could almost clearly see the clouds outside.
Diatribe	A harsh forceful verbal attack E.g.: My mother's lengthy and vehement diatribe against my fiancé irritated me a lot.
Dichotomy	A separation or contrast between two things E.g.: There was both an increase in unemployment and an increase in consumer confidence, creating an interesting economic dichotomy.
Dictum	A short statement that expresses a general principle E.g.: Yesterday night, my father's advice made me to recall old dictum that "truth emerges more readily from error than from confusion".
Diffidence	Lacking in self-confidence E.g.: I failed in my interview because of my diffidence.
Diffident	Shy because lack of self confidence E.g.: His diffident behavior forced board to eliminate him from finals.
Diffusion	The action of spreading over a wide area E.g.: The diffusion of a rumor that a psycho is moving in the area forced me to cancel my journey.
Dilate	Make or become wider, larger or more open. E.g.: After listening patting words by my principal, my chest got dilated by two inches.

Word	Meaning and sample usage
Dilatory	Slow to act E.g.: The workforce has become very dilatory since the announcement of the low pay rise agreement.
Dilettante	A person who dabbles in a subject for enjoyment but without serious study E.g.: The dilettante felt that his superficial knowledge of art qualified him to judge the artist work.
Diplomatic	Having to do with diplomacy; tactful E.g.: His diplomatic attitude made him as the Director.
Dirge	A lament for the dead E.g.: At the end of the memorial service, a mournful dirge was played on the bagpipes by three pipers.
Disabuse	Persuade (someone) that an idea or a belief is mistaken E.g.: He quickly disabused the notion that the world was flat when he returned with evidence that it was indeed round.
Discern	Recognise or be aware of E.g.: The reasons behind this sudden change in my wife's behavior are difficult to discern.
Discerning	Having or showing good judgment E.g.: His discerning nature made me as his favourite.
Discombobulated	Disconcert or confuse E.g.: I am hesitant to forward him to my Manager's attention as I am sure he would discombobulate him since he is easily confused.
Discomfited	Making uneasy or embarrassed E.g.: The young man was discomfited being the only male in the play.
Discordant	Not in harmony or agreement E.g.: My marriage was discordant; I couldn't take another minute of such misery.
Discrepancy	A difference between the things expected to be the same E.g.: A small discrepancy in his application form forced us to eliminate him in finals.
Discrete	Individually separate and distinct E.g.: Because of his discrete attitude, his children also left him.
Disingenuous	Not sincere, especially in pretending ignorant of something E.g.: Politicians make disingenuous promises during the election campaign knowing full well that they could never fulfill them.

Word	Meaning and sample usage
Disinterested	Not influenced by personal feelings, impartial E.g.: Once, he responded to a client's request with a bunch of pie charts that measured his disinterest in the job.
Disjointed	Lacking coherence, disconnected E.g.: His disjointed answers made jury to send him for mental checkup.
Disparage	Speak of (someone or something) as being of little worth E.g.: His intent was to disparage his opponent.
Disparate	Very different in kind E.g.: The team achieved success despite the many disparate personalities.
Dissemble	Hide or disguise one's motives or feelings E.g.: In a television interview the politician tended to dissemble rather than to answer questions truthfully.
Disseminate	Spread information widely E.g.: Main motto of Internet is to disseminate knowledge.
Dissonance	Lacking harmony; discordant E.g.: The cognitive dissonance that results from $smoking_2$ despite knowing that smoking is unhealthy for you is massive.
Distaff	A stick or spindle on to which wool or flax is wound for spinning E.g.: Could you please hold this distaff, whilst I relieve myself in the toilet?
Distend	Swell because of internal pressure E.g.: The poor, malnourished child's belly was distended so far she couldn't see her feet, almost as if she was pregnant.
Distill	Extract the most important aspects of E.g.: The essay would be much better if you could distill the information and reduce the essay to half of its present length.
Dither	Be indecisive E.g.: The dithering old women needed help from her nurse to function.
Diurnal	Of or during the day time ; daily
Diverge	Be different from E.g.: My Mother and I have very divergent ideas about how I dress for my marriage function.
Divine	Having to do with God; excellent E.g.: His divine character forced everyone to accept him as our priest.
Doctrinaire	Very strict in applying beliefs or principles E.g.: His continued support of impractical theories marked him as a doctrinaire.

Word	Meaning and sample usage
Dogma	A principle or principles laid down by authority and is intended to accept without question E.g.: The Catholic Church has endless dogmatic beliefs.
Dormant	Temporarily inactive E.g.: In our country, one can find both active and dormant volcanoes.
Droll	Amusing in a strange or unexpected way E.g.: He enjoyed their walk and thought she had a droll sense of humor.
Dupe	Deceive E.g.: He duped me again in my recent trip by his car.
Dyspeptic	Relating to or suffering from dyspepsia; irritable E.g.: I have been plagued with dyspepsia ever since I took this job last year.
Ebullient	Cheerful and full of energy E.g.: I was attracted by my wife because of her ebullient personality.
Eclectic	Deriving ideas or style from a wide range of sources E.g.: I have an eclectic taste in music.
Edify	Improve the mind or character of someone by teaching E.g.: A preacher's responsibility is to edify his pupil.
Efficacy	The ability to produce an intended result E.g.: His efficacy in all the projects raised him to the coveted post of chairman.
Effigy	A sculpture or a model of a person E.g.: Chief Minister's effigy was burnt by angry mob.
Effrontery	Insolence or impertinence E.g.: He has the effrontery to accuse Kant of barbarous jargon.
Effusive	Expressing pleasure or approval in an unrestrained way E.g.: She was so effusive in describing her husband that I felt drowned in words.
Elicit	Produce or draw out a response or a reaction E.g.: Her tears elicited great sympathy from her audience.
Eloquent	Clearly expressive E.g.: Her eloquence was sufficient to persuade her manager.
Embellish	Make more attractive; decorative E.g.: When I grow up, I do not want to live life with my face held down, I want to embellish life.

Word	Meaning and sample usage
Empower	Give authority or power to E.g.: It is high time to empower newly recruited teachers to meet 21st century needs.
Emulate	Try to be equal or better than E.g.: She emulates her mother and grandmother which made us to laugh a lot.
Encomium	A speech or piece of writing expressing praise E.g.: An encomium by the President greeted the returning hero.
Endemic	(of a disease or condition) regularly found among particular people or a certain area E.g.: Charminar is a memorial of people who died of plague endemic in Hyderabad.
Enervate	Cause to feel drained of energy E.g.: Prolonged exposure to the sun and dehydration enervated the shipwrecked crew, leaving them almost too weak to hail the passing vessel.
Engender	Give rise to E.g.: Continued interference by the West in the affairs of Iran, could engender another war.
Enigma	A mysterious or puzzling person or thing E.g.: The crossword puzzle was nothing but an enigma for the confused student.
Enmity	Hostility E.g.: Continued enmity of Pakistani people with Indians pains me a lot.
Enumerate	Mention one by one E.g.: She has enumerated all the difficulties that she is facing.
Ephemeral	Lasting or living for a very short time E.g.: Her ephemeral smile always amuses me.
Epicure	A person who takes particular pleasure in good food or drink E.g.: I love going to parties at an epicure's house because they have the best food.
Equanimity	Calmness E.g.: My equanimity at the accident site where my wife is dead raised doubts about my involvement in the accident.
Equivocate	Use language that can be understood in more than one way to avoid the truth E.g.: Her equivocal statements proved she is best suitable for political career.
Erratic	Not even or regular in pattern or movement E.g.: His erratic behavior during first month of his stay made me to think about his candidature.
Ersatz	Made as a poor quality substitute for something else E.g.: When I brought home an ersatz diamond ring, my fiancé threw me out.

Word	Meaning and sample usage
Erudite	Having or showing knowledge or learning E.g.: His erudite speech made all of us to clap continuously.
Eschew	Deliberately avoid doing something E.g.: I try to eschew answering questions like these.
Esoteric	Intended for or understood by only a small number of people with special knowledge E.g.: This is a compilation of esoteric philosophical theories
Estimable	Worthy of great respect E.g.: He is one of the estimable young professor of our university.
Ethos	The characteristic spirit of a culture, era or a community E.g.: The free-love movement was typical of the ethos of the late 1960's.
Eulogy	A speech or piece of writing that praises someone highly E.g.: I was asked to give the eulogy at my best friend's funeral.
Euphemism	A less direct word used instead something that is harsh or blunt when referring to something unpleasant or embarrassing
Euphony	The quality of having a pleasant sound E.g.: Her speaking ability was excellent and her euphonious voice made all of us to listen to her without murmuring a single word.
Euphoria	A feeling of great happiness E.g.: He was filled with euphoria when he heard the news of the birth of his son.
Exacerbate	Make a problem or bad situation worse E.g.: My chairman's vociferous speech exacerbated our company's share values.
Exculpate	Show or declare to be not guilty of wrongdoing E.g.: The jury had to exculpate the defendant due to lack of evidence.
Exigent	Pressing E.g.: I became more exigent over his pronunciation.
Exonerate	Officially declare free from blame E.g.: This evidence in your hand will certainly exonerate you; you can no longer be charged.
Explicit	Clear and detailed. E.g.: His explicit directions helped us to reach quickly.
Exponent	A person who does a particular thing skillfully E.g.: Jim was an exponent of the idea that "god" is a clown suit for the laws of physics.

Word	Meaning and sample usage
Expurgate	Remove matter seen as obscene or unsuitable E.g.: The writings of the female Prophets were expurgated from the old texts.
Extensive	Covering a large area E.g.: My extensive lawn area is covered with Australian grass.
Extrapolation	1. To infer or estimate by extending or projecting known information 2. *Mathematics* To estimate (a value of a variable outside a known range) from values within a known range by assuming that the estimated value follows logically from the known values E.g.: I am able to extrapolate after her long description of the incident.
Facetious	Treating serious issues with inappropriate humor E.g.: I was irritated with John's facetious remarks about the game.
Facilitate	Make easy or easier E.g.: We were relieved after seeing John as we were confident that he can facilitate us in all unexpected situations during the proposed tour.
Fallible	Capable of making mistakes or being wrong E.g.: Because of his fallible attitude, he missed his selection.
Fallow	1. (of farmland) ploughed but left for a period being planted with crops 2. when very little is done or achieved E.g.: For many contract overseas workers there is a cycle of good earnings followed by fallow periods.
Fanatical	A person filled with excessive enthusiasm for an extreme political or a religious cause E.g.: He is a fanatic.
Fatuous	Silly and pointless E.g.: Gamblers often fatuously lose their money at casinos.
Fawn	Try to gain favor by servile flattery or attentive behavior E.g.: It is so annoying how she fawns all over him.
Fecund	Highly fertile E.g.: I have been working hard to make this soil fecund.
Felicitous	Well chosen or appropriate E.g.: I have given felicitous answers in the interview, thus chances are more for my selection.
Fervid	Fervent; very passionate E.g.: His fervid speech opposing child labor, touched the hearts of the listeners.

Word	Meaning and sample usage
Fervor	Intense and passionate feeling E.g.: The presidential candidates have much fervor in their campaigns.
Fetid	Smelling very unpleasant E.g.: The fetid odor of spoiled sea bass wafted from the fishing trawler.
Florid	Having a red or flushed complexion E.g.: The boy's face was florid after he got caught wearing his mother's underwear.
Flout	Openly fail to follow (a rule or a custom) E.g.: She shouldn't have flouted that red light. You have flouted the laws of this institution and for that you will be punished.
Foment	Stir up(revolution or conflict) E.g.: Fidel Castro fomented a coup de tat and soon took over Cuba.
Forbearance	Patient self-control E.g.: I pleaded with my enemy to show forebearance, but instead he overwhelmed me with his hatred.
Ford	A shallow piece in a river or a stream where it can be crossed E.g.: We are looking for a ford to cross easily.
Fortuitous	Happening by chance, Lucky E.g.: I am very fortuitous to receive this job offer.
Fractious	Bad tempered, Difficult to control E.g.: My friend was particularly fractious when it came to the topic of politics.
Frenetic	Fast and energetic in a disorganized way E.g.: Calm down and stop those frenetic movements.
Frugality	Sparing with money or food E.g.: My mother is frugal when it comes to spending for herself.
Fulminate	Express strong protest E.g.: Public officials across the political spectrum fulminated against the perceived security threat.
Furtive	Trying to avoid being noticed in a secretive or in a guilty way E.g.: The young convict made many furtive attempts to escape from prison.
Gainsay	Deny or contradict; speak against E.g.: One cannot gainsay the fact that the proliferation of contract positions in Ministries and departments creates dissatisfaction among those serving officers.
Gambol	Run or jump about playfully E.g.: We watched the lambs as they gamboled in the field.

Word	Meaning and sample usage
Garner	Gather or collect E.g.: When you have learned the definition of garner, you will have garnered the knowledge to use it in a sentence.
Garrulity	The quality of being extremely talkative E.g.: With the confidential garrulity of a man who has dined too well, he plunged into his darling topic, and I looked past him at the clock.
Garrulous	Extremely talkative E.g.: She was unable to sleep on the flight because of the garrulous passenger sitting beside her.
Gestation	1. The process of carrying or being carried in the womb between conception and birth 2. Duration of pregnancy
Glib	Able to express oneself well but not meaning what one says E.g.: The congress man found it easy to give glib answers when asked a direct question.
Glower	Have an angry or sullen look on one's face E.g.: The bad sport sat in a corner and decided to glower at his opponents.
Gradation	A scale of successive changes, stages or degrees E.g.: Apples gradation takes place before anything in jam factory.
Gratuitous	Done without good reason or purpose E.g.: I am a victim of a gratuitous Mother-in-Law.
Gregarious	1. Fond of company(sociable) 2. (of animals) living flocks or colonies E.g.: Her gregarious appearance made me to love her at first meeting.
Grievous	Serious physical injury inflicted on a person by the action of another E.g.: It was grievous to see in a short time how poorly they lived.
Grovel	Crouch or crawl on the ground E.g.: My dog knew he was in trouble; he began to grovel as soon as I came in and saw the mess he had made.
Guile	Sly or cunning intelligence E.g.: He borrowed without guile, someone else's work as his own.
Guileless	Very honest or sincere E.g.: The guileless youth was no match for the tricks of the street-smart hood.
Gullible	Easily persuaded to believe something E.g.: I am always worried about my young daughter's gullible nature.

Word	Meaning and sample usage
Hamper	Slow down or prevent the movement or progress of E.g.: I don't want your presence here because my work gets hampered.
Hapless	Unlucky E.g.: I am hapless even after working hard and spending money.
Harangue	Criticize at length in an aggressive manner E.g.: She delivered her harangue with much venom, leaving her audience in shock.
Hegemony	Leadership or dominance especially by one country or social group E.g.: I rendered my hegemony to the confounded people of this country.
Hermetic	(of a seal or a closure) complete air-tight E.g.: When I bought new containers, they were very hermetic.
Hoary	1. Having grey hair 2. Old and unoriginal E.g.: Everything told of long use and quiet slow decay; the very bell-rope in the porch was frayed into a fringe, and hoary with old age.
Hyperbole	Exaggerated statements that are not meant to be taken in strict sense of words E.g.: Was the car really as fast as a bullet, or is that just hyperbole?
Iconoclast	A person who attacks popular beliefs or established values and practices E.g.: I would never be an iconoclast because I would never attack cherished beliefs.
Idiosyncrasy	A person's particular way of behaving or thinking E.g.: My maternal uncle's idiosyncrasy divided our family.
Ignoble	Not good or honest E.g.: Getting someone else to do your homework for you is fairly ignoble.
Imbue	Fill with a feeling or quality E.g.: He was imbued with a desire for social justice.
Imminent	About to happen E.g.: It is imminent to sack our president.
Immutable	Unchangeable E.g.: His immutable attitude made him as a best karate trainer.
Impair	Weaken or damage E.g.: If I cut my fair hair it will be sure to impair.
Impasse	A situation in which no progress is possible E.g.: We have come to an impasse in our discussion.

Word	Meaning and sample usage
Impassive	Not feeling or showing emotion E.g.: The doctor made his examination with impassive face.
Impecunious	Having little or no money E.g.: Reddy was certainly not an impecunious man.
Imperturbable	Unable to be upset or excited E.g.: We couldn't believe that Rafael would be that imperturbable in the midst of a riot.
Impervious	Unable to be affected by E.g.: After repeated application of this jelly, roof becomes impervious to water.
Impetuous	Acting or done quickly without thought or care E.g.: He ducked impetuously under the ball.
Impious	Not showing respect or reverence E.g.: The church accused him of impiety and had all his writings burned.
Implacable	Unwilling to stop being hostile towards someone or something E.g.: I am implacable in my opposition to your proposal.
Implicit	Suggested though not directly expressed E.g.: It is implicit that he is not interested in my project.
Imprecation	A spoken curse E.g.: The wizard muttered imprecations as he worked.
Imprudent	Not showing care for the consequences of an action; rash E.g.: Her majesty took a great dislike at the imprudent behavior of many of the ministers and readers.
Impugn	Express doubts about the truth or honesty of E.g.: By not being honest, I fear that people will impugn your motivations.
Incarnadine	To make incarnadine, especially to redden.
Inchoate	Just begun and not so fully formed or developed E.g.: The project was still inchoate and far from completion.
Inculcate	Fix an idea or a habit in someone's mind by repetition E.g.: A good teacher inculcates good practices that are followed by the students.
Indiscriminate	Done or acting without careful judgment E.g.: Indiscriminate removal of trees increased earth's temparatures.
Indolence	Lazy E.g.: His indolence stopped him from going farther in his job.

Word	Meaning and sample usage
Inequity	Lack of fairness or justice E.g.: There was such inequality of positions in the office that I just quit.
Inertia	A tendency to do nothing or to remain unchanged E.g.: I decided to remove inertia in my elder son such that he finds a job at the earliest.
Inexorable	Impossible to stop or prevent E.g.: The inexorable truth of the matter was, she was not going to escape this penitentiary alive.
Ingenuous	Innocent and unsuspecting E.g.: His ingenuous reply made all of us to laugh.
Ingrate	An ungrateful person
Ingratiate	Bring oneself into favor with someone by flattering or trying to please them E.g.: Please ingratiate my answer because I answered this all I expect to get back is the best answer.
Inherent	Existing in something as a permanent, essential or characteristic attribute E.g.: His inherent characteristic of calmness fetched this job.
Inimical	Unfriendly; hostile E.g.: The inimical police officer writes the guy a ticket.
Iniquity	Immoral or grossly unfair behavior
Innocuous	Not harmful or offensive E.g.: Mary was scared that the dog was going to hurt her baby, but after a while she found that the dog was innocuous.
Inquest	An official inquiry to gather facts relating to an incident E.g.: The inquest was scareed when thrown before the judge.
Insensible	Numb; without feeling
Insipid	Lacking flavour. Lacking liveliness or interest E.g.: Because of a his mothers bad health, Rao's disco function became too insipid.
Insurrection	A violent uprising against an authority E.g.: We hear that the country is full of insurrection as loyalists and rebels rise against each other.
Inter	1. Place a dead body in a grave or in a tomb 2. Between
Interregnum	A period between regimes when normal Government is suspended.
Intractable	Hard to deal with E.g.: His statements are intractable thus we could not win the treasure hunt.

Word	Meaning and sample usage
Intransigent	Refusing to change one's views or behavior E.g.: A person who refuses to agree or compromise can be described an intransigent.
Intrepid	Fearless; adventurous E.g.: Columbus was an intrepid explorer, showing his endurance, resolute fearlessness and fortitude.
Inundate	Flood; overwhelm with things that are to be dealt with E.g.: River next to my town inundated my farm house.
Inure	Become used to something unpleasant E.g.: By taking cold showers, a person can become inured to short exposure to cold environments.
Invective	Strongly abusive or critical language E.g.: I thought his answer was quite invective; I rushed to Principal.
Investiture	The action of formally investing a person with honors or rank E.g.: The Federal government's proposed investiture of fifteen billions dollars may not be sufficient to save the ailing big three automakers.
Invidious	Unacceptable and unfair and likely to arouse resentment or anger among other people E.g.: Radha became jealous when her two best friends began to forge a friendship of their own, so she began to make invidious comparisons about each one behind the other's back, in an effort to split them up.
Irascible	Hot-tempered; Irritable E.g.: When I get irascible, I cursed and yelled.
Itinerant	Travelling from place to place.
Itinerary	A planned route or journey E.g.: I always ask my office people to make my itinerary ready before on month of my travel date.
Jargon	Words or expressions used by a particular group that will be difficult for others to understand E.g.: The science student found it hard to understand the jargon of the astronomers.
Jettison	Throw or drop from an aircraft or a ship
Jingoism	Excessive support for one's country
Jocular	Humorous E.g.: His jocular behavior eliminated him from army selections.
Judicious	Having or done with good judgment E.g.: I expect judicious treatment for this patience.

Word	Meaning and sample usage
Juncture	A particular point of time E.g.: At this juncture, I don't want to withdraw money from bank.
Keen	Interested in E.g.: A visiting team shown keen interest in my project.
Kindle	Light (a flame) or arouse (an emotion) E.g.: My husband and I rekindled our love for each other by sharing a cozy evening by the kindling fire.
Knell	The sound of a bell rung solemnly E.g.: As the mourners gathered at the graveside, they heard the solemn knell of the church bell.
Kudos	Praise and honor E.g.: John received some well-deserved kudos for finishing at the top of his class.
Lachrymose	Tearful and sad E.g.: The death of her husband turned her into a lachrymose woman, ready to cry at the drop of a hat.
Laconic	Using very few words E.g.: All I received in response to my request was the laconic reply, Wait.
Lament	A passionate expression of grief E.g.: The whole family was lamenting over their uncle's unexpected death.
Lampoon	Publicly mock or ridicule E.g.: The orator was lampooned in the editorial pages for giving an ill conceived answer.
Languid	Reluctant to exert oneself physically E.g.: His lack of interest was apparent with the languid manner in which he sulked across the stage.
Lapidary	Relating to the engraving , cutting or polishing of stones or gems E.g.: A lapidary artist creates beautiful jewelry with the use of rocks and minerals.
Larceny	Theft of personal property E.g.: Once the trial ended, the man was charged with the crime of larceny.
Largess	1. Generosity 2. Money or gifts given generously E.g.: The vassal was able to make a secure living thanks to the largess of the king.
Lascivious	Feeling or showing open or offensive sexual desire E.g.: Lessica Rabbit is a most lascivious woman.
Lassitude	Physical or mental weariness; lack of energy E.g.: Mr. Marshall hated teaching during the period following lunch, it was impossible to overcome his students' lassitude and capture their attention.

Word	Meaning and sample usage
Latent	Existing but not yet developed, apparent or active E.g.: The employee saw a latent value of the Egyptian lamp.
Laud	Praise highly E.g.: He always lauds my efforts which are my main strength in passing examinations.
Laudatory	Expressing praise and commendation E.g.: For those who believe in and love God, inherent in their expressions of God are laudatory statements of Him.
Lavish	Very rich, elaborate or luxurious E.g.: These lavish arrangements for my marriage are at the demand of my father.
Leery	Cautious or wary E.g.: She hesitated stepping into the friendly crowd, because she was leery of strangers.
Legerdemain	Trickery E.g.: The magician's legerdemain was quite evident in the complexity of his tricks.
Lethargic	Lacking energy or enthusiasm E.g.: I became lethargic after my poor performance in mathematics.
Levity	The treatment of a serious matter with humor or lack of respect E.g.: The only moment of levity during the conference came when one speaker sang at the end of her speech.
Liberal	Given, used or giving in generous amounts E.g.: My mathematics teacher is liberal in giving marks; so all of us are fans of her.
Libertine	A man who behaves immorally E.g.: His search for fun at the expense of others clearly showed John's libertine attitude towards life.
Licentious	Sexually immoral E.g.: I have never met such a licentious slut in my life.
Limpid	Clear E.g.: My limpid remarks made my son to realise the facts.
Lionize	Treat as a celebrity E.g.: Today we will lionize her since its her birthday.
Lissome	Slim, supple and graceful E.g.: The lissome actress's dance training is apparent in the way she moves on stage
Listless	Lacking energy or enthusiasm E.g.: Sometimes students have trouble paying attention in summer school because the heat makes them listless.

Word	Meaning and sample usage
Livid	Furiously angry E.g.: I was livid when I found none in my class who didn't know what is the correct value of 2+3*4.
Loquacious	Talkative E.g.: I have this loquacious girl who sits behind me in my class that is always getting me into trouble.
Lucid	Easy to understand; clear E.g.: I love my Physics teacher because of her lucid explanations.
Lugubrious	Sad and dismal E.g.: The widow was very lugubrious at her rich dead husband's funeral.
Lumber	Relating to the lower back E.g.: The Semi Truck lumbered up the hill with it's over sized load.
Luminous	Bright or shining especially in the dark E.g.: The luminous stars lit up the dark sky
Machination	Plots and scheming E.g.: To succeed in life, she made many risky machinations.
Maelstrom	A powerful whirlpool E.g.: As I entered the store I was engulfed by a maelstrom of holiday shoppers.
Magnanimity	Generous or forgiving towards a rival or enemy E.g.: My grandfathers magnanimity is well known in my district.
Magnate	A wealthy and influential businessman or businesswoman
Malediction	A curse E.g.: The poor boy couldn't shake the malediction that was put upon him by the old witch.
Malevolent	Wishing evil to others E.g.: I was forced to suspend the services of my personnel assistant because of her malevolent behavior.
Malinger	Pretend to be ill in order to avoid work E.g.: Lazy students malinger, rather than prepare, hence their low grades.
Malleable	Easily influenced E.g.: Usually mothers are malleable compared to fathers.
Mannered	Behaving in a specified way E.g.: Because of my well mannered dress, I was selected.
Mar	Spoil the appearance or quality of E.g.: He was careful not to mar the surface as he carried the priceless antique up the stairs.
Martinet	A person who enforces strict discipline E.g.: I'm a martinet in the classroom because I don't believe in kids cheating on their homework by having us write their sentences for them.

Word	Meaning and sample usage
Maudlin	Sentimental and full of self-pity E.g.: I was joyful yet maudlin as I recalled my childhood.
Maverick	An unconventional or independent-minded person E.g.: The gambler was a real maverick, making bets seemingly at random and without reason.
Mendacious	Untruthful E.g.: The feuding neighbors frequently called the police to report mendacious complaints about each other.
Mendicant	Regularly engaged in begging E.g.: The mendicant orders depend directly on charity for their livelihood.
Mercurial	Tending to change mood suddenly E.g.: Everyone stayed away from John, his temper was mercurial.
Meretricious	Appearing attractive but in reality having no value E.g.: The plastic surgery that she had done had made her appearance nothing but meretricious.
Metaphor	A figure of speech in which a word or phrase is used to represent or stand something else.
Meticulous	Very careful and precise E.g.: His meticulous plan avoided shut down of our factory.
Militate	Be a powerful or decisive factor E.g.: There are several thing that militate against a relationship.
Mirth	Laughter E.g.: The joke that the teacher told, left the whole class in mirth, even when they thought about what he had said they cracked up.
Misanthrope	A person who dislikes and hates other people E.g.: His misanthropic attitude forced the village to throw him from the village.
Missive	A letter E.g.: I sent a missive to my mother while I remembered to do so.
Mitigate	Make less severe, serious or painful E.g.: I have been nominated by the president to mitigate the situation in the eastern province.
Mollify	Lessen the anger or anxiety of E.g.: I have been nominated by the president to mollify the people eastern province who is demanding separate state.
Molt	Shed old feathers, hair, or skin, or an old shell, to make way for a new growth E.g.: My pet penguin is not molting, penguins do not molt.
Monastic	Relating to monks or nuns or their communities E.g.: The pet detective lived a monastic life after the death of a raccoon he was trying to save.

Word	Meaning and sample usage
Monogamy	The state of having only one life partner
Monotony	Lack of variety and interest; tedious repetition and routine E.g.: After my wife's death, I am suffering from monotony.
Mores	The customs and conventions of a community E.g.: Pathetically, our educational mores have it that the more A's a student makes, the better their education.
Multifarious	Having great variety E.g.: Leonardo da Vinci's notebook reveals that he was a man of multifarious interests.
Mundane	Lacking interest or excitement E.g.: He led a rather mundane life, thus he advised his children to plan for joyful life.
Mutinously	Disposed to or in a state of mutiny E.g.: The tide of undecipherable signatures of mutinous adolescents which has washed over and bitten into the facades of monuments and the surface of public vehicles in the city where I live.
Myopic	Short sighted E.g.: Mr. Jones sought a stronger prescription for his myopic night driving.
Nadir	The lowest or worst point
Nascent	Just coming into existence and beginning to develop E.g.: The nascent theories of the creation of the Universe seem not to agree with the Big Bang theory.
Naive	1. Lacking experience, wisdom or judgment 2. Lacking sophistication E.g.: His naïve solution fetched our company huge returns.
Neologism	A new word or expression E.g.: I assume this is more of a problem with regard to artificially coined neologisms than with words from the spoken language.
Neophyte	A person who is new to a subject , skill or belief E.g.: He was a neophyte to the track team and had the slowest times.
Nettle	Annoy E.g.: He was nettled by her mannerisms in the board meeting.
Noisome	Having a very unpleasant smell E.g.: His table manners were little short of noisome.
Nominal	In name but not in reality E.g.: Teacher warned me and said this punishment is nominal as this is first time I acted mischievously.

Word	Meaning and sample usage
Nuance	A very slight difference in meaning, expression, sound etc E.g.: The nuances of red in her hair glistened in the afternoon sun.
Numismatics	The study or collection of coins, banknotes, etc
Obdurate	Stubbornly refusing to change one's mind E.g.: Although all evidence was against her, Radha was still obdurate in her conclusion that the prisoner was innocent.
Obfuscate	Make unclear or hard to understand E.g.: Politicians keep obfuscating the national issues, making the issues obscure, confusing, and unclear.
Oblique	At an angle; slanting
Obsequious	Obedient or respectful to an excessive degree E.g.: His sales pitch was so obsequious, I was turned off, and I went across the street and bought it from another store.
Obstinate	Stubbornly refusing to change one's mind E.g.: My daughter and I are both obstinate.
Obviate	Remove or prevent a need or difficulty E.g.: I was nominated to obviate the problems in the University.
Occlude	Close up or block(an opening or a passage)
Officious	Asserting authority in an overbearing way E.g.: If you describe someone as officious, you are critical of them because they are eager to tell people what to do when you think they should not.
Onerous	Involving much effort and difficulty E.g.: My father has left with me the onerous task of arranging marriages of my three sisters.
Opine	State as one's opinion E.g.: You might write an article about the virtues of the macroeconomic theory of Keynes, to which I would opine that Keynes was an idiot and that Friedman has already refuted him.
Opprobrium	Harsh criticism E.g.: Kanye West faced vast opprobrium after interrupting Taylor Swift at the VMAs.
Orotund	Deep and impressive E.g.: The speaker has a deep, orotund voice that projects to the farthest corners of the room.
Oscillate	Move or swing back and forth at a regular rate
Ossify	Turn into bone or bony tissue E.g.: The cartilage was ossified when it was converted into bone.
Ostensible	Apparently true but not necessarily so E.g.: Her ostensible purpose was borrowing sugar, but she really wanted to see the new furniture.

Word	Meaning and sample usage
Ostentation	Showy display which is intended to impress
Ostracize	Exclude from a society or a group E.g.: Due to his Constant questioning of the Greek Political Figures, Socrates was Ostracized, then eventually, sentenced to death by drinking Hemlock-laced wine.
Overwrought	Very worried or nervously excited E.g.: I was overwrought when we were going to the party.
Palatial	Resembling a palace in being spacious and splendid E.g.: I have several 55" HD TVs in his palatial home.
Palliate	Make (a disease or its symptoms) less severe or unpleasant without removing the cause. Allay or moderate (fears or suspicions) E.g.: The nurse provided palliative care to the injured patient.
Pallid	Pale, typically because of poor health E.g.: I knew my kid wasn't lying about being sick. His skin was pallid so I let him stay home from school
Panache	Flamboyant confidence of style or manner E.g.: Despite tripping on the way down the stairs, the contestant continued her walk with great panache.
Panegyric	A public speech or published text in praise of someone or something E.g.: Tom is very panegyric while talking to his co-workers.
Panoply	A splendid display E.g.: The knight wore his panoply during battle.
Panoramic	bird's-eye: as from an altitude or distance
Paradigm	A typical example or pattern of something; a model
Paradox	A seemingly absurd or self-contradictory statement or proposition that when investigated or explained may prove to be well founded or true
Paragon	A person or thing regarded as a perfect example of a particular quality E.g.: Einstein is a paragon for scientific research community.
Pare	Trim (something) by cutting away its outer edges E.g.: Please pare the peel from the apple.
Pariah	An outcast E.g.: The strange boy was viewed as a social pariah by his entire student body.
Parley	A conference between opposing sides in a dispute, esp. a discussion of terms for an armistice E.g.: The Night riders, and the Eagles had a parley a week before the big game.

Word	Meaning and sample usage
Parry	Ward off (a weapon or attack), esp. with a countermove E.g.: Did I "parry" your question's intent?
Parsimonious	Unwilling to spend money or use resources; stingy or frugal E.g.: I always feel sorry for my father's parsimonious attitude though I am bearing the house expenses.
Pastiche	An artistic work in a style that imitates that of another work, artist, or period E.g.: The medley of hits was a pastiche of Johnny Cash's most popular songs.
Pathogenic	Able to cause disease E.g.: The air borne pathogens on the aircraft made everyone ill.
Pathological	Involving, caused by, or of the nature of a physical or mental disease
Peccadillo	A small relatively unimportant offence or sin E.g.: The politician hoped the voters would overlook his peccadillos and focus on his long service record.
Pedantic	Of or like a pedant E.g.: It is important to understand pedantic terminology before beginning a lecture.
Pejorative	Expressing contempt or disapproval E.g.: The pejorative comment deepened the dislike between the two families.
Penchant	A strong or habitual liking for something or tendency to do something E.g.: Elizabeth Taylor has a penchant for husbands.
Pendant	A piece of jewelry that hangs from a chain worn around the neck
Penury	Extreme poverty E.g.: I spent half of my life in penury, thus I know the value of a penny.
Peregrinate	Travel or wander around from place to place E.g.: He brought back souvenirs and photos from his peregrination abroad.
Perennial	Lasting or existing for a long or apparently infinite time; enduring E.g.: Lack of medicines is a perennial problem in our tribal areas.
Perfidious	Deceitful and untrustworthy E.g.: I trusted Tara but it turned out, that she was perfidious.
Perfunctory	Carried out with a minimum of effort or reflection E.g.: One perfunctory answer is to use it carefully.

Word	Meaning and sample usage
Peripatetic	Traveling from place to place, esp. working or based in various places for relatively short periods E.g.: The peripatetic animal never stayed in one place for too long.
Permeable	(of a material or membrane) Allowing liquids or gases to pass through it E.g.: Only the small, non-polar molecules diffused through the selectively permeable membrane.
Permeate	Spread throughout E.g.: Focused ultrasound helps exogenous genes permeate targeted cells' outer membranes.
Perpetual	Never ending or changing E.g.: His perpetual spending made his children's as beggers.
Persistent	Continuing firmly or obstinately in a course of action in spite of difficulty or opposition E.g.: My consistent persuasion made my elder son to continue the discontinued course.
Perspicacious	Having a ready insight into and understanding of things E.g.: He was a perspicacious student of human nature, and knew how we would react.
Pervade	Spread through and be perceived in every part of E.g.: Because of poor educations, AIDS pervaded the world very quickly.
Pervasive	Spreading widely throughout an area or a group of people E.g.: There is a pervasive trend toward casual dress in businesses.
Phalanx	A body of troops or police officers, standing or moving in close formation E.g.: The phalanx defense used by the Spartans helped them to defeat vast numbers of enemies.
Philanthropy	The desire to promote the welfare of others, expressed especially by the generous donation of money to good causes
Philistine	A person who is hostile or indifferent to culture and the arts, or who has no understanding of them E.g.: A philistine girl and her apprentice walked down the streets near the Mediterranean sea.
Phlegmatic	Having an unemotional and stolidly calm disposition E.g.: He was phlegmatic in his attitude toward work.
Pithy	Concise and forcefully expressive E.g.: Old Mr Brown was not known for talking much, but what he did say was always pithy. Pithy means that there is substance and wisdom in what is said.

Word	Meaning and sample usage
Placate	Make someone less angry or hostile E.g.: I have placate my wife whenever he is angry my elder son.
Platitude	The quality of being dull, ordinary, or trite E.g.: A common platitude is the greeting "How are you?"
Plebeian	A commoner E.g.: The plebeian population ends up paying the most taxes.
Plethora	An excessive amount E.g.: When I accepted his offer, he filled my hose with plethora of druses.
Plucky	Having or showing determined courage in the face of difficulties E.g.: Telling that bully to leave your brother alone was a plucky thing to do, David. I am proud of you.
Polemic	A strong verbal or written attack on someone or something E.g.: Thus the Islamic polemic against Christianity has centered on the doctrine of Trinity.
Politic	(of an action) Seeming sensible and judicious under the circumstances
Polyglot	Knowing or using several languages
Posit	Assume as a fact; put forward as a basis of argument E.g.: Posits that immigrants with probable mental on the public.
Potentate	A monarch or ruler E.g.: The king, despite his potentate ways humbled himself like a servant.
Pragmatic	Dealing with things sensibly and realistically in a way that is based on practical rather than theoretical considerations E.g.: His pragmatic speeches lead Democrats into white house.
Prattle	Talk at length foolishly E.g.: Johnny was a bad boy therefore he got the prattle across the ass.
Precarious	Not securely held or in position; dangerously likely to fall or collapse E.g.: His precarious treatment spoiled my health.
Precipitate	Cause (an event or situation, typically a bad one) to happen suddenly, unexpectedly, or prematurely E.g.: Army is deputed with the fear that the same problems may precipitate soon.
Precursor	A person or thing that comes before another of the same kind; a forerunner E.g.: I applied for a loan as a precursor to purchasing a car.

Word	Meaning and sample usage
Predestined	(of God) Destine (someone) for a particular fate or purpose E.g.: When John met Mary, he was immediately sure that they were predestined for each other.
Preen	(of a person) Devote effort to making oneself look attractive and then admire one's appearance E.g.: The contented cat sat preening himself by the fire.
Prescient	Having or showing knowledge of events before they take place E.g.: He had the prescience to realize a confrontation would be deadly.
Prestigious	Inspiring respect and admiration; having high status.
Presumptuous	(of a person or their behaviour) Failing to observe the limits of what is permitted or appropriate E.g.: It was presumptuous of her to join us for dinner uninvited.
Pretentious	Attempting to impress by affecting greater importance, talent, culture, etc., than is actually possessed E.g.: It was pretentious to assume I would change my whole schedule for his benefit.
Prevaricate	Speak or act in an evasive way E.g.: While trying to get to my destination, I ended up on the wrong street, on the wrong side of town, all thanks to the prevarication of the stranger whom I asked directions of.
Pristine	In its original condition; unspoiled E.g.: The fresh snowfall, without tracks or marks of any kind, looked as pristine as mother's white tablecloth pressed to perfection for Thanksgiving Day.
Probity	The quality of having strong moral principles; honesty and decency E.g.: His reputation is largely inherited from his mother, revered for her probity.
Problematic	Constituting or presenting a problem or difficulty
Proclivity	A tendency to choose or do something regularly E.g.: 'Children have the proclivity to keep on playing even when they can hardly keep their eyes open.'
Prodigal	Spending money or resources freely and recklessly; wastefully extravagant E.g.: The prodigal waste of money given to the gambling industry is in the billions of dollars.

Word	Meaning and sample usage
Prodigy	A person, esp. a young one, endowed with exceptional abilities E.g.: My mentoring to my grandson made him as a young prodigy.
Profligate	Recklessly extravagant or wasteful in the use of resources E.g.: The profligate prosecutor was laughed out of the courtroom as the judge yelled at him for constantly wasting her time with minor offenses.
Proliferate	(of a cell, structure, or organism) Reproduce rapidly E.g.: The bugs on the trail were so proliferate that many people decided to give up and go home.
Prolific	(of a plant, animal, or person) Producing much fruit or foliage or many offspring E.g.: He is such a prolific writer; I became his favourite after reading his first novel.
Promiscuous	(of a person) Having many sexual relationships, esp. transient ones E.g.: A promiscuous woman is called a slut, while her male counterpart is called a stud.
Propensity	An inclination or natural tendency to behave in a particular way E.g.: My friend has a real propensity for baking great pies.
Propitiate	Win or regain the favour of E.g.: After offending his boss, John tried to propitiate him by inviting him to dinner.
Propriety	The quality of behaving appropriate.
Prudence	Acting with and showing care for the thought and for the future E.g.: Prudence is demonstrating wisdom based on rational thinking.
Prudent	Acting with and showing care for the thought and for the future
Puerile	Childishly silly E.g.: You need to stop acting in such a puerile manner and act your age.
Pugilism	Boxing; fighting with the fists E.g.: The sideways lean of his nose suggested a pugilistic background.
Pulchritude	Beauty E.g.: When she was alive, Anna Nicole Smith exemplified feminine pulchritude.
Pungency	Having a sharply strong taste or smell
Pungent	Having a sharply strong taste or smell E.g.: By the time we returned from evening walk, the pungent smell of the dog permeated the entire room.

Word	Meaning and sample usage
Pusillanimous	Lacking courage E.g.: The pusillanimous toddler, who had not yet learned social protocol, barked at his mother from an unsocial distance.
Qualified	Meet the necessary standards or conditions to be able to do or receive something
Querulous	Complaining in an irritable way E.g.: My neighbours querulous attitude irritated me a lot.
Quiescent	In a state or period of inactivity E.g.: My quiescence at the dinner table prompted my mother to ask me about my day.
Quixotic	High-minded and unselfish to an impractical extent E.g.: The villain's plan of world domination was obviously quixotic.
Quotidian	Ordinary or every day E.g.: The rules of quotidian face-to-face life are suspended or even inverted.
Raconteur	A person who tells stories in an interesting way E.g.: We always loved to listen to my grandfather talk about life in the old country because he was a great raconteur.
Rarefy	Lessen the density or solidity of E.g.: A shortage of calcium in the bones can cause them to rarefy resulting in the disease of osteoporosis.
Rebuke	Criticize or reprimand sharply E.g.: King rebuked at US invasion.
Recant	Withdraw a former opinion or belief E.g.: I would like to state that my answer to you is positive, but then I would have to recant it to make my intentions true.
Recrimination	An accusation in response to one from someone else E.g.: There was a period of bitter recrimination in my house.
Redoubtable	Worthy of respect or fear E.g.: She is a redoubtable opponent; you must respect and fear her at all times.
Redress	Set right something unfair or wrong E.g.: I have to redress my son for his bad behavior in the class.
Refractory	Stubborn or difficult to control E.g.: Several people offered him really sound advice on the matter but he's an intensely refractory individual and wouldn't be swayed.
Refute	Prove(a statement or a person to be wrong) E.g.: I have to refute my colleagues in front of my superiors.

Word	Meaning and sample usage
Rejoinder	A quick or witty reply E.g.: The comedian's witty rejoinder, delivered with breathtaking celerity, silenced the heckler.
Render	Provide a service or help E.g.: I called a voluntary organisation in my city and rendered my help on every Sunday.
Repast	A meal E.g.: Thanksgiving dinner is the most satisfying repast of the year.
Repentant	Feel or express regret or remorse E.g.: The young boy was repentant for what he had done to his friends.
Replete	Filled or well –supplied with E.g.: My old friends stay since one week depleted my stock, I have to replete the same.
Repose	A state of calm or peace E.g.: After a hard day at work, I took my repose on the couch.
Reprobate	A person who behaves in an immoral way E.g.: Our family priest, a person with high level of dignity, on Sundays you could always find the old reprobate in his easy chair with a six pack of cheap beer.
Repudiate	Refuse to accept or be associated with E.g.: I repudiate that argument.
Requite	Make appropriate return for (a favour or service); reward E.g.: The man's death penalty was requital given the number of murders he committed.
Rescind	Cancel a law or order or agreement E.g.: It looks bad for me to rescind a promotion I have already given you.
Restive	Unable to keep still or silent E.g.: It was difficult to tame the restive horse.
Reticent	Not revealing one's feelings or thoughts readily E.g.: He was reticent to tell the truth for fear he would be punished.
Reverent	Showing reverence E.g.: Yes, I can use the word reverence in a sentence. I just did.
Rhetoric	The art effective or persuasive speaking or writing E.g.: I got swayed by her rhetoric into donating all my savings to the charity.
Ribald	Humorous in a coarse way E.g.: The American radio host Howard Stern is known for his ribald humor.

Word	Meaning and sample usage
Rococo	A style of art, especially architecture and decorative art, that originated in France in the early 18th century and is marked by elaborate ornamentation, as with a profusion of scrolls, foliage, and animal forms E.g.: The detailed rococo of the houses paint and wall paper was amazing.
Rustic	Having to do with life in the country E.g.: My new bed has a rustic design in keeping with my country style
Sacrosanct	Seen as too important or too valuable to be changed or questioned E.g.: Julie did not let Buster enter her office because she considered it a sacrosanct space.
Sagacious	Having good judgment ; wise E.g.: Vivekananda is famous for his sagacity.
Salient	Most important E.g.: I have given salient details of the projects such that the young team can start their work.
Salubrious	Good for one's health E.g.: The salubrious mountain water has inspired many modern innovators to come up with the idea of selling fresh bottled-water in the market today.
Sanguine	Cheerfully confident about the future E.g.: The good news made her sanguine in mood and she was very excited.
Sardonic	Mocking E.g.: His sardonic smile and tone of voice told me he was jealous.
Satiate	Give(someone) as much or more than they want E.g.: I tried my level best to satiate my in-laws family at my wife's request.
Saunter	Walk in a slow, relaxed way E.g.: I like saunter through the park during winter season.
Savor	A characteristic taste, flavour, or smell, especially. a pleasant one E.g.: The resilient kitten was doing well in her new home. Barbara savored the delicious wedding cake.
Scintilla	A tiny trace or amount E.g.: The victim could be proven guilty in court because he hadn't left a scintilla of his crime.
Sedition	Action or speech urging rebellion against the authority of a state or a ruler E.g.: She was arrested after making a speech that the government considered to be seditious.

Word	Meaning and sample usage
Sedulous	Showing dedication and great care E.g.: You must be a sedulous student, if you want to understand vocabulary,
Sentient	Able to perceive and feel things E.g.: As they lifted the cover off the crashed saucer, the groans of the alien pilot showed that it was, indeed, a sentient being.
Seraphic	A type of angel associated with light and purity E.g.: He had a look of seraphic contentment on his face.
Sinecure	A job for which a holder is paid but which requires no or little work E.g.: Mowing that tiny lawn is a sinecure.
Sinewy	A piece of tough fibrous tissue that joins muscle to bone E.g.: Relying on running instead of weightlifting for his physical fitness, his build was more sinewy than bulky.
Skeptic	A person inclined to question or doubt all accepted opinions
Slake	Satisfy (a desire, thirst etc)
Sobriquet	A person's nickname E.g.: His mother always insisted in calling him William and not by his friends sobriquet of Bill.
Solecism	A grammatical mistake.
Sophistry	The use of false arguments E.g.: The senatorial candidate argued that his opponent was using sophistry in an effort to distort his plan for education reform.
Soporific	Causing drowsiness or sleep E.g.: The soporific lecture given by the professor caused everyone to yawn
Spartan	Lacking in comfort or luxury E.g.: Compared to the luxury suite at Hyderabad, the apartment in Vizag is rather spartan.
Specious	Misleading in appearance E.g.: The student had some good ideas, but usually used specious arguments in his written work.
Sporadic	Occurring at irregular intervals or only at few places E.g.: His sporadic movements in the lawn of Collector made police to suspect him.
Sportive	Playful; light-hearted
Spurious	False or fake E.g.: His spurious remark made me mad.
Stasis	A period or state where there is no change or development.

Word	Meaning and sample usage
Stentorian	(of a person's voice) loud and powerful E.g.: I could no longer stand my mother-in-law's stentorian voice.
Stigma	A mark or sign of disgrace E.g.: The Senator's racial remark will be a political stigma difficult to overlook in next year's election.
Stint	Restrict how much can someone have of (something) E.g.: In these hard economic times, you can cut corners here and there, but don't stint when it comes to your health.
Stipulate	Demand or specify as part of an agreement
Stolid	Calm, dependable and showing little emotion E.g.: John was stolid in his opinion on who was going to win the elections, I couldn't budge him.
Stratagem	A plan or scheme intended to outwit an opponent E.g.: The devised stratagem fooled the girl into tripping and falling in the ditch.
Sublime	Of very high quality and causing great admiration E.g.: if you make the room too hot you will sublime the ice in this glass.
Subside	Become less strong, violent or severe
Substantiate	Provide evidence to prove the truth of E.g.: The committee asked me to substantiate my argument.
Sully	Spoil the purity or cleanliness of E.g.: You don't want to sully something that was meant to be great.
Superfluous	Unnecessary because more than is needed E.g.: We can save hungry in the world by sharing our superfluous food.
Supplant	Take the place of E.g.: Ms. Gomez will supplant Mrs. Sprinkle for a couple of months.
Supposition	Assumption
Surfeit	An excess E.g.: Container showed great surfeit even after he ate 25 sandwiches.
Surly	Bad-tempered and unfriendly E.g.: I can't stand talking to surly teenagers.
Surrogate	A person who stands in for another in a role or office
Sybarite	A person who is very fond of luxury or pleasure E.g.: The lifelong sybarite was devastated when she had to live in a homeless shelter after she went bankrupt.

Word	Meaning and sample usage
Sycophant	A person who flatters someone important to try to gain favor with them
Symbiosis	A mutually beneficial relationship between different people or groups E.g.: Elephants and the birds that eat bugs off them use symbiosis, therefore, each benefit.
Syncopation	The act of syncopating; the contraction of a word by taking one or more letters or syllables from the middle; syncope
Taboo	Prohibited or restricted by custom
Tacit	Understood or implied without being stated E.g.: My wife and I are having tacit agreement about utensil cleaning.
Taciturn	Reserved or uncommunicative in speech; saying little E.g.: I was in a taciturn mood last night due to the bad day that I had.
Tactual	haptic: of or relating to or proceeding from the sense of touch.
Talon	1. A claw, esp. one belonging to a bird of prey 2. The shoulder of a bolt against which the key presses to slide it in a lock
Tangential	1. Of, relating to, or along a tangent 2. Diverging from a previous course or line; erratic
Tawdry	Showy but cheap and of poor quality E.g.: Those people have a lot of money, but no taste. Their gold toilet is so tawdry.
Tenacity	Doggedness: persistent determination E.g.: Although the job was difficult, he had the tenacity to stay with it until it was completed.
Terrestrial	An inhabitant of the earth
Tirade	A long, angry speech of criticism or accusation E.g.: The pope's trading speech made me mad.
Toady	Act in an obsequious way
Tome	A book, esp. a large, heavy, scholarly one E.g.: I opened the *Bhagavad Gita* and was instantly enthralled by the ancient tome.
Torpor	A state of physical or mental inactivity; lethargy E.g.: He fell into a deep torpor after his son's death.
Tractable	Easy to control or influence E.g.: My tractable nature made many people to use my help.
Transgression	The act of transgressing; the violation of a law or a duty or moral principle E.g.: He commited a transgression of the law, and has been sentenced to death.

Word	Meaning and sample usage
Transitory	Not permanent E.g.: His transitory attitude makes my helpless.
Trenchant	Vigorous or incisive in expression or style E.g.: His most trenchant criticism is reserved for the party leader, whom he describes as 'the most incompetent and ineffectual the party has known.
Truculence	Expressing bitter opposition; scathing E.g.: He was not popular with his boss because of his truculent attitude.
Turgid	Swollen and distended or congested E.g.: His sore knuckles and turgid lip led us to believe that Billy had been in a fight earlier that day.
Tyro	A beginner or novice E.g.: The girl next to my house is a tyro in Hindi.
Ubiquitous	Present, appearing or found everywhere E.g.: God is ubiquitous.
Umbrage	Take offence or become annoyed E.g.: I took umbrage to the large amount of bank fees I was charged and promptly changed my financial institution.
Unconscionable	Not right or reasonable E.g.: To Ram, it was unconscionable to use such underhanded tactics to get ahead.
Unequivocal	Leaving no doubt; clear in meaning E.g.: His unequivocal statements about general strike made us to reconsider his request.
Upbraid	Scold or criticise E.g.: The senator was upbraided by his peers for being found guilty of a felony. He could lose his senate seat because of this conviction.
Usury	The practice of lending money at usually high rates of interest E.g.: I charge high usury to James because he doesn't usually pay in time.
Vacillate	Waver between different opinions or actions E.g.: Unable to decide on which restaurant to enter, she vacillated for hours before opting for Italian.
Vacillation	Indecision in speech or action
Vacuous	Showing a lack of thought or intelligence E.g.: This movie is really completely vacuous.
Variegated	Exhibiting different colors, esp. as irregular patches or streaks E.g.: The variegated rainbow's colours were a beautiful sight after the storm.
Venerable	Greatly respected because of age, wisdom or character E.g.: He was a man of eternal self-sacrifice, and that is always venerable.

Word	Meaning and sample usage
Venerate	Regard with great respect
Veracious	Speaking or representing the truth E.g.: The jury's job is to determine whether or not the claims of the defendant are veracious.
Veracity	The quality of being true or accurate
Verbose	Using more words than are needed
Verdant	Green with grass or other lush vegetation E.g.: The verdant grass was damp with dew.
Vernacular	The language or a dialect spoken by the ordinary people of county or a religion E.g.: Even though she was a fluent Spanish speaker she had learned the language in Mexico and was completely thrown off by the vernacular of Barcelona.
Vestige	A remaining trace of something that once existed E.g.: The mummy had decomposed so badly that only vestiges of the cotton gauze could be seen.
Vex	Make annoyed or worried E.g.: My father's remarks on my spending vexed me a lot.
Viable	Capable of working successfully, feasible E.g.: Banks always sees viability of a project.
Vicissitude	Changes of circumstances or fortune E.g.: Given the vicissitudes of life, it's doubtful that I will live here to the end of my life
Vim	Energy; enthusiasm E.g.: At the age of 90 also, my father did not lost his vim.
Viscous	Having a thick, sticky consistency between solid and liquid
Vituperate	Bitter and abusive language E.g.: Each Sunday the stern old minister would climb into the pulpit and vituperate the congregation with a litany of vices.
Volatile	Easily evaporated at normal temperature
Voluble	Speaking easily and at length E.g.: The minister was very voluble, but there was little substance to what he said.
Wan	Pale and appearing ill or exhausted E.g.: Influenza had left her appearing wan, which she remedied, before the party, with a skillful application of makeup.
Wanton	(of a cruel or violent action)deliberate and unprovoked E.g.: She was too wanton in her ways. The old ladies of the town were shocked
Waver	Be indecisive E.g.: Realising the dangerous situation she was in, Agent Ashby wavered between deciding to jump or answering her cell.

Word	Meaning and sample usage
Whimsical	1. Playfully old-fashioned or fanciful 2. Showing sudden changes of directions or behavior E.g.: She has a whimsical sense of humor
Wily	Skilled at gaining an advantage especially deceitfully E.g.: The wily coyote built an elaborate trap to catch the speedy road runner.
Winsome	Attractive or appealing E.g.: The evil little trickster won me over with her winsome looks.
Wizened	Shrivelled or wrinkled with age E.g.: A wizened little man with gray hair.
Wraith	A ghost or ghostly image of someone especially one seen before or after their death E.g.: Evan looked out over the cemetery and was sure he saw a wraith rising from old Mr. Saibaba's grave.
Xenophobia	Strong dislike or fear of people from other countries
Yoke	A wooden crosspiece that is fastened over the necks of two animals and attached to the plow or cart that they are to pull
Zeal	Great energy or enthusiasm for a aim or a cause E.g.: I have completed this project with great zeal.
Zealot	An excessively enthusiastic or strict follower of a religion or policy E.g.: The animal activist was sentenced to six months in jail by the judge who described him as a extreme zealot who solely cared about causing mayhem and anarchy.
Zenith	The point in the sky directly overhead
Zephyr	A soft gentle breeze E.g.: We were delighted to have zephyr in our new house.

10.2 Numerical Reasoning and Interpretation

10.2.1 Sums and Sequence

Sequence: Sequence is an ordered set of numbers which could be defined as a function whose domain (x-values) consists of consecutive positive integers and the corresponding value is the range (y-values) of the sequence.

Term number: Term number is an ordered set of numbers which could be defined as a function whose domain (x-values) consists of consecutive positive integers.

Term: Term is the corresponding value (the range y-value) of the sequence.

Finite: Finite is a sequence with a limited number of terms.

Infinite: Infinite is a sequence with an unlimited number of terms.

Arithmetic sequence: It is a sequence in which a constant ***d*** (common difference) can be added to each term to get the next term.

Common difference: It is a constant difference, usually denoted as ***d.***

Geometric Sequence: It is a sequence in which a constant ***r*** can be multiplied by each term to get the next term.

Common ratio: It is a constant ratio, usually denoted by ***r.***

10.2.1.1 Arithmetic Sequence

$$t_n = t_1 + (n-1)d$$

Sum of an Arithmetic series:

$$S_n = \frac{n(t_1 + t_n)}{2}, \quad \text{or} \quad S_n = \frac{n}{2}[2t_1 + (n-1)d]$$

Arithmetic Mean: It is an average between 2 numbers.

$$\frac{(a+b)}{2}$$

10.2.1.2 Geometric Sequence

$$t_n = t_1 \bullet r^{n-1}$$

Geometric Mean: It is the term between two given terms of a geometric sequence as defined by the following formula:

$$\sqrt{ab}$$

Sum of a geometric series:

$$S_n = \frac{t_1(1-r^n)}{1-r}$$

Infinite Geometric Series

Theorem: An infinite geometric series is convergent and has a sum "***S***" if and only if its common ratio, ***r*** meets the following condition: $|r| < 1$

If our infinite series is convergent ($|r| < 1$), we can calculate its sum by the formula:

$$S = \frac{t_1}{1-r}$$

If the geometric series is defined as $\sum_{k=0}^{\infty} x^k$, for $-1 < x < 1$ then

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$$

A Geometric Series

- converges if $|r| < 1$, then the sum is as shown above.
- diverges if $|r| \geq 1$, then the sum becomes infinite.

■ **Example** Calculate

$$\sum_{n=3}^{\infty} 2\left(\frac{1}{2}\right)^{n-1}$$

■ **Answer:**

First term, $t_1 = 2(1/2)^{3-1} = 1/2$

Second term= $t_2 = 2(1/2)^{4-1} = 1/4$

Therefore, $r = t_2/t1 = 1/2$

Therefore, sum of the series=t1/(1–r)=1/2/(1–1/2)=1

■ **Example** Find the sum: 4 – 6 + 9 – 13.5 + ...

■ **Answer:** If we observe the series, r, common ratio becomes –1.5 which is >1. Thus, becomes infinite sum. Also, terms are increasing type.

■ **Example** In the first month, 15 000 m^3 of oil was produced from a well. Its production is dropping by 2.9% each month. How much oil will be produced over the next year? If the well works until it is dry, how much oil will be produced?

■ **Answer:** r = 1–0.029 = 0.971

n = 12

S12 = $a(1-r^n)/(1-r)$

= $15000^*(1-0.971^{12})/(1-0.971)$

= 153,892.33 m^3

To answer next question, we assume n = ∞. Thus total oil produced from the well

= $a/(1-r)$ =15000/(1–0.971)

= 517,241.38m^3

■ **Example** A ball is dropped from 38.28 m. After hitting the ground it raises to 60% of its previous height. What will be the total vertical distance ball traverses before it comes to stand still?

■ **Answer:** Here, r =0.6. Therefore total vertical distance it traverses before coming to stand still = 2*(38.28/(1–0.6))–38.28 = 153.12m

(Note that the ball hits and goes up. Thus, we are taking two times of the vertical heights. Also, we are subtracting 38.28 as ball is initially falling from that height.

■ **Example** Assume that a bread piece is cut into four equal parts, and give 3 people each one quarter of it. Then take the left over piece, and cut it into quarters, and give each of the same three people that part. Take the left over piece, and continue this process.

(a) Write a series representing the amount of bread each person gets after each cut.
(b) What will be the sum of the infinite sequence?
(c) Let's repeat the process with 4 students and 5 pieces.

■ **Answer:** What a person gets can be represented as:

$1/4 + 1/4^2 + 1/4^3 + \ldots$.

Here, r = 1/4

Sum= 1/4 /(1 – 1/4) = 1/3. That is, $1/3^{rd}$ of the bread is given to each person.

If 5 pieces are distributed among the 4 students, then the resulting series what a student gets can be represented as: $1+ 1/5 + 1/5^2 + \ldots$

Harmonic Series

Given by

$$\sum_{k=1}^{\infty} \frac{1}{k} = 1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\ldots$$

– Divergent series.

Hyperharmonic series (p-series)

Given by

$$\sum_{k=1}^{\infty} \frac{1}{k^p} = 1+\frac{1}{2^p}+\frac{1}{3^p}+\frac{1}{4^p}+\ldots$$

converges if $p > 1$ and diverges if $0 < p \le 1$.

Alternating Series

Alternating Series are series containing alternately positive and negative terms. General form:

$$\sum_{k=1}^{\infty} (-1)^{k+1} a_k = a_1 - a_2 + a_3 - a_4 + \ldots$$

$$\sum_{k=1}^{\infty} (-1)^{k} a_k = -a_1 + a_2 - a_3 + a_4 - \ldots$$

where the a_k's are assumed to be positive.

Alternating Series Test

An alternating series converges if both of the following conditions are satisfied:

(i) $a_1 > a_2 > a_3 > \ldots > a_n > \ldots$(terms decreasing in magnitude)
(ii) $\lim a_k = 0$

Note

$\sum_{k=1}^{\infty} (-1)^{k+1} \frac{1}{k}$ is the ***alternating harmonic series.*** It converges, whereas the ***harmonic series*** diverges.

10.2.1.3 Binomial Expansions and Powers of Binomials

Binomial expansion: $(a+b)^n$

The Binomial Theorem: for any binomial (a + b) and any whole number n, then $(a+b)^n =$

$${}_nC_0 a^n + {}_n C_1 a^{n-1}b + {}_n C_2 a^{n-2}b^2 + {}_n C_3 a^{n-3}b^3 + \ldots + {}_n C_n b^n$$

To find the *r*th term of a binomial expansion raised to the *n*th power, use the following formula:

$$\binom{n}{r-1}a^{(n-r+1)}b^{(r-1)}$$

Which is the same as:

$$({}_nC_{r-1})a^{(n-r+1)}b^{(r-1)}$$

10.2.2 Probability: An Introduction

10.2.2.1 Terminology

Population – The greater body of the sample you are taking.
Sample – The people or things that you are actually studying or performing on.
Random Sample – A group of subjects randomly chosen from a defined population.
Frequency Distribution – It can be defined as the number of times things occur in a sample.
Events
An event is defined as any outcome that can occur. There are two main categories of events: **Deterministic and Probabilistic.**
A **deterministic** event always has the same outcome and is predictable 100% of the time.

- Distance traveled = time x velocity
- The speed of light
- The sun rising in the east
- James Bond winning the fight without a scratch

A **probabilistic** event is an event for which the exact outcome is not predictable 100% of the time.

- The number of heads in ten tosses of a coin
- The winner of the World Series
- The number of games played in a World Series
- The number of defects in a batch of product

In a boxing match there may be three possible events.

- Fighter X wins
- Fighter Y wins
- Draw among X and Y

Basic Types of Events

- ***Mutually Exclusive Events:*** These are events that cannot occur at the same time. The cause of mutually exclusive events could be a force of nature or a man made law. Being twenty-five years old and also becoming president of the United States are mutually exclusive events because by law these two events cannot occur at the same time.
- ***Complementary Events:*** These are events that have two possible outcomes. The probability of event A plus the probability of A' equals one. P(A) + P(A') = 1. Any event A and its complementary event A' are mutually exclusive. Heads or tails in one toss of a coin are complementary events.
- ***Independent Events:*** These are two or more events for which the outcome of one does not affect the other. They are events that are not dependent on what occurred previously. Each toss of a fair coin is an independent event.
- ***Conditional Events:*** These are events that are dependent on what occurred previously. If five cards are drawn from a deck of fifty-two cards, the likelihood of the fifth card being an ace is dependent on the outcome of the first four cards.

10.2.2.2 Probability

Probability is defined as the chance that an event will happen or the likelihood that an event will happen. The definition of probability is

$$\text{Probability} = \frac{\text{Number of favorable events}}{\text{Number of total events}}$$

The favorable events are the events of interest. They are the events that the question is addressing. The total events are all possible events that can occur relevant to the question asked. In this definition, favorable has nothing to do with something being defective or non-defective.
What is the probability of a head occurring in one toss of a coin?
The number of favorable events is 1 (one head) and the number of total events is 2 (head or tail). In this case, the probability formula verifies what is obvious.

$$\text{Probability of a head} = \frac{\text{Number of favorable events}}{\text{Number of total events}} = \frac{1}{2}$$

Probability numbers always range from 0 to 1 in decimals or from 0 to 100 in percentages.

Instead of writing out the whole question, the following notation is used.

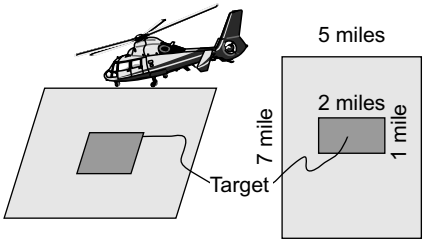
- What is the probability of event A occurring? = Probability (A) = P(A)
- What is the probability of events A and B occurring? = P(A and B) = P(A) and P(B)
- What is the probability of events A or B occurring? = P(A or B) = P(A) or P(B)

Probability in Terms of Areas
Probability may also be defined in terms of areas rather than the number of events.

$$\text{Probability} = \frac{\text{Favorable area}}{\text{Total area}}$$

■ **Example** A plane drops a parachutist at random on a seven by five mile field. The field contains a two by one mile

target as shown below. What is the probability that the parachutist will land in the target area? Assume that the parachutist drops randomly and does not steer the parachute.



$$\text{Probability of hitting target} = \frac{\text{Favorable area}}{\text{Total area}} = \frac{2}{35} = .057$$

Methods to determine probability values

There are three major methods used to determine probability values.

- ***Subjective Probability:*** This is a probability value based on the best available knowledge or maybe an educated guess. Examples are betting on horse races, selecting stocks or making product-marketing decisions.
- ***Priori Probability:*** This is a probability value that can be determined prior to any experimentation or trial. For example, the probability of obtaining a tail in tossing a coin once is fifty percent. The coin is not actually tossed to determine this probability. It is simply observed that there are two faces to the coin, one of which is tails and that heads and tails are equally likely.
- ***Empirical Probability:*** This is a probability value that is determined by experimentation. An example of this is a manufacturing process where after checking one hundred parts, five are found defective. If the sample of one hundred parts was representative of the total population, then the probability of finding a defective part is .05 (5/100). Here a question arise: How is it known that this sample is representative of the total population? If repeated trials average .05 defective, with little variation between trials, then it can be said that the empirical probability of a defective part is .05.

10.2.2.2.1 *Multiplication Theorem*

The multiplication theorem is used to answer the following questions:

- What is the probability of two or more events occurring either simultaneously or in succession?
- For two events A and B: What is the probability of event A and event B occurring?

The individual probability values are simply multiplied to arrive at the answer. The word "and" is the key word that indicates multiplication of the individual probabilities. The multiplication theorem is applicable only if the events are independent. It is not valid when dealing with conditional events. The product of two or more probability values yields the intersection or common area of the probabilities. The intersection can be illustrated via common areas in Venn diagrams. Mutually exclusive events do not have an intersection or common area in Venn diagrams. The probability of two or more mutually exclusive events is always zero.

For mutually exclusive events:

- P(A) and P(B) = 0

For independent events:

- Probability (A and B) = P(A) and P(B) = P(A) X P(B)

For multiple independent events, the multiplication formula is extended. The probability that five events A, B, C, D and E occur is

P(A) and P(B) and P(C) and P(D) and P(E) = P(A) x P(B) x P(C) x P(D) x P(E)

■ **Example** What is the probability of getting a raise and that the sun will shine tomorrow?

■ **Answer :**

Given: Probability of getting a raise = P(r) = .10

Probability of the sun shining = P(s) = .30

The events are independent.

P(raise) and P(sunshine) = P(r) x P(s) = .10 x .30 = .03 or 3%

10.2.2.2.2 *Addition Theorem*

The addition theorem is used to answer the following questions:

- What is the probability of one event or another event or both events occurring?
- What is the probability of event A or event B occurring?

The word "or" indicates addition of the individual probabilities. The answers to the above questions are different depending on whether the events are mutually exclusive or independent.

Mutually exclusive events do not have an intersection or common area. The individual probabilities are simply added to arrive at the answer. For mutually exclusive events:

- P(A or B) = P(A) or P(B) = P(A) + P(B)
- P(A or B or C or D) = P(A) + P(B) + P(C) + P(D)

For two independent events, the intersecting or common area must be subtracted or it will be included twice. (Refer to the Venn diagram in section 11.0).

Probability (A or B) = P(A) or P(B) = P(A) + P(B) – P(A X B)

For three independent events:

P(A or B or C) = P(A) + P(B) + P(C) – P(A X B) – P(A X C) – P(B X C) + P(A X B X C)

■ **Example** What is the probability of getting a raise or that the sun will shine tomorrow?

■ **Answer :**

Given: Probability of getting a raise = P(r) = .10

Probability of the sun shining = P(s) = .30

P(raise) or P(sunshine) = P(r) or P(s) = P(r or s)

P(r or s) = P(r) + P(s) – [P(r) x P(s)] = .10 + .30 – [.10 X .30] = .40 – .03 = .37 or 37%

The word "and" is associated with the multiplication theorem and the word "or" is associated with the addition theorem.

10.2.2.2.3 *Conditional Probability*

Conditional probability is defined as the probability of an event occurring if another has occurred or has been specified to occur simultaneously, and the outcome of the first event affects the probability of the second event. Conditional events are not independent.

The probability of B occurring given that A has already occurred is stated as P(B/A), where the symbol / means "given that."

The formulas for conditional probability are shown below. These are known as Bayes Formulas.

$$P(B/A) = \frac{P(A\&B)}{P(A)}$$

$$P(A/B) = \frac{P(A\&B)}{P(B)}$$

Since the two formulas have a common term P(A & B), they may be used together to solve many problems involving conditional probability.

Conditional events are not independent so P(A & B) is not equal to P(A) X P(B). From Bayes formulas:

P(A & B) = P(B/A) P(A)

P(A & B) = P(A/B) P(B)

10.2.2.2.4 *Bayes Theorem*

$$P(A/B) = \frac{P(A)P(B|A)}{P(B)}$$

■ **Example** A lot of fifteen items contains five defective items. Two items are drawn at random. What is the probability that the second item drawn will be defective?

■ **Answer:**

Let A = event that first item is defective

Let A′ = event that first item is good

Let B = event that second item is defective

The question stated in probability terms: what is P(B) = ?

P(A) = 5/15, P(A′) = 10/15

P(B) = P(A & B) or P(A′ & B) = P(first item defective & second item defective) or P(first item good & second item defective)

P(B) = P(B/A) P(A) or P(B/A′) P(A′)

P(B) = P(B/A) P(A) + P(B/A′) P(A′)

P(B) = (4/14)(5/15) + (5/14)(10/15)

P(B) = (20/210) + (50/210) = 70/210 = .333

■ **Example** It has been found that 10% of certain relays have bent covers and will not work. If 40% have bent covers, what is the probability that a relay with a bent cover will not work?

■ **Answer :**

A and B – bent AND will not work – 0.1

A = 0.4

Let A = event that relays have bent covers

Let B = event that relays will not work

Given: P(A & B) = .10, P(A) = .40

The first formula of the conditional probability formulas, Bayes formulas, gives the following solution:

$$P(B/A) = \frac{P(A \text{ and } B)}{P(A)} = \frac{.10}{.40} = .25$$

■ **Example** Given Probability of Ramu to reach time given that it rains is 60%.

Probability of Ramu to reach time given that it doesn't rain is 80%.

Probability of rain is 40%.

■ **Answer:**

A = Ramu on-time

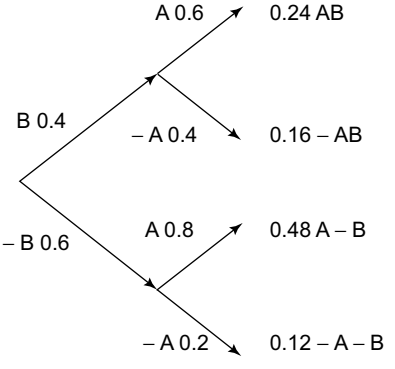
B = raining

P (A/B) = 60%

$P(A/\overline{B}) = 80\%$

What is the probability of Ramu going to school on time?

Tree Diagram



$B + A\overline{B}$ ← probability of Ramu going to school on time (mutually exclusive) 0.24 + 0.48 = 0.72

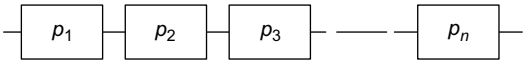
Given that he reached school on time, what is the probability that it rained?

$$P(B\backslash A) = \frac{P(B \cap A)}{P(A)} = \frac{.24}{.72} = \frac{1}{3}$$

10.2.2.2.5 *Reliability of a system: A practical Enginering Application*

An engineering system can be broken down into subsystems just containing elements in series or just containing elements in parallel. We find the reliability of each of these subsystems and find the reliability of whole system.

Series subsystem: in the diagram *pi* = probability that element *i* fails, so 1 – *pi* = probability that it does not fail.



The system only works if all *n* elements work.

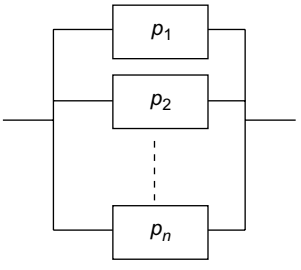
i.e., P(System does not fail) =

P(Element 1 doesn't fail and Element 2 doesn't fail and ... and Element *n* doesn't fail)

= P(Element 1 doesn't fail)P(Element 2 doesn't fail) ... P(Element *n* doesn't fail)

[Special multiplication rule; independence of failures]

$$= (1-p1)(1-p2) \ldots (1-pn) = \prod_{j=1}^{n}(1-p_j)$$



Parallel subsystem: the subsystem only fails if all the elements fail.

i.e., P(System fails) = P(Element 1 fails and Element 2 fails and ... and Element *n* fails)

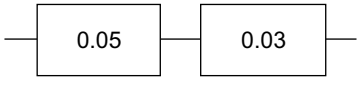
= P(Element 1 fails)P(Element 2 fails) ... P(Element *n* fails)

[Independence of failures]

$$= p1p2 \ldots pn = \prod_{j=1}^{n} p_j$$

■ **Example** What is the probability that the system does not fail in the next year?

■ **Answer:**



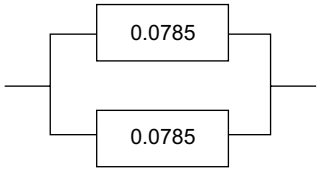
Subsystem 1:

P(Subsystem 1 doesn't fail) = (1 – 0.05)(1 – 0.03)

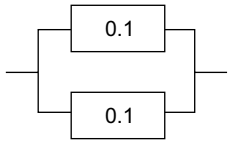
= 0.09215

P(Subsystem 1 fails)= 0.0785

Subsystem 2:

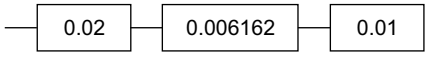


P(Subsystem 2 fails) = 0.0785 × 0.0785 = 0.006162



Subsystem 3:

P(Subsystem 3 fails) = 0.1 × 0.1 = 0.01



System (summarised):

P(System doesn't fail) = (1 – 0.02)(1 – 0.006162)(1 – 0.01)

= 0.964

■ **Example** Find P(System does not fail and component * does fail)

■ **Answer:**

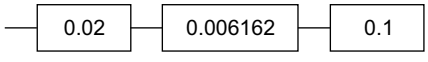
Let *B* = event that the system does not fail

Let *C* = event that component * does fail

We need to find P(*B* and *C*).

Now, P(*C*) = 0.1.

Also, P(*B* | *C*) = P(system does not fail given component * has failed); now if component * has failed, Subsystem 3 has probability of failing of 0.1 instead of 0.01, so that the final reliability diagram becomes:



$$P(B \mid C) = (1 - 0.02) \times (1 - 6.162 \times 10^{-3})(1 - 0.1)$$

$$= 0.8766$$

$$P(B \text{ and } C) = P(B \mid C)\, P(C) = 0.8766 \times 0.1$$

$$= 0.08766$$

■ **Example** Find P(Component * has failed | System has not failed)

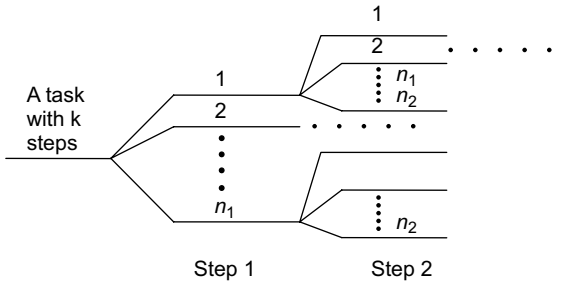
■ **Answer:** Using the previously introduced notation:

$$P(C \mid B) = \frac{P(B|C)P(C)}{P(B)} = \frac{0.8766 \times 0.1}{0.9642} = 0.091$$

10.2.2.2.6 *Counting Techniques: Permutations and Combinations*

Permutations and combinations are simply mathematical tools used for counting. In many cases, it may be cumbersome to count the number of favourable events or the number of total events when solving probability problems. Permutations and combinations help simplify the task.

For a task with k steps, if there are n_1 different ways to perform the first step, n_2 different ways to perform the second step, ..., and finally n_k different ways to perform the k-th step, then there are $n_1 \times n_2 \times ... \times n_k$ different ways to accomplish the task.



The proof is obvious by considering the tree diagram:

■ **Example** r persons are distributed in n rooms randomly. How many possible outcomes are there?

■ **Answer:** Consider all the r persons lined up. The first person can enter any of the n rooms, so can the second, third, ... r-th person. Hence by the fundamental principle of counting there are a total of n.n.n...n = nr possibilities. Note that the same problem can be messy if one tried to visualise the n rooms being lined up, so choosing the right perspective is often crucial to an efficient solution to the problem.

Permutations

A permutation is an arrangement of things, objects or events where the order is important. Telephone numbers are special permutations of the numerals 0 to 9 where each numeral may be used more than once. The order defines each unique telephone number.

In the following example, it is assumed that each object is unique and cannot be used more than once. The letters A, B, and C may be arranged in the following ways:

ABC BAC CAB

ACB BCA CBA

This is an ordered arrangement, because ABC is different than BCA. Since, the order of the letters makes a difference, each arrangement is a permutation. From the above example, It is concluded that there are six permutations that can be made from three objects. The general formula for permutations is

$$nP_r = \frac{n!}{(n-r)!}$$

n = The total objects to arrange

r = The number of objects taken from the total to be used in the arrangements

Permutations of n distinct objects is calculated using nPr

■ **Example** Using the permutation formula and the three letters A, B and C, how many permutations can be made using all three letters?

■ **Answer :**

$$_3P_3 = \frac{3!}{(3-3)!} = \frac{3\times 2}{0!} = \frac{6}{1} = 6$$

■ **Example** How many permutations can be made by using two out of the three letters ?

■ **Answer :**

$$_3P_2 = \frac{3!}{(3-2)!} = \frac{3\times 2}{1!} = \frac{6}{1} = 6$$

The permutations are

AB BA BC

AC CA CB

■ **Example** There are three different assembly operations to be performed in making a certain part. There are nine people working on the floor. How many different assembly crews can be formed?

■ **Answer :** This may be stated as the number of permutations that can be made from nine objects used three at a time.

$$_9P_3 = \frac{9!}{(9-3)!} = \frac{9!}{6!} = \frac{9\times 8\times 7\times 6!}{6!} = 504$$

■ **Example** How many ways are there to arrange 7 people in a row of 7 chairs?

■ **Answer:** 7! = 5040

■ **Example** Six men and four women took the same test, and all got different scores.

(a) How many different rankings are possible?

(b) How many different rankings are possible if men and women are ranked separately ?

■ **Answer:**

(a) 10! = 3628800

(b) 6!4! = 17280

Permutations of n objects, some of which are indistinguishable

Suppose we wish to permute *n* objects, but n_1 of them are alike (i.e. indistinguishable from each other), n_2 are alike of another kind, ..., finally n_k are alike of a *k*-th kind. For example, among the 6 letters PEPPER, there are 3 indistinguishable Ps and 2 indistinguishable

■ **Example** How many different permutations are there in such cases?

■ **Answer:** Let the correct answer be denoted by c. The trick is to pretend we really had all distinguishable Ps (say P_1, P_2, P_3) and E_s (E_1 and E_2, say), then, if there are c arrangements in the indistinguishable case, each arrangement must be further arranged in 3!2! different ways since everything is distinguishable, resulting in a total number of arrangements c 3! 2! which must equal (number of ways to permute 6 different objects) = 6!, hence c = $\frac{6!}{3!\,2!}$. By the same reasoning, there are

$$\frac{n!}{n_1!n_2!.....n_k!} = \text{ways to permute } n \text{ objects, with } n_i \text{ being alike of the } i\text{-th kind}$$

■ **Example** How many distinct permutations can you form from all the letters of each word:

(i) FAST,

(ii) NAKAMURA,

(iii) PROBABILISTICALLY?

■ **Answer:**

(i) 4! = 24 since all 4 letters are distinct

(ii) 8!/3! = 6720 since there are 8 letters among which 3 A's are indistinguishable

(iii) 17!/2!/2!/3!/3! = 2,470,051,584,000 since there are 17 letters among which 2 B's, 2 A's, 3 I's and 3 L's are identical.

Combinations

A combination is a grouping or arrangement of objects where the order does not make a difference. The arrangement of the letters ABC is the same as BCA. The number of combinations that can be made by using three letters, three at a time, is one. This can be expanded to state that the number of combinations that can be made by using n letters, n at a time, is one. A hand of five cards consisting of a Jack, a Queen, a King, and two Aces is the same as a Queen, two Aces, a Jack and a King. The order in which the cards were received makes no difference. There is only one combination that can be made by using five cards, five at a time. The formula for combinations is

$$_nC_r = \frac{_nP_r}{r!} = \frac{n!}{(n-r)!\,r!}$$

n = Total objects to arrange

r = Number of objects taken from the total to be used in the arrangements

The symbol for number of combinations is often shown as

$$\binom{n}{r} \text{ where } \binom{n}{r} = {}_nC_r$$

When the symbol appears in a formula, the number of combinations is to be computed using the combination formula.

$$\binom{n}{r} = \frac{n!}{(n-r)!\,r!}$$

■ **Example** From the three letters A, B and C, how many combinations can be made by using two out of the three letters?

$$\binom{3}{2} = \frac{3!}{(3-2)!\,2!} = \frac{3\times2\times1}{2\times1\times1} = 3$$

The combinations are

AB AC BC

BA is the same as AB. We did not consider both in our counting.

CA is the same as AC. We did not consider both in our counting.

CB is the same as BC. We did not consider both in our counting.

■ **Example** Ten parts have been manufactured. Two parts are to be inspected for a critical dimension. How many different sample arrangements can be made?

■ **Answer:** If the parts are labeled 1 to 10, then parts 1 and 5 make one arrangement, parts 3 and 7 make another, 6 and 8 another, etc. The listing of the various arrangements can be completed and total arrangements counted. The combination formula can perform this task and save a considerable amount of time.

The total arrangements or combinations that can be made:

$$\binom{10}{2} = \frac{10!}{(10-2)!\,2!} = \frac{10!}{8!\times2!} = \frac{10\times9\times8!}{8!\times2\times1} = 45$$

■ **Example** It is proposed to have a committee that consists of 3 male professors and 2 female professors from 7 male professors and 5 female professors. How many ways are possible?

■ **Answer:** First, to choose 3 men out of 7 there are *C*(7, 3) ways. Next, to pick 2 women out of 5 there are *C*(5, 2) ways. So this two-step process has (7.6.5)/(3.2.1) x (5.4)/(2.1) ways of being accomplished.

■ **Example** In surveying, the three-point resection method can be used to determine the coordinates of an unknown point by observing from it three stations with known co-ordinates (Known as Triangulation which is used GPS also). The unknown coordinates can then be calculated by a formula (Tienstra's). For better accuracy and to guard against mistakes, a surveyor plans to observe 5 known stations, apply Tienstra's formula using 3 stations at a time until he exhausts all possible combinations, and average the results if they all agree well. How many sets of coordinates will he need to calculate before averaging?

■ **Answer:** To pick 3 out of 5 available stations for each calculation, there are a total of 5!/3!/2! = 10 ways.

■ **Example** A box contains 11 balls, numbered 1, 2, 3, …, 11. If 6 balls are drawn simultaneously at random, what is the probability that the sum of the numbers on the balls drawn is odd?

■ **Answer:** Number of ways to choose 6 numbers out of 11 = $\binom{11}{6} = 462$.

We have to count the number of these $\binom{11}{6}$ combinations that sum to an odd number. There are 6 odd numbers to pick from and 5 even ones. If the sum is to be odd, an odd number of odd numbers must be chosen. Thus, either 1, 3 or 5 odd numbers is chosen. Add up the number of ways to choose 6 numbers out of 11 in these three separate cases:

Number of ways to choose 1 odd and 5 even = $\binom{6}{1}\times\binom{5}{5} = 6$,

we multiply these two since each choice of a set of odd numbers can be paired up with each choice of a set of even numbers. Using the same logic, we have:

Number of ways to choose 3 odd and 3 even = $\binom{6}{3}\times\binom{5}{3} = 200$, and

Number of ways to choose 5 odd and 1 even = $\binom{6}{5}\times\binom{5}{1} = 30$,

Thus, the desired probability is $\frac{6+200+30}{462} = \frac{118}{231}$

■ **Example** A box contains three shiny pennies and 4 dull pennies. One by one, pennies are drawn at random from the box and not replaced. What is the probability that it will take more than four draws until the third shiny penny appears?

■ **Answer:** The sample space is $\frac{7!}{4!3!} = 35$, the number of permutations of the 3 shiny pennies and 4 dull pennies. Of these 35, we want to find how many of them require more than four draws to pull the third shiny penny. It's easier to use the subtraction principle here and simply count the number of ways in which the 3 shiny pennies all get pulled in four or less turns. We can enumerate these (SSSDDDD, SSDSDDD, SDSSDDD, DSSSDDD) or reason that we must choose 3 of the first four slots for shiny pennies, fixing the last three slots do dull pennies. We can do this in $\binom{4}{3} = 4$ ways. Thus, the probability it will take more than 4 draws to pull the last shiny penny is $\frac{35-4}{35} = \frac{31}{35}$.

■ **Example** Six distinct integers are picked from the set {1, 2, 3,…, 10}. What is the probability that among those selected, the second smallest is 3?

■ **Answer:** There are $\binom{10}{6} = 210$ ways to pick 6 integers out of 10. Of these, we must count how many of these combinations of 6 have 3 as the second smallest value. In order for this to occur, we must choose 1 value from the set {1,2} and 4 values from the set {4, 5, 6, 7, 8, 9, 10}. This can be done in $\binom{2}{1}\times\binom{7}{4} = 70$ ways. (We multiply because each choice from the first set can be paired up with any of the choices from the second set.) Thus, the desired probability is $\frac{70}{210} = \frac{1}{3}$.

■ **Example** A non-zero digit is chosen in such a way that the probability of choosing digit d is $\log_{10}((d+1)/d)$. The probability that 2 is chosen is exactly ½ the probability that the digit chosen is in which of the following sets?

(A) {2,3} (B) {3,4}
(C) {5,6,7,8,9} (D) {4,5,6,7,8}
(E) {4,5,6,7,8,9}

■ **Answer:** The probability that 2 is chosen is = $\log_{10}\frac{3}{2}$.

Thus, the set we must pick must have a probability of $2\log_{10}\frac{3}{2} = \log_{10}\left(\frac{3}{2}\right)^2 = \log_{10}\frac{9}{4}$ of having a number chosen from it. Given a set {a, a+1, a+2, ..., b} the probability of choosing a digit from that set is $\sum_{k=a}^{b}(\log_{10}(k+1) - \log_{10} k)$

$= \log_{10}(b+1) - \log_{10} a = \log_{10}\frac{b+1}{a}$, applying a telescoping sum and the log difference rule.

Setting b+1 = 9 and a = 4, we find that a=4, b=8 and the correct choice is D.

■ **Example** Three balls marked 1, 2, and 3 are placed in an urn. One ball is drawn, its number recorded, and then the ball is returned to the urn. This process is repeated and then repeated once more, and each ball is equally likely to be drawn on each occasion. If the sum of the numbers recorded is 6, what is the probability that the ball numbered 2 was drawn all three times?

■ **Answer:** Let x, y, and z represent the values of the first, second and third ball pulled from the urn, respectively. The sample space is all ordered triplets (x,y,z) such that x+y+z=6. These ordered triplets are (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1) and (2, 2, 2). Of these 7 possibilities, only

1 corresponds to drawing 2 all three times, thus the desired probability is $\frac{1}{7}$.

■ **Example** Let S be the set of permutations of the sequence 1,2,3,4,5 for which the first term is NOT 1. A permutation is chosen randomly from S. What is the probability that the second term is two?

■ **Answer:** There are 5! = 120 permutations of 1, 2, 3, 4, 5 total. Of these, there are 4! = 24 where 1 is in the first position. (We can determine this by fixing 1 in the first position and then observing that there are 4! ways to permute 2, 3, 4, 5 amongst the remaining slots.) Thus, we have 5! - 4! = 96 permutations in the sample space.

Of these 96 permutations we must count how many of them have 2 in the second position. For the first position, we have three choices, 3, 4 or 5. Then for the remaining 3 positions, we are free to permute the remaining three items in 3! = 6 ways. Thus, there are a total of 3x3! = 18 permutations in our sample space where 2 is in the second position. The desired probability is $\frac{18}{96} = \frac{3}{16}$.

■ **Example** First a is chosen at random from the set {1, 2, 3,…, 100} and then b is chosen at random from the same set. What is the probability that the units digit of 3^a+7^b has a units digit of 8?

■ **Answer:** Let's make charts for the possible units digits of 3^a and 7^b in terms of a and b.

N	units digit of 3^n	units digit of 7^n
1	3	7
2	9	9
3	7	3
4	1	1
5	3	7

We can see that the unit's digit for each column repeats every 4 values. Thus, 3, 9, 7 and 1 appear exactly 25 times as units digits in the list 3^1, 3^2, ..., 3^{100}, and the list 7^1 7^2 ..., 7^{100}. In essence each has a probability of 1/4 of occurring as the units digit of 3^a and 7^b. Let (x, y) be the ordered pair where x is the units digit of 3^a and y is the units digit of 7^b. The probability of getting the each of the ordered pairs (3, 7), (3, 9), (3, 3), (3, 1), (9, 7), (9, 9), (9, 3), (9, 1), (7, 7), (7, 9), (7, 3), (7, 1), (1, 7), (1, 9), (1, 3), and (1,1) is 1/16. Of these, three sum to a units digit of 8: (1, 7), (7, 1) and (9, 9). Thus, the desired probability is $\frac{3}{16}$.

■ **Example** An unbiased die marked 1, 2, 2, 3, 3, 3 is rolled three times. What is the probability of getting a total score of 4?

■ **Answer:** Let (x, y, z) be the ordered triplet where x is the value of the first roll, y the value of the second roll and z the value of the third roll. The possible ordered triplets that correspond to a total score of 4 are (2, 1, 1), (1, 2, 1), and (1, 1, 2). The probability of achieving each of these is $\frac{2}{6}\times\frac{1}{6}\times\frac{1}{6} = \frac{1}{108}$, since the probability of rolling a 2 on any given roll is $\frac{2}{6}$, whereas the probability of rolling a 1 on any given roll is $\frac{1}{6}$, and each roll is independent of the others. Since the three ordered pairs are mutually exclusive the total probability is the sum of these three probabilities which is $3\times\frac{1}{108} = \frac{1}{36}$.

■ **Example** If A and B are events and p(A) = 8/15, p(A ⋂ B) = 1/3, p(A | B) = 4/7 calculate p(B), p(B|A) and p(B | ~A), where ~A is the complement of the event A. Are A and B independent? Mutually exclusive?

■ **Answer:**

$$p(A \mid B) = \frac{4}{7} = \frac{p(A \cap B)}{p(B)} = \frac{1/3}{p(B)}, \text{ thus } p(B) = \frac{1/3}{4/7} = \frac{7}{12}.$$

$$p(B \mid A) = \frac{p(A \cap B)}{p(A)} = \frac{1/3}{8/15} = \frac{5}{8}$$

$$p(B \mid \sim A) = \frac{p(\sim A \cap B)}{p(\sim A)} = \frac{p(B) - P(A \cap B)}{1 - p(A)}$$

$$= \frac{7/12 - 1/3}{1 - 8/15} = \frac{1/4}{7/15} = \frac{15}{28}$$

A and B aren't independent, since p(B | A) ≠ p(B) and p(A | B) ≠ p(A).
A and B aren't mutually exclusive since p(A ⋂ B) ≠ 0.

■ **Example** Edit distance or time warping distance between two strings *a* and *b* of equal length to be the minimum number of letter substitutions needed to make in string *a* in order to obtain string *b*. For example, the edit distance between the strings "HELLO" and "JELLO" is 1, since only 'J' must be substituted for 'H' in order to obtain the second word from the first. Also, the edit distance between "HELLO" and "JELLY" is two, since in addition to the first substitution described, a 'Y' must be substituted for 'O'. For all three parts of this question, assume that all strings are *case insensitive*.

(a) How many alphabetic strings of length 5 have an edit distance of 1 from the string "HELLO"?

(b) Let *n* be your answer to question a. One may argue that the number of alphabetic strings of length 5 that

have an edit distance of 2 from the string "HELLO" is *n2*. In essence, one would argue that in order to find a string with an edit distance of 2 away from "HELLO", one must change one letter at random, and then repeat that operation on the intermediate string. (i.e. "HELLO" → "JELLO" → "JELLY") To count how many ways this can be done, since the first operation is independent of the second, we would simply use the multiplication principle and multiply the number of ways the first operation can be done by the number of ways the second operation can be done. Both of these values are *n*, leading to a final answer of *n2*. What is the flaw with this argument?

(c) Determine the actual number of alphabetic strings of length 5 with an edit distance of 2 from the string "HELLO".

■ **Answer:**

(a) We can choose one of the five locations in the string to make a change. For each of these five choices, we can change the letter to 25 other options. (It's not 26 since we can't change the letter to itself). Thus, using the multiplication principle, there are 5×25 = 125 total strings with an edit distance of 1 from "HELLO".

(b) Not all distinct ordered pairs of operations lead to distinct strings. Consider the two following distinct ordered pairs of operations:

"HELLO" → "JELLO" → "JELLY" and

"HELLO" → "HELLY" → "JELLY"

In the n^2 count, both of these two operations would be counted for two different words with an edit distance of 2 from "HELLO". But, as we can see, they should really only be counted as one word.

One may actually say then, that we can simply divide n^2 by 2 to obtain our answer. But this also, is faulty. This doesn't take into account, the following type of ordered pair of operations:

"HELLO" → "JELLO" → "HELLO" or

"HELLO" → "JELLO" → "MELLO"

In spite of the fact that both operations are distinct, they don't result in a final string that is actually an edit distance of 2 from "HELLO".

(c) Out of the 5 characters, we must choose exactly 2 to edit. This can be done in $\binom{5}{2} = 10$ ways, since we are choosing 2 characters out of 5. For each of the two characters we change, we have exactly 25 possible choices. The choice of one character is completely independent of the other, so, we can change the characters in 25×25 = 625 ways. Using the multiplication principle, multiply the choices of pairs of characters to change with the number of ways to change them to obtain 10×625 = 6250 as the final answer.

■ **Example**

(a) How many four digit numbers (between 1000 and 9999, inclusive) do NOT contain any repeating digits?

(b) A number is defined as *ascending* if each of its digits are in increasing numerical order. For example, 256 and 1278 are *ascending* numbers, but 1344 and 2687 are not. How many four digit numbers (between 1000 and 9999, inclusive) are *ascending*?

(c) A number is defined as *descending* if each of its digits are in decreasing numerical order. For example, 652 and 8721 are *descending* numbers, but 4431 and 7862 are not. How many four digit numbers (between 1000 and 9999, inclusive) are *descending*?

■ **Answer:**

(a) We can only choose 9 values for the first digit as 0 is not permissible in the first digits place. Also, 9 digits for the second digit place which includes 0 and excluding the digit used in the first digits place. Similarly, 8 digits can be used (excluding first and second place) for the third digit place; and 7 digits (excluding first, second, and third digit places) for the last digit using similar reasoning. Thus, the final answer is 9(9)(8)(7) = 4536.

(b) Since the first digit cannot be 0, none of the digits can be 0. For each combination of four digits from the set {1, 2, 3, 4, 5, 6, 7, 8, 9} we can create exactly one ascending number. Thus, the total number of ascending numbers is $\binom{9}{4} = 126$.

(c) A descending number can contain 0. Thus, for each combination of four digits from the set {0, 1, 2, 3, 4, 5, 6, 7 ,8 9} we can create exactly one descending number. Thus, the total number of descending numbers is $\binom{10}{4} = 210$.

■ **Example**

(a) A class has 8 girls and 4 boys. If the class contains 6 sets of identical twins, where each child is indistinguishable from their twin, how many different ways can the class line up to go to recess? (Do not count two configurations as distinct if the only difference between the two is twins swapping spots in line.)

(b) Unfortunately, each day when the class (the same class with 6 pairs of twins described in part A) lines up to go to recess (this is done once a day), if two boys are adjacent to each other in line, they always cause problems. But, the kids also cause problems if they are

ever lined up the same way on two separate days. How many possible orders can the class line up in without having any problems?

■ **Answer:**

(a) All that is important here is that there are 6 pairs of twins, we are arranging 12 people in line, where 6 pairs are indistinguishable. This the exact same question as computing the number of permutations of a 12 letter word comprised of 6 pairs of letters. Using the formula for permutations with repetitions, we find the answer to be $12!/(2!)^6$.

(b) First we will consider the possible orders of boys and girls, and then we will consider the different valid permutations while only interchanging boys with boys and girls with girls.

Consider laying out the girls with gaps in between as follows:

__ G __ G __ G __ G __ G __ G __ G __ G __

We can choose any 4 of the 9 gap(___) locations for the boys. This can be done in ${}_9C_4$ ways.

Now, let us consider the total number of orders for the boys for each of these in ${}_9C_4$ arrangements. There are 4 boys, but 2 pairs of twins. Using the formula for permutations with repetition, we get $4!/(2!)^2$ orders. Now, consider the number of ways the girls can be permuted for each of the ${}_9C_4$ arrangements discussed above. Here have 4 pairs of twins. Applying the same formula, we get $8!/(2!)^4$ permutations.

Multiply these three terms to get the final answer $({}_9C_4)4!8!/2^6$.

■ **Example** Consider four boxes (R1, R2, R3, and R4) containing marbles. The marbles are either red, or white, or blue but are otherwise indistinguishable.

R1: Has 10 red, 10 white, and 10 blue marbles.

R2: Has 10 red marbles.

R3: Has 10 white marbles.

R4: Has 10 blue marbles.

Marbles are selected from the boxes and laid out in a row. (Thus, the order in which the marbles are chosen makes a difference. For example, RWWWBRR is a different order than RRWWWB.) How many linear arrangements can be created under the following circumstances?

(a) Seven marbles are chosen, all from R1.

(b) Ten marbles are chosen. The first marble chosen is from R1. Then zero or more marbles are chosen from R2, followed by zero or more marbles form R3, followed by zero or more marbles from R4. The total number of marbles chosen from these last three receptacles must be nine. (For example, WRRRBBBBBB is permissible, while, WRRWRBBBBB is not.)

■ **Answer:**

(a) There are 3 choices for each of 7 marbles. Using the multiplication principle, that is $\mathbf{3^7}$ possible orders.

(b) There are three choices for the first marble.

The following 9 choices are chosen out of three bins, in that order.

Let r be the number of marbles chosen from R2.

Let w be the number of marbles chosen from R3.

Let b be the number of marbles chosen from R4.

We must find the total number of solutions to the equation r + w + b = 9, where r, w, and b are all non-negative integers.

We are essentially distributing 9 marbles amongst 3 bins. This can be done in $\binom{9+3-1}{3-1}=\binom{11}{2}=55$ ways.

Using the product rule, we find a total of 3(55) = 165 permissible orders.

■ **Example** Students A, B, C, D, E, F, G, H, I, and J must sit in ten chairs lined up in a row. Answer the following questions based on the restrictions given below. (Note that each part is independent of the others, thus no restriction given in part a applies to the rest of the parts, etc.)

(a) How many ways can the students sit if the two students on the ends of the row have to be vowel-named students?

(b) How many ways can the students sit if no two students with vowel names can sit adjacent to each other?

(c) Given that students A, B, C, and D are male, and that the rest of the students are female, how many ways can the students can be arranged such that the average number of females adjacent to each male is 0.25? (Note: to determine the average number of females each male is adjacent to, sum up the total number of females adjacent to each male and then divide by the total number of males. For example, in the arrangement AEBFCDGHIJ, each male is adjacent to 1.25 females, on average.)

■ **Answer:**

(a) There are three choices for the student on the left, and then 2 choices for the student on the right. Following those two choices, we can arrange the rest of the 8 students left in 8! ways. Thus, the total number of ways the students can sit is (3)(2)(8!).

(b) Place all seven consonants like so (C designates an arbitrary consonant):

__ C __ C __ C __ C __ C __ C __ C __

Now, the empty slots (___) represent possible locations for the vowels. There are P(8,3) = (8)(7)(6) ways

to place the vowels. The 7 consonants can be ordered in 7! ways. Thus, there are (8)(7)(6)(7!) ways the students can sit without any vowel-named students sitting next to each other.

(c) Notice that the only ways in which the average number of females adjacent to males is 0.25 is when all four males are at the left or right end of the row of chairs. If this isn't the case, then more than one female will be adjacent to a male. If this occurs, then the average will be at least 0.5. Since the males and female can sit an any arrangement amongst themselves, for both cases, they can sit in (4!)(6!) ways. Adding both the possibilities (males to the left, males to the right), the total number of arrangements desired is (2)(4!)(6!).

■ **Example** Consider a language with the following characteristics:

(1) The alphabet is composed of three symbols: a, b, and c.

(2) Each word in the language is a concatenation of four of these symbols.

(3) Each command in the language is composed of three words.

(a) How many distinct commands can be created if words in a single command can be repeated and two commands are identical only if the three words AND the order in which the words appear are identical? (Thus, the commands "aaca baaa aaca" and "baaa aaca aaca" are two DIFFERENT commands.)

(b) How many distinct commands can be created if word position does not affect meaning and a given word may appear at most once in a single command? (Thus, "abca bbac abbb" and "bbac abbb abca" should NOT count as different commands, and "aaca baaa aaca" is an INVALID command.)

■ **Answer:**

(a) Total of 12 symbols in a command. For each of these symbols, we have 3 choices without any restrictions. These choices are independent of one another, so the total number of commands we have is 3^{12}.

(b) Since we are not allowed to repeat words and word order doesn't matter, we are essentially choosing three words out of a possible number of words. Thus, we must first figure out the possible number of words. There are three choices for each of four symbols, using the multiplication principle as we did in part a, we have $3^4 = 81$ possible words. Of these, we can choose three to make a valid command. Thus, the total number of possible commands here is ${}_{81}C_3 = (81)(80)(79)/6 = 85320$

■ **Example** Consider six-digit numbers with all distinct digits that do NOT start with 0. Answer the following questions about these numbers. Leave the answer in factorial form.

(a) How many such numbers are there?

(b) How many of these numbers contain a 3 but not 6?

(c) How many of these numbers contain either 3 or 6 (or both)?

■ **Answer:**

(a) There are 9 choices for the first digit, and then 9 choices for the second digit (0 has been added as a choice), 8 for the third, 7 for the fourth, 6 for the fifth, and 5 for the sixth. Total = (9)(9)(8)(7)(6)(5) = 9(9!)/4! = 136080.

(b) We need to separate the counting into two categories

(1) 3 is the first digit

(2) 3 is NOT the first digit

For the first category, we have one choice for the first digit, followed by 8 choices for the second digit (not 3 or 6), 7 choices for the third digit, 6 choices for the fourth digit, 5 choices for the fifth digit and 4 choices for the sixth digit.

Total = (8)(7)(6)(5)(4) = 8!/3!

For the second category, we have 7 choices for the first digit (not 0, 3, or 6), now we must guarantee that a 3 is picked. There are five PLACES to place the 3. For the remaining 4 digits, we have 7 choices, 6 choices, 5 choices and 4 choices, respectively for each of these. (To see this, imagine the 3 was placed 2nd. Then for the third digit you could choose any number except for the first digit, 3 and 6. Similarly, no matter where the 3 is placed, you always have 7 choices for the next placed digit, then 6, etc.)

Total = (7)(5)(7)(6)(5)(4) = 35(7!)/3!

The total of both of these categories is 8!/3! + 35(7!)/3! = 36120

(c) Count the number of numbers that contain neither:

There are 7 choices for the first digit(not 0, 3 or 6), 7 choices for the second digit, 6 choices for the third digit, 5 choices for the fourth digit, 4 choices for the fifth digit and 3 choices for the sixth digit.

Total = (7)(7)(6)(5)(4)(3) = 7(7!)/2!

Now, the answer to the question given is the value above subtracted from the answer in part a. Thus, this answer is 9(9!)/4! - 7(7!)/2! = 118440.

■ **Example**

(a) How many distinguishable ways are there to rearrange the letters in the word COMBINATORICS?

(b) How many distinguishable arrangements are possible with the restriction that all vowels ("A", "I", "O") are always grouped together to form a contiguous block?

(c) How many distinguishable arrangements are possible with the restriction that all vowels are alphabetically ordered and all consonants are alphabetically ordered? For example: BACICINOONRST is one such arrangement.

■ **Answer:**

(a) There are 13 letters in the word COMBINATORICS, including three duplicates, two C's, two O's and two I's. So, the total number of arrangements is $13!/(2!)^3$.

(b) If all five vowels are consecutive, they form a single block. Then first we need to count permutations of the consonants and one block of vowels. Given eight consonants with one duplicate (two C's), we have 9!/2!. But every arrangement of consonants and the block of vowels can be combined with any permutation of vowels inside the block. For five vowels including two duplicates we have $5!/(2!)^2$ possible permutations inside the block. Then by the product rule we get the answer: $(9!\times5!)/(2!)^3$.

(c) Any arrangement is completely defined by specifying which 5 of 13 positions should be occupied by vowels (or equivalently which 8 out of 13 should be occupied by consonants). So we just need to count the number of ways to select 5 positions out of 13 (or equivalently 8 positions out of 13), that is 13!/(8!5!). Given any such selection, both consonants and vowels are distributed alphabetically into assigned slots.

■ **Example** How many 6-letter words can be formed by ordering the letters *ABCDEF* if *A* appears before *C* and *E* appears before *C*?

■ **Answer:** Under given restrictions there are two possible arrangements for letters *A, C* and *E* between themselves: either *A* appears before *E* , or *E* before *A*, i.e. *AEC* or *EAC*, so we have two choices for this task. After that we can choose 3 slots to place letters *A, C* and *E* out of 6 possible slots in a 6-letter word. If the order of *A, C* and *E* is fixed, we count *C* (6, 3) selections. After we fill 3 slots with the letters *A, C* and *E*, we can make 3! permutations of the letters *B*, *D* and *F* using remaining 3 slots. By the product rule the total number of orderings will be $2\cdot C(6, 3)\cdot 3! = 2\cdot6\cdot5\cdot4 = 240$.

■ **Example**

(a) How many permutations of the word FOUNDATION are there?

(b) A valid password is 5 letters long and uses a selection of the letters in the word FOUNDATION. (Thus, a password may have at most 2 N's, at most 2 O's, and at most 1 copy of each of the other letters {A, D, F, I, U, T} in FOUNDATION). How many valid passwords are there?

■ **Answer:**

(a) 10!/(2!2!), since there are 10 letters total with two O's and two N's.

(b) Split up the counting into three separate categories:

(1) Passwords with 5 distinct letters

(2) Passwords with 4 distinct letters

(3) Passwords with 3 distinct letters

(1) We have 8 distinct letters to choose from and we are choosing 5.

There are P(8,5) = 8!/3! ways to do this.

(2) We first choose either two Ns or two Os. This can be done in 2 ways. Then we choose three distinct letters out of the 7 remaining. We can choose the three letters in C(7,3) ways. Thus, we have C(7,3) x2 = 70 ways to choose our letters. Each of these choices give rise to 5!/2! = 60 permutations.

(3) We have to choose all Ns and Os, which leaves us one choice out of the remaining 6 letters. We can choose this letter is 6 ways. For each of these choices, we can make 5!/(2!2!) = 30 permutations. So there is a total of 180 permutations of this kind to count.

Adding up, we get the total number of valid passwords to be:

$8!/3! + 70 \times 60 + 180 = 6720 + 4200 + 180 = 11100.$

■ **Example** An ice cream shop lets its customers create their orders. Each customer can choose up to four scoops of ice cream from 10 different flavours. In addition, they can add any combination of the 7 toppings to their ice cream. (Note: Please leave your answer in factorials, combinations, and powers.)

(a) If a customer is limited to at most two scoops of the same flavour, how many possible orders with exactly 4 scoops and up to 5 toppings can the customer make? (Assume each order has at least one topping.)

(b) Suzanne wants to make 7 separate orders for ice cream. Each order will have exactly 1 scoop and 1 topping. If no flavor or topping is requested more than once, how many combinations of orders can Suzanne make?

■ **Answer:**

(a) If we ignore toppings initially, we have a problem of combinations with repetition. We are choosing 4 items from 10 possible items, allowing for repetition. This can be done in C(4+10-1,4) = 715. But, here we are counting choices that have 3 and 4 scoops of the same flavour. We need to subtract these out. So, our next sub-problem becomes to count the number of ways we can order exactly 4 scoops with one flavour repeated at least 3 times. Since only one flavour can be repeated at

least 3 times, pick this flavour. There are 10 choices for it. Go ahead and pick 3 scoops of this flavor. Now you are left with 1 scoop to pick out of the 10 total flavours. This can be done in 10 ways as well. Thus, there are a total of $10 \times 10 = 100$ combinations of scoops with one flavour repeated at least 3 times. So we have 715 – 100 = 615 ways to choose the scoops of ice cream.

Now, the choice of toppings is independent from the scoops. There are total of 2^7 total combinations of toppings we can receive without restrictions. BUT, we are only allowed to get up to 5 toppings, but at least one topping. Thus we just subtract out the number of ways to get 0, 6 or 7 toppings. There is C(7,0) = 1 way to get zero toppings, C(7, 6) = 7 ways to choose 6 toppings, and C(7, 7)=1 way to choose all 7 toppings. So there are a total of $2^7 - 1 - 7 - 1 = 119$ ways to choose the toppings.

This gives us a final answer of $615 \times 119 = 73185$ possible orders for the customer.

(b) This question is the same as how many injections are there from a set of size 7 to a set of size 10. Imagine the domain being the toppings. Since we are forced to pick each topping exactly once, and none of the flavors are repeated, we are mapping each topping to a distinct element from the co-domain, the set of flavors. We can do this is P(10,7) ways. P(10,7) = 10!/3! = 604800.

■ **Example** Suppose a shipyard produce and sell three brands of motorboats. Define:

- Event A: sell a boat (brands: A1, A2, A3)
- Event B: repair a sold boat.

Suppose Selling Mix: A1 = 70%, A2 = 20% and A3 = 10%

- Likelihood to Repair Given Model A1 = 15%
- Likelihood to Repair Given Model A2 = 10%
- Likelihood to Repair Given Model A3 = 5%

a. Find the probability that a boat you sell will be repaired (7%)

b. Find the posterior (conditional) probabilities p(Ai given B), i =1, 2 and 3 (3% each)

■ **Answer:**

a. $p(A1) = 0.7 \quad p(B/A1) = 0.15$

$p(A2) = 0.2 \quad p(B/A2) = 0.1$

$p(A3) = 0.1 \quad p(B/A3) = 0.05$

$$p(B) = \sum_{i=1}^{3} p(Ai) \cdot p(B/Ai)$$

$$= 0.7 \cdot 0.15 + 0.2 \cdot 0.1 + 0.1 \cdot 0.05 = 1.13$$

b. $$p(A1/B) = \frac{p(A1 \cap B)}{p(B)} = \frac{p(A1)p(B/A1)}{p(B)}$$

$$= \frac{0.7 \cdot 0.15}{0.13} = 0.8077$$

$$p(A2/B) = \frac{p(A2 \cap B)}{p(B)} = \frac{p(A2)p(B/A2)}{p(B)}$$

$$= \frac{0.2 \cdot 0.1}{0.13} = 0.1538$$

$$p(A3/B) = \frac{p(A3 \cap B)}{p(B)} = \frac{p(A3)p(B/A3)}{p(B)}$$

$$= \frac{0.1 \cdot 0.05}{0.13} = 0.0385$$

■ **Example** Determine the overall system reliability (Rs) of a Ship propulsion plant, consisting of two identical engines (1 and 2 in parallel) in series with unit 3, the shafting system, and unit 4, the propeller, if R1=R2=.975, R3=.990 and R4=.980. Keep enough decimal digits for good accuracy, since all units here are quite reliable, as are the main engines, shafts and propellers on most merchant ships.

■ **Answer:**

Rs = prob (system operates)

= prob [(1 OR 2) and 3 AND 4] =

= (R1 + R2 –R1*R2) *R3*R4

= 0.999375*0.99*0.98

= 0.96959

■ **Example** A professor wishes to schedule an appointment with each of her eight teaching assistants, four men and four women, to discuss her large introductory course. Suppose all possible orderings of appointments are equally likely to be selected.

(a) What is the probability that at least one female assistant is among the first three with whom the professor meets? (7%)

(b) What is the probability that after the first five appointments, she has met with all female assistants? (7%)

■ **Answer:**

(a) P(at least one F among 1st 3) = 1 – P(no F's among 1st 3) $= 1 - \frac{4 \cdot 3 \cdot 2}{8 \cdot 7 \cdot 6} = 1 - \frac{24}{336} = 1 - .0714 = .9286$

An alternative method to calculate P(no F's among 1st 3) would be to choose none of the females and 3 of the 4 males, as follows:

$$\frac{\binom{4}{0}\binom{4}{3}}{\binom{8}{3}} = \frac{4}{56} = .0714$$ obviously producing the same result.

(b) P(all F's among 1st 5) = $\dfrac{\binom{4}{4}\binom{4}{1}}{\binom{8}{5}} = \dfrac{4}{56} = .0714$

10.2.2.3 Probability Distributions

Probability distributions and their associated formulas and tables allow us to solve a wide variety of problems in a logical manner. Probability distributions are classified as discrete or continuous. Three discrete distributions will be reviewed in this chapter. Continuous distributions are covered in the next chapter. Probability distributions are used to generate sampling plans, predict yields, arrive at process capabilities, determine the odds in games of chance and many other applications.

The three discrete distributions that will be reviewed:

- The Hypergeometric Probability Distribution
- The Binomial Probability Distribution
- The Poisson Probability Distribution

One of the most difficult tasks for a beginning student in probability is to know which distribution or formula to use for a specific problem. A roadmap is given in section 10.0 of this chapter to assist in the task.

The quality engineer may be asked to calculate the probability of the number of defects or the number of defective units in a sample. There is a difference between the two phrases. A defect is an individual failure to meet a requirement. A defective unit is a unit of product that contains one or more defects. Many defects can occur on one defective unit.

10.2.2.3.1 *The Hypergeometric Probability Distribution*

The hypergeometric distribution is the basic distribution of probability. The hypergeometric probability formula is simply the number of favorable events divided by the number of total events. It can be described as the true basic probability distribution of attributes. To use the hypergeometric formula, the following values must be known.

N = The total number of items in the population (lot size)

n = The number of items to be selected from the population (sample size)

A = The number in the population having a given characteristic

B = The number in the population having another characteristic

a = The number of A that is desired to occur

b = The number of B that is desired to occur

The hypergeometric probability formula is

$$P(a \text{ and } b) = \frac{\binom{A}{a}\binom{B}{b}}{\binom{N}{n}}$$

■ **Example** An urn contains fifteen balls, five red and ten green. What is the probability of obtaining exactly two red and three green balls in drawing five balls without replacement?

This question may also be stated as:

- What is the probability of obtaining two red balls?
- What is the probability of obtaining three green balls?

All three questions are the same. When setting up the problem, all events must be considered regardless of how the question is asked.

In this case, the probability of a single event is not constant from trial to trial. This is the same as sampling without replacement. The outcome of the second draw will be affected by what was obtained on the first draw. The number of favorable events and the number of total events must be computed.

The number of ways that red balls may be selected:

$$\binom{5}{2} = \frac{5!}{3!2!} = \frac{5\times4\times3\times2\times1}{3\times2\times1\times2\times1} = 10$$

The number of ways that green balls may be selected:

$$\binom{10}{3} = \frac{10!}{7!\,3!} = \frac{10\times9\times8\times7!}{7!\times3\times2\times1} = 120$$

The total number of ways to select a sample of five balls from a population of fifteen balls:

$$\binom{15}{5} = \frac{15!}{10!\,5!} = \frac{15\times14\times13\times12\times11\times10!}{10!\times5\times4\times3\times2\times1} = 3003$$

$$\text{Then } P(2r \text{ and } 3g) = \frac{\binom{5}{2}\binom{10}{3}}{\binom{15}{5}} = \frac{10\times120}{3003} = .3996$$

This is a specific application of the hypergeometric probability formula. Many similar problems may be solved using this method. To use the hypergeometric formula, the population must be small enough so that the number of items with the characteristics in question can be determined.

■ **Example** A box contains ten assemblies of which two are defective. A sample of three assemblies is selected at random. What is the probability that the two defective parts will be selected? (For this to occur there must be two defective parts and one good part in the sample.)

$$P(\text{2 defective units}) = P2 = \frac{\binom{2}{2}\binom{8}{1}}{\binom{10}{3}} = \frac{1\times 8}{120} = .0667$$

10.2.2.3.2 *The Binomial Probability Distribution*

The binomial probability formula is used when events are classified in two ways such as good/defective, red/green, go/no-go, etc. The prefix Bi means two. The events or trials must be independent. When the binomial formula is used, it is assumed that the lot size is infinite and the probability of a single success is constant from trial to trial.

The binomial probability formula is be used to answer the following question: What is the probability of x successes in n trials where the probability of a single success is p?. The binomial formula is

$$P(x) = \binom{n}{x}(p)^x (1-p)^{n-x}$$

■ **Example** A coin is tossed five times. (This is the same as a sample size of five). What is the probability of obtaining exactly two heads in the five tosses?

It is known, by prior knowledge, that the probability of a single success (probability of a head in one toss of a coin) is fifty percent. The question is looking for two successes or two heads in five tosses of a coin. A success is the outcome that is desired to occur.

For this example:

- The number of trials = n = 5
- The probability of a single event = p = 1/2
- The number of successes that the question is seeking (x = 2).

To arrive at the answer to the question the values are entered in the binomial formula.

$$P(\text{2Heads}) = \binom{5}{2}(1/2)^2 (1/2)^3 = 10 \times 1/4 \times 1/8$$

$$= .3125 \text{ or } 31.25\%$$

■ **Example** In manufacturing screwdrivers, it was empirically determined that the process yields, on average, 5% defective product. What is the probability that in a sample of ten screwdrivers there are exactly three defective units?

$$n = 10,\ p = .05,\ x = 3$$

$$P(\text{3 defective units}) = \binom{10}{3}(.05)^3 (.95)^7$$

$$= 120 \times .000125 \times .6983 = .0105 \text{ or } 1.05\%$$

■ **Example** A company produces electronic chips by a process that normally averages 2% defective products. A sample of four chips is selected at random and the parts are tested for certain characteristics.

a. What is the probability that exactly one chip is defective?

$$P(\text{1 defective chip}) = P(1) = \binom{4}{1}(.02)^1 (.98)^3$$

$$= 4\,(.02)\,(.9412) = .753$$

b. What is the probability that more than one chip is defective?

More than one defective chip in a sample of four means two, three or four defective chips. The probability of each may be calculated using the binomial formula.

P(more than 1 defective chip) = P(2) or P(3) or P(4) = P(2) + P(3) + P(4)

In any trial or sample, the sum of the probabilities of the individual events always equal one. In this problem: P(0) + P(1) + P(2) + P(3) + P(4) = 1

P(more than 1 defective) = 1 – [P(0) + P(1)] = 1 – [.9224 +.0753] = .0023

■ **Example** A customer has approached a bank for a loan. Without further information, the bank believes there is a 4% chance that the customer will default on the loan. The bank can run a credit check on the customer. The check will yield either a favourable or an unfavourable report. From past experience, the bank believes that *P*(favourable report being received | customer will default) = 0.03 and *P*(favourable report being received | customer will *not* default) = 0.99.

(a) What is the probability that a favourable report will be received?

(b) If a favourable report is received, what is the probability that the customer will default on the loan?

■ **Answer :** Let D be the event that the customer defaults on the loan; ND is the event that the customer doesn't default. Let F be the event that the credit check yields a favourable report; UF is the event that it doesn't. We are given the following probabilities:

P(D)=.04, P(ND)=.96, P(F | D) = 0.03, P(F | ND) = 0.99.

(a) We want to know P(F). Based on the total probability rule:

P(*F*) = P(*D*)×P(*F* | *D*) + P(*ND*)×P(F | *ND*) = (.04×.03) + (.96×.99) = .95

(b) We want to compute P(D | F). Based on the Bayes' rule:

P(D | F) = P(F | D) × P(D) / P(F) = 0.03 × 0.04 / 0.95 = 0.001

■ **Example** Exactly two cab companies operate in Belleville. The Blue Company has blue cabs, and the Green Company has green cabs. Exactly 85% of the cabs are blue

and the other 15% are green. A cab was involved in a hit-and-run accident at night. A witness, Wilbur, identified the cab as a Green cab. Careful tests were done to ascertain peoples' ability to distinguish between blue and green cabs at night. The tests showed that people were able to identify the colour correctly 80% of the time, but they were wrong 20% of the time. What is the probability that the cab involved in the accident was indeed a green cab, as Wilbur says?

■ **Answer:** P(G) = .15 This is the *base rate* of green cabs in the city. It gives the *prior probability* that the cab in the accident is green. Similarly, P(B) = .85.

P($SG|G$) = .80. This is the probability the witness will be correct in *s*aying *g*reen when the cab in fact is *g*reen, i.e., given that the cab really was *g*reen. Similarly, P($SB|B$) = .80. These are the probabilities that witnesses are correct, so by the complement rule, the probabilities of misidentifications are: P($SG|B$) = .2 and P($SB|G$) = .2 .

What we want to know is the probability that the cab really was green, given that Wilbur said it was green, i.e., we want to know P($G|SG$).

According to **Bayes' rule**, this probability is given by:

P($G|SG$) =P(G)×Pr($SG|G$) / Pr(SG) .

We have the values for the two expressions in the numerator: P(G) = .15 and P($SG|G$) = .8, but we have to do a little work to determine the value for the expression P(SG) in the denominator. According to the total probability rule:

$$\begin{aligned} P(SG) &= P(G) \times P(SG|G) + P(B) \times P(SG|B) \\ &= (.15 \times .80) + (.85 \times .20) = .12 + .17 = .29 \end{aligned}$$

Finally, we substitute this number, .29, into the denominator of **Bayes' Rule**:

$$\begin{aligned} P(G|SG) &= P(G) \times P(SG|G) / P(SG) \\ &= 15{\times}.80 / .29 = .414 \end{aligned}$$

So the probability that the witness was correct in saying the cab was green is just a bit above .4—definitely less than fifty/fifty—and (by the complement rule) the probability that he is wrong is nearly .6. This is so even though witnesses are pretty reliable. How can this be? The answer is that the high *base rate* of blue cabs and the low *base rate* of green cabs make it somewhat likely that the witness was wrong in this case.

■ **Example** Officials at an Electronics service center know that 2% of their hotline calls are actually related to genuine problem. A simple verbal test is proposed to help identify genuine calls. She found that :

80% of the people who are having genuine problem will have a positive score on this test; but only 5% of those who are not having a genuine problem will have a positive score on this test.

If we get a positive identification from a caller on this test, what is the probability that he would actually facing genuine problem?

■ **Answer:** Let S be the event that the caller will be having genuine problem; NS that he will not. Let P be the event that the caller has a positive score on the test; NP is the event that the score is not positive. We are given the following probabilities: P(S)=.02, P(NS)=.98, P(P | S) = 0.8, P(P | NS) = 0.05.

We want to know P(S | P). First compute the probability that a caller will have a positive score on the test, P(P). Based on the total probability rule:

$$\begin{aligned} P(P) &= P(S) \times P(P|S) + P(NS) \times P(P|NS) \\ &= (.02 \times .8) + (.98 \times .05) = .065 \end{aligned}$$

Then based on the Bayes' rule:

$$\begin{aligned} P(S|P) &= P(P|S) \times P(S) / P(P) \\ &= 0.8 \times 0.02 / 0.065 = 0.246 \end{aligned}$$

■ **Example** Three manufacturing plants, say A, B and C, produce 20, 30 and 50 percent of a company's output, respectively. The manager of plant C is very quality conscious and only 1% of the items from that plant are defective. Plants A and B have defective rates of 3% and 5% respectively.

(a) What is the probability that a randomly-chosen item from the company's warehouse is defective?

(b) An item is selected at random from the company's warehouse and found to be defective. Calculate the probability it was manufactured in plant C.

■ **Answer:** Let A, B and C be the events that a randomly-chosen item is from plants A, B and C respectively. Let D be the event that a randomly-chosen item is defective. We are given the following probabilities: P(A) = .2, P(B) = .3, P(C) = .5, P(D | A) = .03, P(D | B) = 0.05, P(D | C) = 0.01 .

(a) We want to compute P(D). Based on the total probability rule:

$$\begin{aligned} P(D) &= P(A) \times P(D|A) + P(B) \times P(D|B) + P(C) \\ &\quad \times P(D|C) \\ &= (.2{\times}.03) + (.3{\times}.05) + (.5{\times}.01) = .026 \end{aligned}$$

(b) We want to compute P(C | D). Based on the Bayes' rule:

$$\begin{aligned} P(C|D) &= P(D|C) \times P(C) / P(D) \\ &= 0.01 \times 0.5 / 0.026 = 0.19 \end{aligned}$$

10.2.2.3.3 *The Poisson Probability Distribution*

The Poisson distribution is the mathematical limit to the binomial distribution and may be used to approximate binomial probabilities. The Poisson is also a distribution in its own right when solving problems involving defects per unit rather than fraction defectives.

The Poisson distribution is a discrete distribution. It is often used as a model for the number of events (such as the number of telephone calls at a business, number of customers in waiting lines, number of defects in a given surface area, airplane arrivals, or the number of accidents at an intersection) in a specific time period. It is also useful in ecological studies, e.g., to model the number of prairie dogs found in a square mile of prairie. The major difference between Poisson and Binomial distributions is that the Poisson does not have a fixed number of trials. Instead, it uses the fixed interval of time or space in which the number of successes is recorded.

If n is large and p is small so that n times p (np) is a positive number less than five, then the Poisson is a good approximation to the binomial. The value p and the ratio n/N should be less than 0.10. Here, the terms n, x and p are the same as in the binomial formula. The task is to calculate the probability of x successes in n trials, where the probability of a single success is p. Remember that p is a fraction defective when used to approximate the binomial, and p is defects per unit when counting the number of defects instead of the number of defective units. In some cases neither n nor p is given, but the product np may be given. If p is a fraction defective then np is the average number of defective units in the sample. If p is in terms of defects per unit then np is the average number of defects in the sample. The Poisson formula is

$$P(x) = \frac{e^{-np}(np)^x}{x!}$$

We can also represent the same assuming the mean is λ.

$$p(x, \lambda) = \frac{e^{-\lambda}\lambda^x}{x!} \text{ for } x = 0, 1, 2, ...$$

If we observe, λ is the parameter which indicates the average number of events in the given time interval which is same as n*p in the previous expression.

■ **Example** In making switches, it has been determined by empirical studies that there is, on average, one defect per switch. What is the probability of selecting a sample of five switches that contains zero defects?

■ **Answer:** There are two methods to solve this problem. The first method is to use the above formula where x = 0, n = 5, and p = 1, therefore

$$np = 5 \times 1 = 5.$$

$$P(0) = \frac{e^{-5}(5)^0}{0!} = \frac{.00674}{1} = .00674 \text{ or } .674\%$$

■ **Example** In a paper making operation it was found that each 1000 foot roll contained, on average, one defect. One roll is selected at random from the process.

a. What is the probability that this roll contains zero defects?

■ **Answer:** Use the Poisson table where x = 0 and np = 1. The Poisson table value for P(0) = .368.

b. What is the probability that the roll contains exactly three defects?

■ **Answer:** The Poisson table value for P(3) = .061

c. What is the probability that this roll contains more than one defect?

■ **Answer:** P(more than one defect) = P(2) + P(3) + P(4) + … + P(∞)

= 1 – [P(0) + P(1)]

= 1 – [.368 + .368] = .264

1. If the expected value of a discrete random variable X is $E(X) = 5$, then $E(2X + 4)$ is 14.
2. Suppose that in the binomial probability mass function $b(x; n, p)$, we let $n \to \infty$ and $p \to 0$ in such a way that np approaches a value $\lambda > 0$. Then the binomial distribution approaches the Poisson.
3. If the random variable X has a Poisson distribution with parameter $\lambda = 4$, then the standard deviation of X is half of 4, i.e., 2.

■ **Example** In manufacturing the Que model car, a study determined that on average there are three defects per car. What is the probability of buying a Que with less than three defects?

■ **Answer :**

P(less than 3 defects) = P(0) + P(1) + P(2)

Use the Poisson tables and find P(0), P(1) and P(2) where np = 3

P(less than 3 defects) = .049 + .149 + .224 = .422

■ **Example** On an average Friday, a waitress gets no tip from 5 customers. Find the probability that she will get no tip from 7 customers this Friday.

■ **Answer :** The waitress averages 5 customers that leave no tip on Fridays: λ = 5.

Random Variable : The number of customers that leave her no tip this Friday.

We are interested in P(X = 7)=0.24

■ **Example** During a typical football game, a coach can expect 3.2 injuries. Find the probability that the team will have at most 1 injury in this game.

■ **Answer :** A coach can expect 3.2 injuries : λ = 3.2.

Random Variable : The number of injuries the team has in this game.

We are interested in $P(X \leq 1)$.

■ **Example** A small life insurance company has determined that on the average it receives 6 death claims per day. Find the probability that the company receives at least seven death claims on a randomly selected day.

■ **Answer :**

$P(x \geq 7) = 1 - P(x \leq 6) = 0.393697$

■ **Example** The number of traffic accidents that occurs on a particular stretch of road during a month follows a Poisson distribution with a mean of 9.4. Find the probability that less than two accidents will occur on this stretch of road during a randomly selected month.

■ **Answer :**

$P(x < 2) = P(x = 0) + P(x = 1)$

$= 0.000860$

■ **Example** Textbook authors and publishers work very hard to minimise the number of errors in a text. However, some errors are unavoidable. Mr. J.A. Chapman, statistics editor, reports that the mean number of errors per chapter is 0.8. What is the probability that there are fewer than 2 errors in a particular chapter?[1]

■ **Answer:** This is a Poisson problem, since it involves successes per page. Let's define 'success' as an error. The probability distribution function is as follows:

$$P(X) = \frac{\mu^x e^{-\mu}}{x!},$$

Where, e = 2.7183

X = # of successes

μ = average (mean) number of successes (0.8 in this case)

$$P(X<2) = P(X=0) + P(X=1) = \frac{0.8^0 e^{-0.8}}{0!} + \frac{0.8^1 e^{-0.8}}{1!}$$

$= 0.4493 + 0.3595$

$= 0.8088$

■ **Example** The number of road construction projects that take place at any one time in a certain city follows a Poisson distribution with a mean of 3. Find the probability that exactly five road construction projects are currently taking place in this city. (0.100819)

■ **Example** The number of road construction projects that take place at any one time in a certain city follows a Poisson distribution with a mean of 7. Find the probability that more than four road construction projects are currently taking place in the city. (0.827008)

■ **Example** The number of traffic accidents that occur on a particular stretch of road during a month follows a Poisson distribution with a mean of 7.6. Find the probability that less than three accidents will occur next month on this stretch of road. (0.018757)

[1]From Lind et al. *Basic Statistics for Business and Economics*, 6th ed.

■ **Example** The number of traffic accidents that occur on a particular stretch of road during a month follows a Poisson distribution with a mean of 7. Find the probability of observing exactly three accidents on this stretch of road next month. (0.052129)

■ **Example** The number of traffic accidents that occur on a particular stretch of road during a month follows a Poisson distribution with a mean of 6.8. Find the probability that the next two months will both result in four accidents each occurring on this stretch of road. (0.009846)

■ **Example** Suppose the number of babies born during an 8-hour shift at a hospital's maternity wing follows a Poisson distribution with a mean of 6 an hour. Find the probability that five babies are born during a particular 1-hour period in this maternity wing. (0.160623)

■ **Example** The university policy department must write, on average, five tickets per day to keep department revenues at budgeted levels. Suppose the number of tickets written per day follows a Poisson distribution with a mean of 8.8 tickets per day. Find the probability that less than six tickets are written on a randomly selected day from this distribution. (0.128387)

■ **Example** The number of goals scored at State College hockey games follows a Poisson distribution with a mean of 3 goals per game. Find the probability that each of four randomly selected State College hockey games resulted in six goals being scored. (.00000546)

■ **Example** Suppose the number X of tornadoes observed in Kansas during a 1-year period has a Poisson distribution with $\lambda = 9$.

a. Compute $P(X \leq 5)$ (3%)
b. Compute $P(6 \leq X \leq 9)$ (6%)
c. Compute $P(10 \leq X)$ (3%)
d. How many tornadoes can be expected to be observed during the 1-year period? What is the standard deviation of the number of observed tornadoes? (4%)

■ **Answer:**

a. $P(X \leq 5) = F(5; 9) = .116$
b. $P(6 \leq X \leq 9) = F(9; 9) - F(5; 9) = .587 - .116 = .471$
c. $P(X \geq 10) = 1 - P(X \leq 9) = 1 - F(9; 9)$
   $= 1 - 587 = .413$
d. $E(X) = \lambda = 9$, $\sigma_r = \sqrt{\lambda} = 3$

■ **Example** A student has to answer 8 out of 10 questions in an exam. (i) How many choices does he have? (ii) How many if he must answer the first 3 questions?

■ **Answer:**

(i) To pick 8 out of 10 questions, there are $C(10, 8) = (10.9)/(2.1) = 45$ ways.

(ii) The first 3 must be answered, so he has to answer 5 out of the remaining 7 and there are $C(7, 5) = C(7, 2) = (7 \times 6)/(2 \times 1) = 21$ ways.

■ **Example** Consider a one-dimensional array of N magnets, in which each magnet can only point "up" or "down".

(a) How many different configurations can the system have?

(b) How many ways are there to have $N\uparrow$ "up" magnets (and thus $N\downarrow = N - N\uparrow$ "down" magnets)?

(c) If each "up" magnet carries a spin of +1 and each "down" magnet has spin –1, what are the possible values of the total net spin, S, of the system?

(d) For a given S, in how many ways can the system have the same total net spin S?

■ **Answer:**

(a) The first magnet can be either "up" or "down", so can the second magnet, and so on, hence by the fundamental principle of counting, there are $2 \times 2 \times 2 \times \ldots \times 2 = 2N$ different states that the system could assume.

(b) One may label the magnets from 1 to N, so the problem is to pick $N\uparrow$ out of N labels to be "up", with the rest being "down". Hence the answer is

$$\frac{N!}{N_\uparrow! N_\downarrow!}$$

(this result will be useful when we derive the binomial distribution)

(c) S could be any integer between $-N$ ("all-down") and $+N$ ("all-up"). Note that

$$S = N\uparrow - N\downarrow$$

(d) Since $N = N\uparrow + N\downarrow$ is fixed, specifying $N\uparrow$ determines $N\downarrow$, and hence S, i.e. specifying $N\uparrow$ "up" magnets enforces a particular net spin S $(= N\uparrow - (N - N\uparrow) = 2N\uparrow - N)$. So the answer is the same as that to (b), but rewritten in terms of S (and N), giving

$$\frac{N!}{\left(\frac{N+S}{2}\right)!\left(\frac{N-S}{2}\right)!}$$

■ **Example** A subway train consists of N cars connected in series, m of them being trailer units (TUs) while the rest are electrical units (EUs) powered by the overhead wires. To minimise the chances of immobility in case cars become decoupled, it is required that no TUs are placed next to each other. How many configurations are permissible? (The driver car at each end of the train is ignored in this analysis)

■ **Answer:** One way to tackle this problem is to line up all the $(N - m)$ EUs, reserving one car space between each pair of EUs. The situation is indicated in the picture below:

Space 1	**EU**$_1$	Space 2	**EU**$_2$	……	Space $(N-m)$	**EU**$_{N-m}$	Space $(N - m + 1)$

It is seen that there are (N – m + 1) spaces where it is OK to put in one (or zero) TU; in fact, we will only occupy m of them. So we simply need the number of ways to pick m out of (N – m + 1) available spaces, which is $\binom{N-m+1}{m}$.

■ **Example** A part fails in ten years out of 5,000,000 parts. What is the probability three or more fail in ten years?

■ **Answer:** Let X = number failing in ten years, out of 5,000,000.

X has approximately Poisson, $\lambda = np = 5000000 \times 10_{\text{sup}} = 5.0$.

$P(\text{Three or more fail}) = P(X \geq 3) = 1 - P(X = 0) - P(X = 1) - P(X = 2)$

$$= 1 - \frac{e^{-5}5^0}{0!} - \frac{e^{-5}5^1}{1!} - \frac{e^{-5}5^2}{2!}$$

$$= 1 - e^{-5}(1 + 5 + 12.5) = 0.875$$

■ **Example**

(a) Find the probability of 5 messages arriving in a 2-sec interval.

(b) For how long can the operation of the centre be interrupted, if the probability of losing one or more messages is to be no more than 0.05?

■ **Answer:** Times of arrivals form a Poisson process, rate $\nu = 1.2$/sec.

(a) Let Y = number of messages arriving in a 2-sec interval.

Then $Y \sim$ Poisson, $\lambda = \nu t = 1.2 \times 2 = 2.4$.

$$P(Y = 5) = \frac{e^{-2.4}2.4^5}{5!} = \frac{e^{-2.4}79.626}{120} = 0.060.$$

(b) Let the required time = t seconds.

Let W = number of messages in t seconds, so that $W \sim$ Poisson, $\lambda = 1.2 \times t = 1.2t$

$P(\text{At least one message}) = P(W \geq 1) = 1 - P(W = 0) = 1 - e^{-1.2t} \leq 0.05.$

$\therefore \quad e^{-1.2t} \geq 0.95$

$-1.2t \geq \ln(0.95) = -0.05129$

$t \leq 0.043$ seconds.

Mean (or expected value) of a distribution

For a random variable X taking values 0, 1, 2, ... , the mean value of X is:

$$\mu = \Sigma k P(X = k) = 0 \times P(X = 0) + 1 \times P(X = 1) + 2 \times P(X = 2) + \ldots$$

The mean is also called: population mean expected value of X (or $E(X)$) expectation of X.

Intuitive idea: if X is observed in repeated independent experiments and $\overline{X}_n$ is the sample mean after n observations $\left(=\frac{1}{n}\sum_{i=1}^{n} X_i\right)$, then as n gets bigger, $\overline{X}_n$ tends to μ.

Variance and standard deviation of a distribution

μ is a measure of the "average value" of a distribution.

The standard deviation, σ, is a measure of how spread out the distribution is.

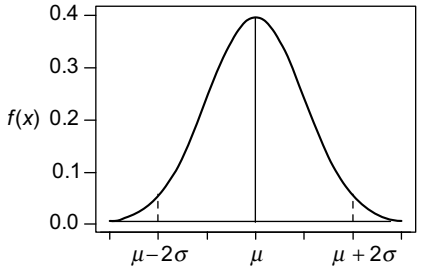
$$\begin{aligned}\text{Variance} = \sigma^2 &= \text{var}(X)\\ &= \Sigma(k-\mu)^2 P(X=k) \quad \text{(definition)}\\ &= \Sigma k^2 \, P(X=k) - \mu^2 \quad \text{(often easier to evaluate in practice).}\end{aligned}$$

10.2.2.3.4 *Normal Distribution*

The continuous random variable X has the Normal distribution with mean μ and variance σ^2 if:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \qquad -\infty < x < \infty$$

The pdf is symmetric about μ. X lies between $\mu - 1.96\,\mu$ and $+ 1.96\,\sigma$ with probability 0.95 i.e., X lies within 2 standard deviations of the mean approximately 95% of the time.



10.2.2.4 More Complicated Situations

Facts: there are

(1) $\binom{p-1}{m-1}$ solutions to the equation

$$\begin{cases} y_1 + y_2 + \ldots + y_m = p \\ \text{all } y_i \text{ non-negative integers} \end{cases};$$

(2) $\binom{n+m-1}{m-1}$ solutions to the equation

$$\begin{cases} x_1 + x_2 + \ldots + x_m = n \\ \text{all } x_i \text{ non-negative integers} \end{cases};$$

Proof:

For (1):

An equivalent problem is "if there are a total of p (indistinguishable) balls, how many ways are there to distribute them into m boxes, if each box must receive at least one ball?", which we solve as follows: line up all p balls, and, to the right of each ball (except the very last one) there's a slot where one could insert a rod as a "separator" to signify the start of the next box. Thus, there are a total of $(p-1)$ such slots, as indicated by the shaded areas in the first row in the picture below. However, we only need to insert $(m-1)$ rods (shaded areas in second row) to separate the balls into m groups.

Ball #	1		2	/	3		.....	$p-1$	/	p
Rod #		no rod		1		no rod	.....		$m-1$	

Hence there are $\binom{p-1}{m-1}$ ways to have the m boxes (each non-empty) carry p balls.

For (2):

Take any one solution $(y_1, y_2, \ldots, y_m)$ from (1), and construct the vector $(x_1, x_2, \ldots, x_m)$ where $x_i = y_i - 1$. Thus, the x's add up to $(p-m)$, and are all non-negative integers. The underlined problem has $\binom{p-1}{m-1}$ solutions; each is obtained by reconstructing a solution from (1). Rewriting $\binom{p-1}{m-1}$ as $\binom{(p-m)+m-1}{m-1}$, we see that there are $\binom{n+m-1}{m-1}$ solutions to the problem "add m non-negative integers together to have a total of n".

■ **Example** Another way to solve the train problem: refer to the picture below, in which all m TUs are lined up. The x's denote the number of EU(s) to connect. Since there must be at least one EU between any two adjacent TUs, $x_2, x_3, \ldots, x_m$ are all positive integers. There can be zero (or more) EU at each end, hence $x_1 \geq 0$, $x_m \geq 0$.

x_1	TU_1	x_2	TU_2	.....	TU_{m-1}	x_m	TU_m	x_{m+1}

Note that the x's must add up to $N-m$ (number of EUs). Now let $X_1 = x_1 + 1$, $X_{m+1} = x_m + 1$, and $X_i = x_i$ for $i = 2 \ldots m$, so the *(m + 1) X's add up to (N – m) + 2 and are all positive integers*. This is the situation described by (1). Hence there are $\binom{(N-m+2)-1}{(m+1)-1} = \binom{N-m+1}{m}$ solutions, which is the same answer as before.

■ **Example** In a test for ESP (Extra Sensory Perception), a subject is told that cards the experimenter can see, but the subject cannot, contain either a star, a circle, a triangle,

a square, or three wavy lines. As the experimenter looks at each of 40 cards in turn, the subject names the shape on the card. A subject who is just guessing has probability 0.20 of guessing correctly on each card.

a. The count of correct guesses in 40 cards has a binomial distribution. What are *n* and *p*? n = 40, p = .2
b. What is the mean number of correct guesses in 40 repetitions? np = 8
c. What is the probability of exactly 5 correct guesses? Binompdf (40,.2,5) = .854
d. What is the probability of 7 or fewer correct guesses? Binomcdf (40,.2,7) = .437
e. What is the probability of getting the first card wrong, but then the next two cards right? (.8)(.2)(.2) = .032.

■ **Example** Suppose that a basketball player has a 78% chance of making a free throw, and each free throw she takes is independent of all the others. Suppose she ends up taking 12 free throws over the course of a game.

a. What is the probability that she makes her first five free throws? $(.78)^5 = .2887$
b. What is the probability that the first free throw she makes is her fourth attempted? = $(.22)^3(.78)$ = geometpdf (.78,4) = .008305
c. What is the probability that she makes exactly 5 out of the 12 free throws? It's binompdf(12,.78,5) = .005704
d. What is the probability that she makes at least 8 of the free throws? We do this by calculating 1-binomcdf(12,.78,7)=.8979
e. Let *X* be a random variable defined by the number of points she scores shooting free throws in this game. (Free throws are worth one point each.) The possible values of *X* are 1, 2, 3, … , 12. Find the expected value of points, *E*(*X*).

So there's a long way and a short way. The long way is to do:

$$\sum_{x=0}^{12} x \cdot binompdf(12,.78,x) =$$

0.*binompdf* (12,.78,0) + 1. *binompdf* (12,.78,1) + 2. *binompdf* (12,.78,2) + … 12. *binompdf* (12,.78,12) = 9.36

The easy way is to find the expected value of the number of free throws she makes using what we know about the binomial probability model: *np* = 12(.78) = 9.36

■ **Example** A shipyard produces 3 standardized ships at a cost of $50 million a ship. The ships will be sold at a price of $100 million each. The scrapyards purchase any unsold ships from the shipyard after a time period for $20 million a ship. Let *X* be the # of ships sold at the end of the period. You are given that P(x = 0,1,2,3) are 0.2, 0.3, 0.3 and 0.2, respectively. Calculate the expected value and the variance of the shipyard total profit (or loss).

■ **Answer:**

X: Number of ships sold at the end of the period.

$$E[X] = \sum_{0}^{3} X \cdot p(X) = 0 \cdot 0.2 + 1 \cdot 0.3 + 2 \cdot 0.3 + 3 \cdot 0.2 = 1.5$$

$$E[X^2] = \sum_{0}^{3} X^2 \cdot p(X) = 3.3$$

$$Var(X) = E[X^2] - E[X]^2 = 3.3 - 1.5^2 = 1.05$$

Total Profit = 100*X* – 50*3 + (3 – *X*)*20 = 80*X* – 90
E[Total Profit] = *E*(80*X* – 90) = 80**E[X]* – 90 = **$30 m**
Var(Total Profit) = *Var*(80*X* – 90) = 80^2 *Var(X)* = **$^{2}6720 m**

10.2.2.5 Pack of Cards – Simple Probability Questions

What is the probability of choosing, at random one of the following cards from a normal pack of 52 playing cards?

1.	A red card	26//52	½
2.	A black card or 'not a red card'	26/52	½
3.	A spade	13/52	¼
4.	Not a spade	39/52	¾
5.	An ace	4/52	1/13
6.	Not an ace	48/52	12/13
7.	The ace of spades	1/52	
8.	A picture card	12/52	3/13
9.	A number card or 'not a picture card'	40/52	10/13
10.	A card that is either a heart or a club	26/52	½
11.	A card that is neither a heart or a club	26/52	½
12.	A 4 or 5	8/52	2/13
13.	A 4 or 5 but not a spade	6/52	3/26
14.	An even numbered card	20/52	5/13
15.	A card that would be higher that a 3, Aces high	44/52	11/13
16.	A card that would be higher that a 7, Aces high	28/52	7/13
17.	A card that would be higher that a Jack, Aces high	12/52	3/13
18.	A card that would match or be higher than a 7, Aces high	32/52	8/13

Dice – Simple Probability Questions

What is the probability of rolling from a normal 6 sided die?

1.	A 6	1/6	
2.	Higher than a 4	2/6	1/3
3.	An odd number	3/6	½
4.	A factor of 24 (1, 2, 3, 4, 6)	5/6	

What is the probability of rolling the following from a 4 sided die?

5.	A4	¼	
6.	An even number	2/4	½
7.	Higher than 3	¼	
8.	A factor of 24 (1, 2, 3, 4)	1, certain	

What is the probability of rolling the following from a 10 sided die?

9.	A 10	1/10	
10.	A 6 or higher	5/10	½
11.	Less than a 5	4/10	2/5
12.	A 5 or less	5/10	½
13.	A factor of 24 (1, 2, 3, 4, 6, 8)	6/10	3/5
14.	A number that when spelt begins with a 't'	3/10	

What is the probability of rolling the following from a 12 sided die?

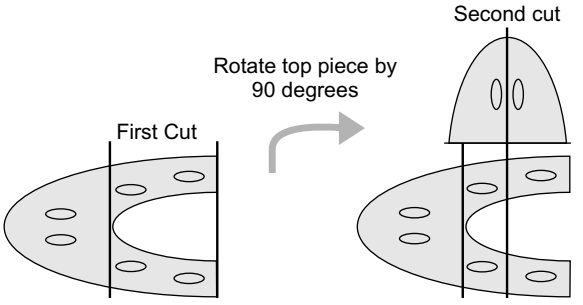
15.	A 12	1/12	
16.	Higher than a 7	5/12	
17.	A factor of 24 (1, 2, 3, 4, 6, 8, 12)	7/12	
18.	A teen number	0	
19.	A number than when spelt with a 't'	4/12	1/3

Solved Questions (Including Previous Years' GATE Questions)

1. We have two jars with equal volumes of a liquid and water in each. A big sized table spoon full of liquid is taken from first jar and added to the jar with water. The second jar content is thoroughly stirred and the same table spoon is used to move one spoon full of content from second jar to first jar. What will be volumes of both the jars? Also comment on the weights of the liquids of both the jars.

Answer: Volumes will be same. However, weights depends on their relative specific gravities.

2. See the following figure which contains a horse shoe with six holes in it to fix nails. Identify with how many minimum number of cuts, we can divide this horse shoe into pieces such that one piece contains one whole.



Answer: With two vertical cuts we can solve this problem. First we draw a vertical line as shown in the figure and separate the top half. Rotate this top half by 90 degrees and position as shown in the figure and then make another vertical cut.

3. Egg-drop experiment to find out how many falls are needed to know from which floor egg gets broken. Assume the building is 32-storey high and floors are safe to drop eggs from.

a. An egg that survives a drop can be used again.
b. A broken egg cannot be used again.
c. The effect of a fall is the same for all eggs for all floors. That is, we assume egg reaches the ground with same speed, and other properties.
d. If an egg breaks when dropped from some floor, it would break also if dropped from a higher floor. This is very important assumption.
e. If an egg survives a fall when dropped from some floor, it would survive also if dropped from a lower floor. This is another important assumption which is crucial in solving this puzzle.
f. There are no pre-existing assumptions concerning when the egg will break. It is possible that a drop from the first floor in the special container would break an egg. It is also possible that a drop from the 36th floor in the special container would not break an egg.

How do we find the floor from which the egg breaks if we have only one egg?

To obtain the required result, we may start by dropping the egg from the first floor. If it breaks, we know the answer. If it survives, we drop it from the second floor and continue upward until the egg breaks. Thus, the worst-case scenario would require 32 drops to determine the egg-breaking floor.

Assume that we have been given two eggs. Find out minimalistic number of falls and from which floor egg fails.

Answer: We can follow divide and concur policy. First, we may try from 16^{th} floor. If it breaks, then you can carry from 1^{st} floor to 16^{th} floor sequential with the other egg. Otherwise, we have search in 16^{th} floor and above. Next, we try from the floor 24 with the non-broken egg. If it breaks, then we have search floors 17 to 23 in sequential manner. Otherwise you have search 25-32 in a binary search fashion.

4. A person starts from a station X and reaches station Y with the speed of 40KMPH. He returns immediately from Y and reaches X with the speed of 60 KMPH. What is his average speed during this journey?

Answer: Assume the distance between X and Y as SKM

Time for X to Y = S/40

Time needed for Y to X = S/60

Therefore, average speed = (2*S)/(S/40 + S/60))=(2*S)/(100*S/240)= 2*240/100 = 48KMPH.

5. Two gas stoves are having two burners each. Each of the stoves are equipped with gas cylinders which burn a burner for one hour. The inmate got a phone call to attend a meeting for which he has to start exactly after 45 minutes and he does not have any means of counting 45 minutes. He has decided to use gas stoves and measured successfully 45 minutes. Explain how he might have done?

Answer: He burns both the burners of one stove and only one burner of the other stove. After exactly half an hour, first cylinder gets emptied and burners of first stove stops burning. At that instance second burner of the second sound will be lighten. When both the burners of the second stove stops, he can consider that instance as the 45^{th} minute.

6. A person has small scale farming industry with 20 rabbits, 30 hens/chickens , 50 buffalos, and 40 bullocks. He has the habit of calling bullocks as chickens? How many chickens he has?

Answer: 30

7. We have eight balls of same shape and colour. However, one of the balls is little heavy. You have been given a old fashioned mechanical weighing scale with two plates both sides. How many minimum number of weighing's are needed to separate out this ball from others.

Answer: Two. First, we place three balls in each side of the scale. If both side weighs same, then the ball is in the left over two balls. So, we can make another weighing operation to identify the same. Otherwise, ball can be either in the left three or right three. We select the three balls which weighs more than other three and carry second weighing operation in which we place one ball in each side and keep the third one away. I hope it is clear by now.

8. Yesterday there was a down pour. What will be the chance of sunny day after three days from today.

Answer: 0. No basis to predict.

9. After every 60 minutes hours and minutes indicators of a watch are covering or crossing each other. Does the watch work properly?

Answer: No

10. After ____ minutes, hours and minutes indicators of a correctly functioning clock crosses each other

A. 60 B. 58.22 C. 65.4545 D. None

Explanation:

Hours indicator rotational speed = 360/(12*60) = 1/2 Degrees per minute

Minutes indicators rotational speed = 360/60 = 6 Degrees per minute

Consider a situation at which both hours and minutes indicators are at 0 (or 12).

Let after x minutes both will cross each other. Rather, minutes indicator makes a full round and some more angle (y) before crossing. That is, the angle y made by hours indicator is same as the angle made by the minutes indicator with x-60 minutes. Now, based on this equality we have to solve for x. That is,

(x – 60)*6 = x/2

Therefore, x = 65.4545 minutes

11. After ____ seconds minutes and seconds indicators of a correctly functioning clock crosses each other

A. 60 B. 61.0169 C. 65.4545 D. None

Explanation:

Minutes indicators rotational speed = 360/(60*60) = 1/10 Degrees per second

Seconds indicators rotational speed = 360/60 = 6 Degrees per Sec

Consider a situation at which both seconds and minutes indicators are at 0 (or 12).

Let after x seconds both will cross each other. Rather, seconds indicator makes a full round and some more angle y before crossing. That is, the angle y made by minutes indicator is same as the angle made by the seconds indicator with x-60 seconds. Now, based on this equality we have to solve for x. That is,

(x – 60)*6 = x/10

Therefore, x=61.0169 seconds

12. There is a table on which a number of coins are placed. There are as many coins with Heads up as many coins with Tails up. Divide the coins (number of coins is even) into two equal piles such that number of coins with Heads up and Tails up in either piles be the same. Assume your eyes are covered with cloth such that you cannot determine the sides (for sure) if you are blinded. Also, the head or tail cannot be decided by touch. **(Microsoft Interviews)**

Answer: Divide the coins in half by quantity then, flip all the coins in one pile.

13. There are N doors in a row numbered from 1 to N. Initially all the doors are closed. We make N passes over all the doors. In pass 1 we toggle all doors (1,2,3,4....) starting from the first door. In the second pass we toggle every second door (2,4,6,8,...). In the third pass we toggle all third doors (3,6,9...). Similarly we make N passes. What is the state of door k after N passes. **(Amazon Interviews)**

Answer:

We take a sample of 16 doors. Here, o means open and x means closed.

Door	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Initially	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Pass 1	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
Pass 2		x		x		x		x		x		x		x		x
Pass 3			x			o			x			o			x	
Pass 4				o				0				x				0
Pass 5					x					o					o	
Pass 6						x						o				
Pass 7							x							o		
Pass 8								x								x
Factors	1	1,2	1,3	1,2,4	1,5	1,2, 3,6	1,7	1,2, 4,8	1,3,9	1,2, 5,10	1,11	1,2,3, 4,12	1,13	1,2, 7,14	1,3, 5,15	1,2,4, 8,15
No of factors	1	2	2	3	2	4	2	4	3	4	2	5	2	4	4	5

From the above table, we can find that the doors gets flipped that many times as that of number of factors of the door number including 1 and door number. That is, for example door number 8 gets flipped 4 times with 1, 2,4 and 8. Thus, its final status is closed. Rather for kth door if the number factors (including 1 and k) are even then its final status is closed else opened.

14. Assume that a corner of a cake got broken in a non-linear fashion which you want to make into halves. How would you do?

Answer: Slice the cake horizontally along parallel to its base.

15. Is it possible to place four points in a x-y plane such that all the points are equidistant from other points? What about if we assume 3-D system?

Answer: It is not possible to place four points in 2D plane. However, in 3D it is possible. We assume that the points are four corners of a tetrahedron.

16. A person of height H enters into a room with an accurate wrist watch. There is hanging incandescent bulb which is hanging exactly from ceiling and just touching his head. He wants to find the height of room. Unfortunately he doesn't have scale. Of course, he has successfully completed his 10+2 and undergone simple pendulum based lab experiment and theory behind it. Explain how he can calculate room height?

Answer: If we find the length of wire (l) for which bulb is attached then by adding this to H we can find out the height of the room. He swings the bulb with his head and calculates the average time period of the oscillation with the help of his wrist watch. May be he measures time for few number of oscillations instead of single oscillation to avoid error in estimation. Then, he calculates the wire height (l) using the following formulae where g is the acceleration due to gravity which he may take as 9.8.

$$T = 2\pi\sqrt{\frac{L}{g}}$$

17. How do you measure height of a building when you are at the top of the building with a stone in your one hand and wrist watch in the other hand?

Answer: Drop the stone and find the time taken for the stone to reach the ground. Find height using the formula s = a + gt (s = height, a= initial velocity=0, g=9.8m/s, t = time taken which is measured through the wrist watch)

18. A racing court perimeter is 80KM. In the first run, a participant drove with the speed of 40KMPH. During his second trip he increased his speed. What is the acceptable or practically possible average speed of this participant over these two rounds?

A. 55KMPH B. 100KMPH
C. 120KMPH D. 160KMPH

19. A person makes a random walk from a point. He moves one mile east, half mile in north direction, quarter mile in west direction and then one by eighth mile in south direction. What is his displacement from the starting point?

Answer: Assuming (0, 0) as the starting points then next points of his path can be given as: (1, 0), (1, 1/2), (3/4, 1/2) and (3/4, 3/8). Therefore, displacement from starting point can be given as: sqrt($(3/4)^2 + (3/8)^2$)

20. A set of numbers are organised in rows. A is the largest out of the largest values of each row while B is the smallest. Which is the largest out of A and B?

A. A B. B
C. Neither A nor B D. None

21. When a person started his journey by spending 1/3 of the money what he has in his packet. He observed 1/5 of the remaining is spent on food during the travel. He donated 1/5 of the remaining money to a beggar. When he has reached the destination he has 100 Rupees left in his packet. With how mach money has he started his journey.

Answer:

Let the money in his packet in the beginning is x.

Travel =x/3

Food = (x-x/3)*1/5 =2x/15

Donation= (x – x/3 -2x/15)1/5=8x/75

Left over money=100= (x – x/3 – 2x/15 – 8x/75)

= 32x/75

Now solve for x.

x = 234.4

22. When a person started his journey by spending 1/3 of the money what he has in his packet. He observed 1/5 of the remaining is spent on food during the travel. He donated 1/5 of the remaining money to a beggar. When he has reached the destination he has 100 Rupees left in his packet. With what money he has started his journey?

Answer:

Let the money in his packet in the beginning is x.

Travel =x/3

Food = (x-x/3)*1/5 =2x/15

Donation= (x – x/3 -2x/15)1/5=8x/75

Left over money=100= (x – x/3 – 2x/15 – 8x/75)

= 32x/75

Now solve for x.

x=234.4

23. When a person started his journey by spending 1/3 of the money what he has in his packet. He observed 1/5 of the remaining is spent on food during the travel. He donated 100 rupees to a beggar which he understood that it 1/5 of the remaining money in his packet. With how much money has he started his journey?

Answer:

Let the money in his packet in the beginning is x.

Travel =x/3

Food = (x-x/3)*1/5 =2x/15

Donation= (x – x/3 -2x/15)1/5=8x/75

As donation is 100, i.e., 100=8x/75. Now solve for x.

x=937.5

24. Three chains with number of with O type links are proposed to be joined to make a single chain. Number of links in each of the chains are: 3, 3, 1. How many links have to be opened by the blocksmith to make a single chain?

A. 1 B. 2
C. 3 D. None

25. A shooter made a single huge thundering shot at the birds on a tree and 2 birds have fallen. How many more birds does he gets for his second shot?

A. 1 B. 4 C. None

26. A big cube with all the sides painted is divided into small cubes of side ¼ of the big cube. Then, the number of cubes which does not have their sides painted are:

A. 16 B. 8 C. 2 D. 4

27. A big cube with all the sides painted is divided into small cubes of side ¼ of the big cube. Then, the number of cubes which does have three of their sides painted are:

A. 16 B. 8 C. 2 D. 4

28. A big cube with all the sides painted is divided into small cubes of side ¼ of the big cube. Then, the number of cubes which does have two of their sides painted are:

A. 16 B. 8 C. 24 D. 4

29. A big cube with all the sides painted is divided into small cubes of side ¼ of the big cube. Then, the number of cubes which does have one of their sides painted are:

A. 16 B. 8 C. 24 D. 4

30. A big cube with all the sides painted is divided into small cubes of side ¼ of the big cube. Then,

A. Number of small cubes with no sides painted are same in number with numbered cubes with three sides painted

B. Number of small cubes with two sides painted are same in number with numbered cubes with single side painted

C. Number of small cubes with single sides painted are same in number with numbered cubes with three sides painted

D. A and B

31. A big cube with all the sides painted is divided into small cubes of side ¼ of the big cube. Then, number of small cubes with 0, 1, 2 and 3 sides painted are:

A. 8, 8, 24, 24 B. 8, 8, 8, 8

C. 8, 24, 24, 8 D. 8, 24, 8, 24

32. A 4-inch big cube with all the sides painted is divided into small cubes of side ¼ of the big cube. Then, the total areas of painted and un-painted sides of all the cubes

A. 96,284 B. 8,8,8,8

C. 96,288 D. None

33. A 4-inch big cube with all the sides painted is divided into small cubes of side ¼ of the big cube. It is decided to apply paint to all the sides of small cubes also wherever the sides are not painted. If x is the cost of painting all the sides of the big, how much more we have spend now?

A. 1000 Rupees B. 2x

C. 3x D. None

Explanation: When we divide the big cube, we get 8 small cubes with 0 sides painted, 24 small cubes with single side painted, 24 small cubes with double sided coating, and 8 small cubes three sides painted. Therefore total area required to be painted = 8x6+24x5+24x4+8x3=288. Total painted area of the original big cube is 6x4x4=96. We know x is the cost of painting 96 units. Therefore, 288/96*x=3x is the extra cost needed for painting sides of smaller cubes.

34. A 4-inch big cube is divided into small cubes of side ¼ of the big cube. The ratio of total surface areas of small cubes to big cube

A. 3.4 B. 2 C. 3 D. 4

35. If a unit cube painted on all sides is divided into smaller cubes of sizes ½, ¼, 1/8, 1/16. Then, the number of cubes with no painted sides are:

A. 2, 4, 8 B. 0, 8, 6^3,14^3

C. 2^3,6^3,16^3 D. None

36. If a unit cube painted on all sides is divided into smaller cubes of sizes ½, ¼, 1/8, 1/16. Then, the number of cubes with three painted sides are:

A. 2, 4, 8 B. 0, 8, 6^3,14^3

C. 8,6^3,16^3 D. 8,8,8,8, …..

37. If a unit cube painted on all sides is divided into smaller cubes of sizes ½, ¼, 1/8, 1/16. Then, the number of cubes with two painted sides are:

A. 2, 14, 8,… B. 0, 24, 72, 168,…

C. 8,6^3,16^3 ,… D. 8,8,8,8, …..

38. If a unit cube painted on all sides is divided into smaller cubes of sizes ½, ¼, 1/8, 1/16. Then, the number of cubes with one painted side are:

A. 2, 4, 8 B. 0, 24, 196, 1176,…..

C. 2,8,6^3,16^3 D. 0,0,8,8, …..

39. A unit cube is painted on all sides and divided into smaller cubes of size 1/n, where n is a positive integer which is in the integral power of 2. Then, number of cubes with only two sides painted are:

A. 2n B. 12(n–2)

C. $6(n-2)^2$ D. None

40. A unit cube is painted on all sides and divided into smaller cubes of size 1/n, where n is a positive integer which is in the integral power of 2. Then, number of cubes with only one side painted are:

A. 2n B. 12(n–2)

C. $6(n-2)^2$ D. None

41. A unit cube is painted on all sides and divided into smaller cubes of size 1/n, where n is a positive integer which is in the integral power of 2. Then, number of cubes with only three sides painted are:

A. 2n B. 12(n–2)

C. $6(n-2)^2$ D. 8

42. A unit cube is painted on all sides and divided into smaller cubes of size 1/n, where n is a positive integer which is in the integral power of 2. Then, number of cubes with only four sides painted are:

A. 8 B. 12(n–2) C. $6(n-2)^2$ D. 0

43. A unit cube is painted on all sides and divided into smaller cubes of size 1/n, where n is a positive integer which is in the integral power of 2. Then, number of cubes with three, four, five, and six sides painted are:

A. 0,8,8,8 B. 0,0,0,0, C. 8,0,8,0 D. 8,0,0,0

44. A big cube with all the sides painted is divided into small cubes of side ½ of the big cube. Then, number of small cubes with 0, 1, 2 and 3 sides painted are:

A. 8,8,2,2 B. 8,8,8,8 C. 0,0,0,8 D. 8,24,8,24

45. Every day 20 ml of ethanol gets evaporated from a 100ml opened ethanol bottle. On a windy day, evaporation rate gets doubled. What is the best and worst estimation of days such that some ethanol is found in the bottle after opening the bottle.

A. 5, 0 B. 0,5

C. 5, 2 and half days D. None

46. Every day 20 ml of ethanol gets evaporated from a 100ml opened ethanol bottle. On a windy day, evaporation rate gets doubled. What is the best and worst estimation of days such that some ethanol is found in the bottle after opening the bottle. Assume every alternative day is a windy day.

A. 5, 0

B. 3 and 3 and half days

C. 5, 2 and half days

D. None

Explanation: If starting day is non-windy day, then evaporation: 20,40,20,20 (3 and half days)

If starting day is windy day, then evaporation:40,20,40 (Three days)

47. Every day 20 ml of ethanol gets evaporated from a 100ml opened ethanol bottle. On a windy day, evaporation rate gets doubled. What is the best and worst estimation of days such that some ethanol is found in the bottle after opening the bottle. Probability of next day being windy day is 0.9 if today is non-windy day. Always, there will be a windy day after two consecutive non-windy days. Also, a windy day follows non-windy day.

A. 5, 0

B. 3 and 4

C. 5, 2 and half days

D. None

Explanation: Notation: N-Non-Windy Day W-Windy Day

Possibilities:

N – W – N –W = 20 + 40 + 20 + 20 = 3½ day

N – N – W – N = 20 + 20 + 40 + 20 = 4 days

N – W – N – N = 20 + 40 + 20 + 20 = 4 days

W – N –W = 40 + 20 + 30 = 3 days

48. A ship left a port city and when it is y distance from port a plain started from the same port city in the same direction with n times speed of the ship x. Both met after t time units. Then, acceptable relation

A. 10nt=y+nt B. y+nt=10nx

C. y+xt=nxt D. None

49. A ship when it is at y distance from a port city a plain started from the same port city in the opposite direction with n times speed of the ship x. Both met after t time units. Then, acceptable relation

A. 10nt=y+nt B. (n-1)xt=t

C. y+xt=nxt D. None

50. Evaporation of a liquid is proportional to the exposed surface area to the air of the container. A liquid which is in a 10m cubicle container with top open is shifted to another cylindrical container of diameter 10m and also top circular face opened. Then,

A. Evaporation reduces

B. Evaporation increased

C. Evaporation reduces by π/4 times

D. None

51. A person standing by the side of rail track observed that it took 5 seconds for a train to cross him wholly. After another 10 second another train approached in opposite direction crossed him within 1.5 seconds. What is the speed ratio of first and second trains?

A. 3 B. 5/1.5

C. 1.5

D. Insufficient information

52. A person standing by the side of rail track observed that it took 5 seconds for a train to cross him wholly. When the last guard bogie is about to cross him he started walking opposite direction of the train. At the same time, on another track a train of 200 m length arrived in opposite direction of the first train and along the direction of the person. After another 10 seconds when this person travelled 40 m, the last bogie of the second train crossed him. What is the speed of the second train?

A. 82.34KMPH B. 86.4

C. 87.84KMPH D. Insufficient information

Explanation: Information about the first train is not important here. From the given information, in order to travel (200+40) m, second train took 10 seconds. Therefore, the speed of the second train can be given as: (240*3600)/(10*1000) = 86.4 KMPH

53. Usually in Indian rail track, along the track electric poles are separated by 50m. A guard has observed a 200m long second train's engine to cross him at an electric pole. When he crosses fourth electric pole, guard of the other trained waved him hello. What is the speed ratio of first and second trains?

A. 3.122 B. 2.33 C. 3.33 D. None

54. A father brought some items to home and gave half of them to his favorite child. Eight percent of the leftover is taken by mother and kept 1 for herself and remaining were given to her favorite child. Finally, left 2 were taken by another child. How many items were really available initially?

A. 21 B. 28 C. 40 D. 20

55. A small child has a habit to throw a 70ml cool drink bottle when it still contains 10 ml of drink. Also, he picks up the bottle only when it is full. One day, mother found only the used bottles available in the hose, not a single full bottle is available. How many bottles with left liquid is required to be transferred to satisfy the child's requirement now?

A. 8 B. 6 C. 7 D. None

56. A person said that I have three children out of them older ones are twins. Product of their ages is 36. The children possible ages are:

A. 2,3,3 B. 3,3,4 C. 1,6,6 D. 2,2,9

57. A University conducts online examinations in its computer laboratories with three slots per day 8-11AM, 11AM-2Noon, and 2Noon-5PM. Examinations are conducted for 10 days in a semester and any student can schedule his examinations within this 10 days period. A student has to appear for total 7 papers. Ram is weak in two papers for which he needs full 2 days preparation. Whereas for other papers, he can write on any day without any preparation. In how many days Ram can schedule his examinations such that he will be spending minimal number of days in the examinations?

Answer: 4 days. He will schedule one of the difficult papers on the first day during first slot. On the third day first slot he schedules the left over difficult paper. Remaining papers will be faced till 4^{th} day last slot (including). Thus, four days he will be spending in examinations.

58. An elevator moves down 3 steps per second. In its unmoving position, elevator is observed to be having 50 steps. A boy started climbing in the opposite direction at the speed of 4 steps per second or per jump. Then, time needed for the boy for climbing the next floor.

A. 60/2 B. 60/3 C. 50/1 D. 60/4

59. A family (father and mother) with two male and female children and police with a thief are supposed to cross a river a small boat which can accommodate two people only at a time. Only police, husband and wife can drive the boat. Father dislikes female children while mother hates male children. Thus, male children can go only with father while female children can go along with mother. Also, if police is not around, thief may rob the others. Suggest the possible schedules of the boat along with people such that all people will be crossing the river.

Schedule: One possible schedule is given below for the situation in which boat can accommodate two people only. We assume all the people are left bank of the river. Arrows shows the movement.

Police + Thief ⟶ Towards right
Towards left ⟵ Police
Police + boy ⟶ Towards right
Towards left ⟵ Police+Thief
Father + boy ⟶ Towards right
Towards left ⟵ Father
Father + Mother ⟶ Towards right
Towards left ⟵ Mother
Police + thief ⟶ Towards right
Towards left ⟵ Father
Father + Mother ⟶ Towards right
Towards left ⟵ Mother
Mother + Girl ⟶ Towards right
Towards left ⟵ Police +Thief
Police + girl ⟶ Towards right
Towards left ⟵ Police
Police+Thief ⟶ Towards right

If boat accommodates three people:

Father + 2 Boys ⟶ Towards right
Towards left ⟵ Father
Father + Mother ⟶ Towards right
Toward left ⟵ Mother
Mother + Police + Thief ⟶ Towards right
Towards left ⟵ Mother
Mother + 2 Girls ⟶ Towards right

60. Which weighs more? A pound of rice or a pound of gold?

Answer: A pound of rice. While weighing rice, pound contains 16 ounces while with gold pound contains 12 ounces.

61. A father distributed cash in his wallet to four of his sons equally asking them to spend within a day. All the four except the last one returned 400 rupees and spent the remaining. The last one returned 100 rupees less than other brothers. The eldest son likes the last one, he gave 100 rupees to his last brother soon after he got money from his father. Father observed that the total money returned by all of his sons is half of the money he has distributed initially. How much

money the father distributed? How much the last son returned to his father? Out of all, which son spent more and less?

Answer: Let the money given to each son as x.

Money returned: 400 + 400 + 400 + 300 = 2x

X = 750

Therefore, money distributed by the father = 4x = 4x750 = 3000

The money spent by last son=750+100-300=550

Money spent by each of the second and third sons= 750-400=350

Money spent by first son = 750-400-100=250

Eldest son spent less while youngest spent more.

62. A person in the last coach of 200m train started rushing towards middle coaches as soon as he has seen beginning of his stations 400m platform as foot over bridge to cross towards other platforms is at the other end of the platform. Speed of the train is 60KMPH and it took 10sec for train to halt. He found he is coming out from middle coach. Find out how much distance he has to walk to climb the foot over bridge.

Answer:

We may have to use physics equations related to velocity, distance, acceleration.

Initial velocity (u) =60KMPH= 60*5/18=16.67m/sec

Therefore, deceleration = (from v = u + at formulae)=-16.67/10=1.667m/sec^2

Distance travelled by the train during 10 sec= (s= ut + ½ * at^2)= 16.67*10 – 0.5*1.667*10*10 = 83.35.

As he has dropped from middle coach, he has dropped at 100+83.35m from one end of the platform. Therefore, he has to walk (400-183.35)m to climb the foot over bridge.

63. A station contains a platform of length 200m with one end foot over bridge. At what speed a train of length 200 m has to have when it approaches the other end of platform such that exactly middle bogie will be at the foot over bridge. It took 10 seconds for the train to come to standstill.

Answer:

In order to achieve this, the train has to travel a distance of 300m after reaching the other end of platform.

Initial velocity u = –at

Final velocity v=0

Therefore $0^2 – (–at)^2=2as$

$–a^2t^2=2as$

$a = 2s/t^2 = -2*300/100=–6m/sec^2$

u= –(–6*10) = 60m//sec = 216KMPH

64. Acceptable probability of an event

A. 0.8 B. –0.001 C. 1.0001 D. 0

E. 1

65. Ram has 6 hundred rupee notes and 5 fifty rupee notes in his wallet. If he randomly grabs 2 notes, what is the probability those notes being hundred and fifty?

A. ½ B. 1/6*1/5

C. 3/11 D. 6/11*5/10

66. In a shipment of notebooks, 1/50 of the pieces are defective. What is the ratio of defective to non-defective ones?

A. 1/200 B. 1/50 C. 1/49 D. 49/1

E. 50/1

67. A utensil contains oil that got evaporated 1/3 of the volume on first day and 2/5 of the remaining in the second day then the left over liquid in terms of original liquid volume

A. 1/5 B. 2/6 C. 2/5 D. None

68. A political party orders an arch for the entrance to the ground in which the annual convention is being held. The profile of the arch follows the equation $y=2x-0.1x^2$ where y is the height of the arch in metre. The maximum possible height of the arch is:

(GATE CSE 2012)

A. 8 metre B. 10 metre

C. 12 metre D. 14 metre

Answer: Find out the first derivative of the profile and equate the same to 0 to find for what value of x maximum height is possible.

dy/dx = 2– 0.1*2*x =0

x=1/0.1 = 10 m

Maximum possible height = $2*10 – 0.1*10^2$ = 10m

69. An automobile plant contracted to buy shock absorbers from two suppliers X and Y. X supplies 60% and Y supplies 40% of the shock absorbers. All shock absorbers are subjected to a quality test. The ones that pass the quality test are considered reliable. Of X's shock absorbers, 96% are reliable. Of Ys shock absorbers, 72% are reliable. The probability that a randomly chosen shock absorber, which is found to be reliable, is made by Y is: **(GATE CSE 2012)**

A. 0.288 B. 0.334

C. 0.667 D. 0.720

70. Which of the following assertions are correct?

(GATE CSE 2012)

P: Adding 7 to each entry in a list adds 7 to the mean of the list

Q: Adding 7 to each entry in a list adds 7 to the standard deviation of the list

R: Doubling each entry in a list doubles the mean of the list.

S: Doubling each entry in a list leaves the standard deviation of the list unchanged.

A. P,Q B. Q, R C. P, R D. R, S

71. Given the sequence of terms, AD, CG, FK and JP, the next term is: **(GATE CSE 2012)**

A. OV B. OW C. PV D. PW

72. The cost function for a product in a firm is given by $5q^2$, where q is the amount of production. The firm can sell the product at a market price of 50 rupees. The number of units to be produced by the firm such that the profit is maximised is: **(GATE CSE 2012)**

A. 5 B. 10 C. 15 D. 25

73. 25 persons are in a room.15 of them play hockey. 17 of them play football and 10 of them play both hockey and football. Then the number of person playing neither hockey nor football is: **(GATE CSE 2010)**

A. 2 B. 17 C. 13 D. 3

74. If 137+276=435 how much is 731+672?

(GATE CSE 2010)

A. 534 B. 1403 C. 1623 D. 1513

75. Hari (H), Gita (G), Irfan (I) and Saira (S) are siblings (i.e. brothers and sisters). All were born on 1st january. The age difference between any two successive siblings (that is born one after another) is less than 3 years. Given the following facts:

i. Hari's age + Gita's age > Irfan's age + Saira's age

ii. The age difference between Gita and Saira is 1 year. However Gita is not the oldest and Saira is not the youngest.

iii. There are no twins.

In what order were they born (oldest first)?

(GATE CSE 2010)

A. HSIG B. SGHI C. IGSH D. IHSG

Explanation: Age difference between Gita and Saira is 1, which means we look for GS or SG in the options. Options, b, c, d will have this. However, as Gita is not the oldest and Saira is not the youngest, options b and d will get eliminated further. Now, Hari + Gita age will be more than irfan + Sairas age as. G+1 = S.

H+G >I + S

H > I + S-G

H>I+1

This relation is not satisfied in option. Thus, possible option is C.

76. 5 skilled workers can build a wall in 20 days: 8 semi-skilled workers can build a wall in 25 days; 10 unskilled workers can build a wall in 30days. If a team has 2 skilled, 6 semi-skilled and 5 unskilled workers, how long will it take to build the wall?

A. 20 B. 18 C. 16 D. 15

Explanation : In one day, fraction of work done by 2 skilled, 6 semi-skilled and 5 unskilled workers

= 2*1/(5*20) + 6*1/(8*25) + 5*1/(10*30)

= 40/600 = 1/15

Therefore, number of days needed = 15

77. If Log(P) = (1/2)Log(Q) = (1/3)Log(R), then which of the following options is true?

(GATE MECH 2011)

A. $P^2=Q^3R^2$ B. $Q^2=PR$

C. $Q^2=R^3P$ D. $R=P^2Q^2$

Explanation: From the relation, we can write $P=Q^{1/2} = R^{1/3}$

$P = 1/Q^2 = 1/R^3$

$P = Q^2/R^3$

$Q^2 = PR^3$

78. A container originally contains 10 litre of pure spirit. From this container 1 litre of spirit is replaced with 1 litre of water. Subsequently, 1 liter of mixture is again replaced with 1 litre of water and this process is repeated one more time. How much spirit is now left in the container? **(GATE MECH 2011)**

A. 7.58 litre B. 7.84 litre

C. 7 litre D. 7.29 litre

Explanation:

After first with drawl of spirit left over = 9000ml Spirit + 1000 ml water

After first with drawl of spirit left over = 8100ml Spirit + 1900 ml water

After first with drawl of spirit left over = 7290ml Spirit + 2710 ml water

Directly we can give as: $10*((10–1)/10)^3 = 729/1000 = 7.29$

79. The variable cost (V) of manufacturing a product varies according to the equation V=4q, where q is the quantity produced. The fixed cost (F) of production of same product reduces with q according to the equation F=100/q. How many units should be produced to minimise the total cost (V+F)? **(GATE MECH 2011)**

A. 5 B. 4 C. 7 D. 6

Explanation:

Total cost T=(V+F)= 4q+100/q is minimises when q is 5

We can also find by minimising the equation as:

$dT/dq= 4 + (–1)*100*q^{-2} = 0$

q2 = 25

Therefore, q = 5

80. A transporter receives the same number of orders each day. Currently, he has pending orders (backlog) to be shipped. If he uses 7 trucks, then at the end of the 4^{th} day he can clear all the orders. Alternatively, if he uses only 3 trucks, then all the orders are cleared at the end of 10^{th} day. What is the minimum number of trucks required so that there will be no pending orders at the end of the 5^{th} day? **(GATE MECH 2011)**

A. 4 B. 5 C. 6 D. 7

Explanation: Let y be the orders pending currently and x be the number of orders per day. Therefore, y+4x orders are shipped by 7 trucks for 4 days. Also, y+10x orders are shipped by 3 trucks for 10 days. From these two statements, number of orders which are carried by a truck, assuming each truck carries same number of orders

(y+4x)/(4*7) = (y+10x)/(10*3)

Therefore, y= 80x

Also, one truck carries 3x orders approximately.

Number of trucks needed for completing the within 5 days = (80x+5x)/(5*3x)=5.6 ~ 6 trucks

81. Two policemen, A and B, fire once each at the same time at an escaping convict. The probability that A hits the convict is three times the probability that B hits the convict. If the probability of the convict not getting injured is 0.5, the probability that B hits the convict is: **(GATE AE 2012)**

A. 0.14 B. 0.22 C. 0.33 D. 0.40

Explanation: Given Probability of the convict not getting injured=0.5

Let x is the probability of B hitting the convict. Therefore (1-x) becomes probability of B not hitting the convict. As probability of A hitting convict is three times as that of B, the probability of A hitting convict becomes 3x, while probability of A not hitting the convict becomes (1-3x). Also, given the probability of the convict not getting injured is 0.5, probability of convict getting hit is 0.5; Therefore,

x + 3x = 0.5

x = 0.125

As both are fired, probability of only B is hitting can be said as= Probability of B hitting convict and Probability of A is not hitting.

82. The probability of three cards taken from a deck of cards being aces

A. 1/13 B. 1/5525

C. 1/5233 D. 1/14

Explanation: We know 4 ace's will be available in a deck. Therefore, all the three being ace's can be given as: 4/52 * 3/51 * 2/50= 1/5525

83. If 43% of people wear a seat belt while driving. If two people are called out at random, from a room which contains 100 people, what is the probability that both of them wear a seat belt?

A. 0.43 B. 0.63 C. 0.18 D. None

84. In every shipment of computers of number 100, 5 computers are observed to be defective. A person purchased one computer from current shipment for his son and the second computer for his younger daughter. What is the probability of both being defective.

A. 2/100 B. 1/100 C. 1/400 D. None

Answer: These are two independent events. Therefore, both being defective = Probability of first one being defective and Probability of second one being defective= 5/100 * 5/100=1/400

85. Blocks Problem (Source ICPC Archive)

The problem is to parse a series of commands that instruct a robot arm how to manipulate blocks that lie on a flat table. Initially there are n blocks on the table (numbered from 0 to n–1) with block bi adjacent to block bi_{+1} for all $0 \le i < n - 1$ as shown in the diagram below:

Figure: Initial Blocks World

The valid commands for the robot arm that manipulates blocks are:

- move *a* onto *b*

  where *a* and *b* are block numbers, puts block *a* onto block *b* after returning any blocks that are stacked on top of blocks *a* and *b* to their initial positions.

- move *a* over *b*

  where *a* and *b* are block numbers, puts block *a* onto the top of the stack containing block *b*, after returning any blocks that are stacked on top of block *a* to their initial positions.

- pile *a* onto *b*

  where *a* and *b* are block numbers, moves the pile of blocks consisting of block *a*, and any blocks that are stacked above block *a*, onto block *b*. All blocks on top of block *b* are moved to their initial positions prior to the pile taking place. The blocks stacked above block *a* retain their order when moved.

- pile *a* over *b*

  where *a* and *b* are block numbers, puts the pile of blocks consisting of block *a*, and any blocks that are stacked above block *a*, onto the top of the stack containing block *b*. The blocks stacked above block *a* retain their original order when moved.

- quit

terminates manipulations in the block world.

Any command in which $a = b$ or in which a and b are in the same stack of blocks is an illegal command. All illegal commands should be ignored and should have no affect on the configuration of blocks.

Assume that the following sequence of instructions are given with 10 blocks.

move 9 onto 1
move 8 over 1
move 7 over 1
move 6 over 1
pile 8 over 6
pile 8 over 5
move 2 over 1
move 4 over 9
quit

How many block stacks are having more than one block?

A. 0 B. 1 C. 2 D. 4

86. How many block stacks are having exactly one block?

A. 0 B. 1 C. 2 D. 4

87. How many block stacks are having no blocks at all?

A. 0 B. 4 C. 6 D. 2

88. In which stack, block 3 is seen at the end?

A. 0 B. 3 C. 4 D. 2

89. How many blocks gets moved to stack 5 during the instruction "pile 8 over 5"?

A. 2 B. 3 C. 4 D. None

90. Consider the following algorithm:

```
input n
print n
if n = 1 then STOP
if n is odd then n=3n+1
else n=n/2
GOTO step 2
```

Given the input 22, the following sequence of numbers will be printed 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1. That is, total 16 numbers including 22 and 1 are printed. Here, 6s is called as the *cycle-length* of *22*. Assuming that we have tested the above algorithm with all the natural numbers 1 to 10 (including). Largest cycle size out of all the trials

A. 20 B. 10 C. 30 D. 6

91. We have six sided dice marked 0, 1, 2, 3, 4 and 5 on each of their sides. What is the probability of total being six if two dice are thrown?

A. 18/36 B. 12/36 C. 7/36 D. 5/36

Explanation: We will first write down the number of ways the desired result six can occur. For example, if you want to roll a six, you could achieve that result five different ways: by rolling a one and a five, a two and a four, a three and a three, a four and a two, or a five and a one. We know in total 36 possibilities are there with two dice. Thus, the probability of total being six is given as 5/36.

92. We have six sided dice marked 0, 1, 2, 3, 4 and 5 on each of their sides. What is the probability of total being five if three dice are thrown?

A. 18/36 B. 18/216 C. 7/216 D. 5/36

93. Consider the following multiplication.

```
A       *  *  *  ×
B             *  =
-----------------
C    *  *  *  *
```

Where each of the figures of the three numbers A,B and C (indicated by symbol *) can be only 3 or 5 or 7. What are the three numbers A, B and C?

A. 155,5,775 B. 755,5,3775
C. 735,5,775 D. None

94. A set of three digit integers are framed by using the digits from the set {1, 2, 3} exactly once. If they are kept in ascending order then fourth number is

A. 123 B. 213 C. 231 D. 321

95. A total of 21 boxes are stacked like a pyramid with 6, 5, 4, 3, 2, 1 boxes from bottom to top. Assume that in the bottom six boxes 12, 21, 7, 53, 49 and 35 are stored. Value in any box can be calculated by adding the values of two boxes beneath it and applying modulus operator with 100 for the total. However, for the top most box value is calculated by multiplying the values of the two boxes beneath it. What is the value in the top box?

A. 9602 B. 9708
C. 9702 D. 9892

96. Siddhu must select three DVDs to record on an empty DVD. The available DVDs are labeled as K, O, S, T, V and W. Siddhu must follow the following conditions:

- K must be selected, S must be selected or both must be selected.
- O or V must be selected, but neither V nor S could be selected together with O.

Which one is a valid DVD set to be recorded?

A. K, O and S B. K, S and T
C. K, S and V D. O, S and V
E. O, T and V

97. If K and O were chosen, which item shows a valid set of DVD's that Siddhu could choose without breaking any condition?

A. S and V B. T and W
C. V and W D. S, W and T
E. V, W and T

98. If S is chosen, which DVD must not be chosen?

A. K B. O C. T D. V
E. W

99. If V was not chosen, which DVD pair must be chosen?

A. K and O B. K and T
C. K and W D. O and T
E. O and W

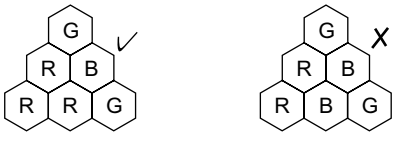
100. Which DVD pair must not be chosen at the same time?

A. K and O B. K and T
C. O and W D. T and W
E. V and W

101. A new colony has 4 north-south roads and 4 east-west roads laid out in a 4 × 4 grid. A security guard who is posted at the intersection of two roads can observe all activity along the length of all the roads which he can see. Minimum number of guards needed to make an eye on all the road stretches is

A. 3 B. 4 C. 5 D. 8

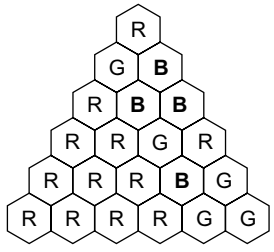
102. Hexagonal card board patches of color red, blue, and green are used to make a triangle shaped quilt. Each hexagon and two beneath should of same color or else all three should be of different colors. For instance, the following figure contains valid and invalid organisation of 5 card board pieces.



Find out how many blue patches are used in the following quilt.

A. 2 B. 3
C. 4 D. 5

Explanation:



103. Number of syllables in a word is approximated using the following rules:

- Each word has at least one syllable
- A vowel or a consecutive sequence of vowels is counted as one syllable. Vowels at the end of a word are ignored.

According to the above rules words "good", "ask", "study", "eerie" are considered to be having one syllable while the word "morning" is considered to be having two syllables. In the same fashion number of vowels in "advance australia fair"?

A. 5 B. 6 C. 7 D. 8

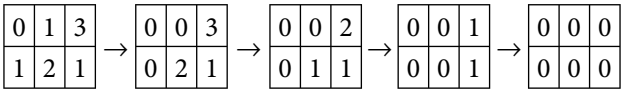
Explanation: See the underlined ones in "**a**dv**a**nce **au**stralia f**a**ir". Thus, five syllables are available in the giving string.

104. Your school is preparing for its annual fire drill. Each building in the school consists of a grid of classrooms, with the top-left classroom left empty. Each time the clock ticks another second, students may leave classrooms according to the following rules:

- If a classroom is adjacent to an empty room (or possibly several empty rooms), one and only one student may leave. This student leaves the building immediately (they do not spend any time walking through the empty rooms).
- If a classroom is not adjacent to any empty rooms, no students may leave.

Note that you can never have several students leaving a room at the same time (even if the room is adjacent to several different empty rooms). Rooms are only adjacent horizontally or vertically, not diagonally.

An example of a 2×3 grid of classrooms is shown below, with the initial numbers of students on the left hand side. It can be seen that the entire building is evacuated after four seconds.



Your building is slightly larger — a 4×4 grid of classrooms, as illustrated below. How many seconds does it take to evacuate your building?

0	2	5	3
1	4	3	2
4	2	5	4
3	1	4	2

A. 11 B. 13
C. 14 D. 15
E. 45

Explanation: See the following workout.

0	2	5	3
1	4	3	2
4	2	5	4
3	1	4	2

After First Second

0	1	5	3
0	4	3	2
4	2	5	4
3	1	4	2

After Second Second

0	0	5	3
0	3	3	2
3	2	5	4
3	1	4	2

After Third Second

0	0	4	3
0	2	3	2
2	2	5	4
3	1	4	2

After Fourth Second

0	0	3	3
0	1	3	2
1	2	5	4
3	1	4	2

After Fifth Second

0	0	2	3
0	0	3	2
0	2	5	4
3	1	4	2

After Sixth Second

0	0	1	3
0	0	2	2
0	1	5	4
2	1	4	2

After Seventh Second

0	0	0	3
0	0	1	2
0	0	5	4
1	1	4	2

After Eighth Second

0	0	0	2
0	0	0	2
0	0	4	4
0	0	4	2

After Ninth Second

0	0	0	1
0	0	0	1
0	0	3	4
0	0	3	2

After Tenth Second

0	0	0	0
0	0	0	0
0	0	2	4
0	0	2	2

After Eleventh Second

0	0	0	0
0	0	0	0
0	0	1	3
0	0	1	2

After Twelth Second

0	0	0	0
0	0	0	0
0	0	0	2
0	0	0	2

After Thirteenth Second

0	0	0	0
0	0	0	0
0	0	0	1
0	0	0	1

After Fourteenth Second

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

105. A rising sequence is a list of numbers where each number is greater than the sum of all the numbers before it. For example, 1;2;4;12;22 is a rising sequence because 1 < 2, 1+2 < 4, 1+2+4 < 12 and 1+2+4+12 <

22. On the other hand, 1;2;4;6;17 is not a rising sequence because 1+2+4 6< 6. You are given the following list of numbers: 3; 5; 8; 11; 25; 30; 45; 50; 60; 95:

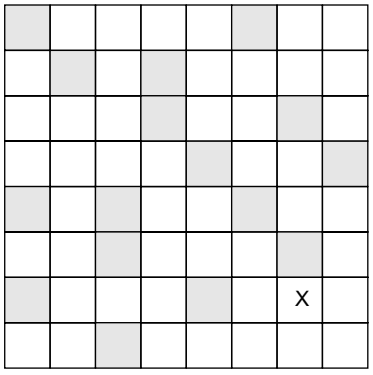
Your task is to form the longest possible rising sequence using numbers from this list. How many numbers are in your rising sequence?

A. 4 B. 5 C. 6 D. 7

E. 8

Explanation: 3, 5, 11, 25, 45, 95.

106. A token (marked 'X' in the diagram) is in a maze. You may move the token around according to the following rule: in each move the token may travel any distance either horizontally or vertically, but it cannot pass over or stop on a shaded square.



For example, from its starting position the token could travel either one square left, one square right, or one square down in a single move. To reach any other square would require more than one move.

What is the minimum number of moves that you need to ensure that the token can reach any white square from its starting position?

A. 9 B. 10 C. 11 D. 12

E. 15

Explanation: See the figure which marks each white cell with number of moves needed to reach it.

We find that minimum of 11 moves are needed to reach any cell.

3	4	4		8	8	7	8
3	4		10	9		7	
3		X	11		5	6	6
2	2	1		3	4		7
3		1	2	2		8	7
2	2	1		3		8	7
	3		5		9		7
4	3	4	4		8	8	7

107. The escape has been planned but the prisoners have the wrong instructions. The prisoner in cell 1 should have instructions A, and the prisoner in cell 2 should have instructions B, and so on. Each day there is an opportunity for prisoners in adjacent cells to swap instructions, but more than one swap per day would be too risky. For instance if the instruction order was CDBA, how many days of swapping are needed to get correct order ABCD.

A. 4 B. 5 C. 7 D. 8

E. 6

Explanation: First 2 and 3 exchange their instructions D and B. Thus, current instructions become CBDA. Now, 3 and 4 exchanges their instructions. Thus, current instructions become CBAD. Now, 1 and 2 exchanges their instructions. Thus, current instructions become BCAD. Now, 2 and 3 exchanges their instructions. Thus, current instructions become BACD. Now, 1 and 2 exchanges their instructions. Thus, current instructions become ABCD.

108. A circus is designing an act consisting of a tower of people standing atop one another's shoulders. For practical and aesthetic reasons, each person must be both shorter and lighter than the person below him/her. Given the heights and weights of each person in the circus, what is the largest possible number of people in such a tower?

(50,100) (65 100) (70 150) (56 90) (75 190) (60 95) (68 110)

A. 4 B. 5 C. 6 D. 3

Explanation: The longest tower is length 6 and includes from top to bottom:

(56,90) (60,95) (65,100) (68,110) (70,150) (75,190)

109. A department has n research students. An array A contains when a student starts and stops on a day. That is, the i-th student starts working at time A[i][1] and stops working at time A[i][2]. Determine the greatest number of students that are working simultaneously. All values of the array refers to hours over 0 to 24 scale.

2	8
1	3
5	7
2	4
3	6
1	4
3	8
2	5
2	8
6	9
9	10

A. 6 B. 8 C. 9 D. 7

110. Three friends divided some bullets equally. After all of them shot 4 bullets the total number of bullets remaining is equal to the bullets each had after division. Find the original number divided.

A. 15 B. 18 C. 9 D. None

111. There is a 50m long army platoon marching ahead. The last person in the platoon wants to give a letter to the first person leading the platoon. So while the platoon is marching he runs ahead, reaches the first person and hands over the letter to him and without stopping he runs and comes back to his original position. In the mean time the whole platoon has moved ahead by 50m. How much distance did the last person cover in that time? Assume that he ran the whole distance with uniform speed.

A. 100 B. 120.71 C. 130.71 D. None

112. There are 3 persons Ram, Ravi and Abhi. On some day, Ram gave money to Ravi and Abhi as much as they already had. After few days, Ravi gave three times the money to Ram and Abhi as they already have. After few days Abhi did the same thing as Ravi did. At the end of this transaction each one of them had 32. The money each originally had is

A. 2,43,433 B. 4,04,438

C. 4,93,710 D. None

113. A colony secretary ordered a workman to assign a three digit unique numbers (001 to 100) for each of the hundred houses. Which digit appears more often over all?

A. 1 B. 2

C. 0

D. All occurs with same frequency

114. A colony secretary ordered a workman to assign a three digit unique numbers (001 to 100) for each of the hundred houses. The digit which appears 21 times is

A. 1 B. 2

C. 0

D. All occurs with same frequency

115. There are 22 gloves in a drawer: 5 pairs of red gloves, 4 pairs of yellow, and 2 pairs of green. You select the gloves in the dark and can check them only after a selection has been made. What is the smallest number of gloves you need to select to have at least one matching pair in the best case?

A. 2 B. 3 C. 4 D. 5

Explanation: Do remember we are not asking the probability. Thus, answer is 2.

116. There are 22 gloves in a drawer: 5 pairs of red gloves, 4 pairs of yellow, and 2 pairs of green. You select the gloves in the dark and can check them only after a selection has been made. What is the smallest number of gloves you need to select to have at least one matching pair in the worst case?

A. 2 B. 12 C. 10 D. None

Explanation: In total 22 gloves are available. Do remember that gloves are different for each hand. In total, we have 11 pairs of gloves. Thus, if we select 12 socks then there will be at least one pair in them. If we take 11 gloves they can be all left hand type or right hand type. Thus, zero pairs we may find in 11 gloves.

117. A mother has washed five distinct size socks of her children and kept in a box. Two socks were taken away by rats. How many children she can make ready with socks in the best and worst cases. Do calculate the probability of best case and worst case, respectively.

Explanation: Best case 4 children and worst case 3 children. Best case occurs if both the missing socks are of same pair. Thus, there will be 4 pairs left. So, she can prepare 4 children's. Now let us try to calculate their probabilities. Probability of selecting 2 socks from 10 is $10C_2$=45. As mentioned about best case situation arises when the missing socks are of a pair. As we have 5 pairs, then the best case may happen in 5 ways. Thus, probability of best case is 5/45=1/9. The probability of worst case is 1–1/9=8/9.

118. The radius of a cylindrical memento is 1 centimetre and the height is 2 centimetre. A fancy paper is proposed to cover this memento along its curved surface. The length of the rectangle sized paper needed. The largest side length of the required rectangle shaped paper sheet is:

A. 2π cm B. 4π cm C. π^2 cm D. $2\pi^2$ cm

Explanation: As we plan to cover only cylindrical surface, perimeter of the cylinder (2p centimetre) and height of the cylinder, i.e., 2 centimetre becomes the dimensions of the paper sheet. Thus, answer is a.

119. See the following figure cleansers bottles availability in the market. Which bottle of cleanser should a person buy to get the best quality for the least amount of money?

A. Regular size, 1 qt 25′
B. Family size, ½ gal. 38′
C. Giant economy size, 1 gal. 60′
D. All cost the same per quart

Explanation: There are 4 quarts in a gallon or 2 quarts in a half-gallon, so the family size bottle at 38′ costs 38′ : 2 = 19′ per quart. The giant economy size bottle costs 60′ : 4 = 15′ per quart. Therefore the giant economy size bottle is the buy that gets a person the greatest amount for the least money.

120. The difference between one-half of a number and one-fifth of it is 561. The number is:

A. 168 B. 2805 C. 1870 D. 5610
E. 187

121. The tv chart for the weather forecast shows:

What does a probability of 30% that it will be raining tomorrow mean?

A. 30% of 12 hours is about 3½ hours, so we will have 3½ hours of rain tomorrow.
B. 30% is less than ½, so we will have rain tomorrow for less than half the day.
C. 30% is less than 50% so more likely than not we will have a dry day tomorrow.
D. You cannot tell because the weather forecast is often wrong

122. A license plate has 3 letters and 4 digits (e.g. ABC 3456). How many different licenses can be formed if repetition of letter or number is not allowed?

A. 782000 B. 7820000 C. 7824000 D. None

Explanation: Let us apply the fundamental principle of counting,

Position	Letter 1	Letter 2	Letter 3	Digit 1	Digit 2	Digit 3	Digit 4
number of choices	26 (can pick any letter)	25 (1 letter is used already)	24 (can't use any of the previous 2 letters)	10	9 (can't use any of the previous digits)	8 (can't use any of the previous digits)	7 (can't use any of the previous digits)

Thus, We see that there are 26.25.24.10.9.8.7 = 7824000 possible outcomes.

123. If the Prob.(Student has Visa Card) = 0.8, and Prob.(Student has MasterCard) = 0.15, and Prob.(Student has Both Cards) = 0.1, then the probability that a student does not have an MC is _____ and the probability that a student has neither card is _____

A. 0.8, 0.2 B. 0.75, 0.25
C. 0.8, 0.8 D. 0.85, 0.15

124. The sum of the deviations from the mean; namely $\sum_{i=i}^{n}(x_i - \bar{x})$, has always the numerical value _____.

A. >0 B. <0 C. 0 D. None

125. A population consists of 500 elements. We want to draw a simple random sample of 50 elements from this population. On the first selection, the probability of an element being selected is

A. 0.100 B. 0.010
C. 0.001 D. 0.002

126. Given the following guess what is ????

2{38}3
4{1524}5
6{3548}7
8{????}9

Answer: 6380

If we observe, we have in between curly braces first number and last number squares after subtracting 1.

A, B, and C are three numbers, Let
@(A, B)= Average of A and B
*(A, B)=Product of A and B
/(A, B)=A divided by B

127. If A=2 and B=4 the value of @(/ (*(A,B),B),A) would be

A. 2 B. 4 C. 6 D. 16

128. Sum of A and B is given by

A. *(@(A, B), 2) B. /(@(A,B),2)
C. @(*(A,B),2) D. @(/(A,B),2

129. Let x<0, 0<y<1, Z>1 which of the following is false:

A. (x^2-z^2)has to be positive.
B. yz can be less than one.
C. xy can never be zero
D. (y^2-z^2) is always negative

130. If A's income is 25% less than B's, by what % is B's income greater than that of A ?

A. 35% B. 25%

C. 30% D. None of these

131. Rama and Abhi have an 8-litre bottle completely filled with Beer that they wish to divide *equally* between them. They also have two empty bottles with capacities of 5 and 3 liters respectively. They want to divide the beer equally in the *fewest* number of pours from one bottle to another. (*There are no volume marking on the cans*). How many minimum numbers of pours are needed?

A. 3 B. 2 C. 4 D. None

Explanation: You need more than 4 pourings.

132. It is given that the platoon and the last person moved with uniform speed. Also, they both moved for the identical amount of time. Hence, the ratio of the distance they covered—while person moving forward and backword—are equal. Let's assume that when the last person reached the first person, the platoon moved X metre forward. Thus, while moving forward the last person moved (50+X) metre whereas the platoon moved X metre.

Similarly, while moving back the last person moved [50-(50-X)] X metre whereas the platoon moved (50-X) metre.

Now, as the ratios are equal,

(50+X)/X = X/(50-X)

(50+X)*(50-X) = X*X

Solving, X=35.355 metre

Thus, total distance covered by the last person

= (50+X) + X

= 2*X + 50

= 2*(35.355) + 50

= 120.71 metre

Note that at first glance, one might think that the total distance covered by the last person is 100 metre, as he ran the total length of the platoon (50 metre) twice. true, but that's the relative distance covered by the last person i.e. assuming that the platoon is stationary.

We can solve this problem by using simultaneous equations also. However, we shall use back tracking to solve the same. We know that at the end each of them are having 32 each. That is, after Abhi giving three times of the money what they have to Ram and Ravi, their amounts became 32. Which indicates they will be having 8 before Abhi paid. Thus, Abhi will be having 32 + 2*24 = 80 be he paid. Similarly, before that Ravi has paid to others. Thus, before Ravi paid they might be having 2, 74 and 20, respectively. Initially, Ram has paid to others what the amount they have. Thus, initially all three will be having 49, 37 and 10 respectively.

133. What will be the maximum sum of 44, 42, 40, ... ?

A. 502 B. 504 C. 506 D. 500

Explanation: We are asking for maximum sum. Thus, we need not required to consider terms 0 and below 0. Thus, we need to calculate the sum of 44, 42, 40, ….2. Number of terms are 22 and the difference value is 2. Therefore, sum becomes 22/2(2 + 44) = 506

134. The access code for a house consists of four digits but the first digit must be 3,4 or 5 and it cannot end in 000. How many possible combinations are there?

A. 3000 B. 2997 C. 2001 D. None

Explanation: The number of 4 digit numbers starting with 3, 4 or 5 are:

3 * 10 * 10 * 10 = 3,000.

There are 3 special cases that aren't allowed: 3000 4000 and 5000

3,000 – 3 = 2,997

135. The cost function for a product in a firm is given by $5q^2$, where q is the amount of production. The firm can sell the product at a market price of Rs. 50 per unit. The number of units to be produced by the firm such that the profit is maximised is **(GATE 2012)**

A. 5 B. 10 C. 15 D. 25

Explanation: $5q^2$ is the cost function. As each item is sold at Rs. 50/-, the profit function becomes $5q^2$–50q.

To maximise this, we calculate derivative and equate 0.

10q–50 = 0

There fore q = 5.

136. Wanted Temporary, Part-time persons for the post of Field Interviewer to conduct personal interviews to collect and collate economic data. Requirements: High School-pass, must be available for Day, Evening and Saturday work. Transportation paid, expenses reimbursed.

Which one of the following is the best inference from the above advertisement?

A. Gender-discriminatory

B. Xenophobic

C. Not designed to make the post attractive

D. Not gender-discriminatory

137. A political party orders an arch for the entrance to the ground in which the annual convention is being held. The profile of the arch follows the equation $y = 2x - 0.1x^2$ where y is the height of the arch in metre. The maximum possible height of the arch is

A. 8 metre B. 10 metre
C. 12 metre D. 14 metre

Explanation: Calculate dy/dx and calculate x.

2 – 0.2x = 0

Therefore, x = 10

If we substitute x = 10 in the y equation, we get y = 10.

138. An automobile plant contracted to buy shock absorbers from two suppliers X and Y. X supplies 60% and Y supplies 40% of the shock absorbers. All shock absorbers are subjected to a quality test. The ones that pass the quality test are considered reliable. Of X's shock absorbers, 96% are reliable. Of Y's shock absorbers, 72% are reliable. The probability that a randomly chosen shock absorber, which is found to be reliable, is made by Y is

A. 0.288 B. 0.334
C. 0.667 D. 0.720

Explanation: Probability of a sample from X = P(X) = 0.6

Probability of a sample from Y = P(Y) = 0.4

Probability of reliable sample from X = P(R/X) = 0.96

Probability of reliable sample from Y = P(R/Y) = 0.72

Probability of a reliable sample = P(R) = P(R/X)*P(X) + P(R/Y)*P(Y) = 0.96*0.6 + 0.72*0.4 = 0.864

Probability of a reliable sample coming from Y, P(Y/R) is required to be calculated. Using Bayes theorem

P(Y/R) = P(R/Y)*P(Y)/P(R) = 0.72*0.4/0.864 = 0.334

139. Which of the following assertions are correct?

P: Adding 7 to each entry in a list adds 7 to the mean of the list

Q: Adding 7 to each entry in a list adds 7 to the standard deviation of the list

R: Doubling each entry in a list doubles the mean of the list

S: Doubling each entry in a list leaves the standard deviation of the list unchanged

A. P, Q B. Q, R
C. P, R D. R, S

140. Given the sequence of terms, AD CG FK JP, the next term is

A. OV B. OW
C. PV D. PW

141. 25 persons are in a room. 15 of them play hockey, 17 of them play football and 10 of them play both hockey and football. Then the number of persons playing neither hockey nor football is: **(GATE 2010)**

A. 2 B. 17
C. 13 D. 3

142. If 137 + 276 = 435 how much is 731 + 672?

A. 534 B. 1403
C. 1623 D. 1513

143. Hari (H), Gita (G), Irfan (I) and Saira (S) are siblings (i.e. brothers and sisters). All were born on 1st january. The age difference between any two successive siblings (that is born one after another) is less than 3 years. Given the following facts:

i. Hari's age + Gita's age > Irfan's age + Saira's age
ii. The age difference between Gita and Saira is 1 year. However Gita is not the oldest and Saira is not the youngest.
iii. There are no twins.

In what order were they born (oldest first)?

A. HSIG B. SGHI
C. IGSH D. IHSG

144. 5 skilled workers can build a wall in 20 days: 8 semi-skilled workers can build a wall in 25 days; 10 unskilled workers can build a wall in 30 days. If a team has 2 skilled, 6 semi-skilled and 5 unskilled workers, how long will it take to build the wall?

A. 20 B. 18
C. 16 D. 15

145. Modern warfare has changed from large scale clashes of armies to suppression of civilian populations. Chemical agents that do their work silently appear to be suited to such warfare; and regretfully, there exist people in military establishments who think that chemical agents are useful tools for their cause.

Which of the following statements best sums up the meaning of the above passage?

A. Modern warfare has resulted in civil strife.
B. Chemical agents are useful in modern warfare.
C. Use of chemical agents in warfare would be undesirable
D. People in military establishments like to use chemical agents in war.

146. Given digits 2,2,3,3,4,4,4,4 how many distinct 4 digit numbers greater than 3000 can be formed?

A. 50 B. 51
C. 52 D. 54

Questions Related to Verbal Ability

147. Choose the most appropriate word from the options given below to the complete the following sentence:

His rather casual remarks on politics _____ his lack of seriousness about the subject. **(GATE 2010)**

A. Masked B. Belied
C. Betrayed D. Suppressed

148. Which of the following options is closest in meaning to the word Circuitous. **(GATE 2010)**

A. Cyclic B. Indirect
C. Confusing D. Crooked

149. Choose the most appropriate word from the options given below to complete the following sentence:

If we manage to _____ our natural resources, we would leave a better planet for our children. **(GATE 2010)**

A. Uphold B. Restrain
C. Cherish D. Conserve

150. The question below consists of a pair of related words followed by four pairs of words. Select the pair that best expresses the relation in the original pair. **(GATE 2010)**

Unemployed: Worker

A. Fallow: land B. Unaware: sleeper
C. Wit: jester D. Renovated: house

151. Choose the most appropriate alternative from the options given below to complete the following sentence:

Despite several _____ the mission succeeded in its attempt to resolve the conflict. **(GATE 2012)**

A. Attempts B. Setbacks
C. Meetings D. Delegations

152. Which one of the following options is the closest in meaning to the word given below? **(GATE 2012)**

Mitigate

A. Diminish B. Divulge
C. Dedicate D. Denote

153. Choose the grammatically incorrect sentence: **(GATE 2012)**

A. They gave us the money back less the service charges of Three Hundred rupees.
B. This country's expenditure is not less than that of Bangladesh.
C. The committee initially asked for a funding of Fifty Lakh rupees, but later settled for a lesser sum.
D. This country's expenditure on educational reforms is very less.

154. Choose the most appropriate alternative from the options given below to complete the following sentence:

Suresh's dog is the one _____ was hurt in the stampede. **(GATE 2012)**

A. That B. Which C. Who D. Whom

155. Choose the most appropriate word(s) from the options given below to complete the following sentence. **(GATE 2011)**

I contemplated _____Singapore for my vacation but decided against it.

A. To visit B. Having to visit
C. Visiting D. For a visit

Explanation: Contemplate is a transitive verb and hence is followed by a gerund. Hence the correct usage of contemplate is verb+ ing form.

156. Choose the most appropriate word from the options given below to complete the following sentence.

If you are trying to make a strong impression on your audience, you cannot do so by being understated, tentative or _____. **(GATE 2011)**

A. Hyperbolic B. Restrained
C. Argumentative D. Indifferent

157. Choose the word from the options given below that is most nearly opposite in meaning to the given word:

Amalgamate **(GATE 2011)**

A. Merge B. Split
C. Collect D. Separate

158. Which of the following options is the closest in the meaning to the word below? **(GATE 2011)**

Inexplicable

A. Incomprehensible
B. Indelible
C. Inextricable
D. Infallible

Explanation: Inexplicable means not explicable; that cannot be explained, understood, or accounted for. So the best synonym here is incomprehensible.

159. Which one of the following options is the closest meaning to the word given below? **(GATE 2013)**

Nadir

A. Highest B. Lowest
C. Medium D. Integration

160. Complete the sentence:

Universalism is to particularism as diffuseness is to _____. **(GATE 2013)**

A. Specificity B. Neutrality
C. Generality D. Adaptation

161. Were you a bird, you _____ in the sky. **(GATE 2013)**

A. Would fly B. Shall fly
C. Should fly D. Shall have flown

162. Choose grammatically incorrect sentence.

A. He is of Asian origin
B. They belonged to Africa
C. She is an European
D. They migrated from India to Australia

ANSWER KEY

1. ?	**2.** ?	**3.** ?	**4.** ?
5. ?	**6.** ?	**7.** ?	**8.** ?
9. ?	**10.** C	**11.** B	**12.** ?
13. ?	**14.** ?	**15.** ?	**16.** ?
17. ?	**18.** A	**19.** ?	**20.** A
21. ?	**22.** ?	**23.** ?	**24.** A
25. C	**26.** B	**27.** B	**28.** C
29. C	**30.** D	**31.** C	**32.** C
33. C	**34.** D	**35.** B	**36.** D
37. B	**38.** B	**39.** B	**40.** C
41. D	**42.** D	**43.** D	**44.** C
45. C	**46.** B	**47.** B	**48.** C
49. B	**50.** A, C	**51.** B	**52.** B
53. B	**54.** D	**55.** B	**56.** C
57. ?	**58.** C	**59.** ?	**60.** ?
61. ?	**62.** ?	**63.** ?	**64.** B, C
65. C, D	**66.** C	**67.** C	**68.** B
69. B	**70.** C	**71.** A	**72.** A
73. D	**74.** C	**75.** C	**76.** D
77. C	**78.** D	**79.** A	**80.** C
81. A	**82.** B	**83.** C	**84.** C
85. C	**86.** C	**87.** C	**88.** B
89. B	**90.** B	**91.** D	**92.** B
93. B	**94.** C	**95.** C	**96.** C
97. B	**98.** B	**99.** B, C	**100.** D
101. B	**102.** C	**103.** A	**104.** C
105. C	**106.** C	**107.** B	**108.** C
109. D	**110.** B	**111.** B	**112.** C
113. C	**114.** A	**115.** A	**116.** B
117. ?	**118.** A	**119.** C	**120.** C
121. C	**122.** C	**123.** D	**124.** C
125. D	**126.** ?	**127.** A	**128.** A
129. A	**130.** D	**131.** D	**132.** ?
133. C	**134.** B	**135.** A	**136.** D
137. B	**138.** B	**139.** C	**140.** A
141. D	**142.** C	**143.** B	**144.** C
145. D	**146.** B	**147.** C	**148.** B
149. D	**150.** A	**151.** B	**152.** A
153. D	**154.** ?	**155.** C	**156.** B
157. B	**158.** A	**159.** B	**160.** A
161. A	**162.** C		

Note

For all those questions with ? as answer, explanations are given after the question itself.

CHAPTER ELEVEN

Model Papers for GATE Examination (with Solutions and Explanations)

Test 1

Section A

1. In ___ system a part of a file is stored in a disk and another part is stored in another disk and vice versa.
A. FAT B. I-node based FS
C. RAID D. None

2. If a program is reading from a file and writing into a file after modifying and employing double buffering then number of outstanding disk requests are __
A. 1 B. 2
C. 4 D. None

3. Advantage of accessing memory through the file interface
A. Improved speed
B. Flexibility
C. Trusted processes access to main memory
D. None

4. File pointers in most Unix systems is
A. 4 bits B. 32 bits
C. 4 bytes D. B & C
E. None

5. File system descriptor is
A. Boot block B. Super block
C. I-node blocks D. None

6. Is a machine language instruction interruptible?
A. No B. Yes
C. Commonly No. But on some machines they are interruptible.
D. None

7. Jacketing
A. Is used to convert blocking system call to non-blocking.
B. Is used to avoid blocking threads.
C. Contains jacket routine code which checks I/O device is busy or not.
D. All

8. Where does the Swap space reside?
A. RAM B. ROM
C. Disk D. On-chip cache

9. Which of the following may give compile time error?
A. char ***s="Hello",**p; *p=s;
B. char ***s="Hello",**p; p=s;
C. char ***s="Hello",**p; *p=*s;
D. None

10. Tail recursion
A. All of its recursive calls are tail recursive
B. Its last statement is recursive call and its return value is not part of any expression
C. While rewinding phase tail recursive functions will not do anything
D. All

11. Number of zero element, subsets a will be having is ___
A. Zero B. n

C. One D. None

12. Find odd one out

A. IBM Mainframe B. Sun Sparc

C. Motorola(MAC) D. IBM PC

13. Class invariant is

A. Collection of Boolean expressions that are always true for objects of the class

B. Static member variable of a class

C. Static member function of a class

D. None

14. In user mode

A. PSW bit 0 value is 1

B. Instructions that modify control registers are not legal and cause a program error

C. All addresses must be less than bound

D. All

15. Find out correct one

A. L1 cache will be in microprocessor

B. L2 cache will be in outside microprocessor

C. Both A & B

D. None

16. Harward architecture

A. Separate caches for data & instructions

B. Same caches for data & instructions

C. Multiple processors

D. None

17. For I = 1 to 1000 do

x=x+I

Assume x is stored in cache currently. Then __ is preferred

A. Write-through B. Write-back

C. Both D. None

18. Traditional LANs example, Ethernet Bus, Token Ring use

A. Circuit Switching B. Packet Switching

C. Frame Relay D. Packet Broadcast

19. A device has two IP addresses. One address is 192.123.46.219. The other address can be

A. 192.123.46.220 B. 192.123.46.0

C. 192.123.47.219 D. None

20. Gateway

A. Protocol conversion B. Packet re-sizing

C. Data rate adjustment D. All

21. Ethernet addresses are

A. 2 bytes B. 6 bytes

C. Both A & B D. None

22. Television channels are 8MHz wide. How many bps can be sent if four-level digital signals are used. Assume cannel is ideal.

A. 32Mbps B. 32MB/s

C. 16Mbps D. 16MB/s

23. CRC polynomial used in Ethernet

A. CRC-12 B. CRC-16

C. CRC-32 D. None

24. With respect to compilers and interpreters which of the following is true ?

A. Syntax analysis is done by compilers but not by interpreters.

B. A program that can be compiled can be interpreted .

C. Interpreters are always written in a high level language but compilers are written in high or assembly language.

D. None

25. Define two operations

T1: complement any two bits

T2 exchange any two bits

Starting with 000000 which of the following cannot be generated using sequence T1 and T2?

A. 110 000 B. 001 001

C. 100 001 D. 000 010

E. 001 100

26. A 2-3 tree is a tree in which each node has either 2 or 3 sons, except the leaves and also has equal path length from root to any leaf. The number of leaves is 9. Then the number of internal nodes can be

A. 7 B. 8

C. 9 D. 6

E. 3

27. Queue with permitted operations

(i) insertion at tail of queue

(ii) output the head of queue

(iii) place the head at the tail

For an input string of 1 2 3 4 5 6 which of the following outputs are possible?

(i) 3 4 1 2 6 5 (ii) 5 2 1 3 4 6

(iii) 5 6 1 2 3 4

A. I only B. II only

C. III only D. I & III

E. I, II & III

28. Grammar G: S → Aa|bB|Sa

A → b|a

B → Aa

Which of the following is true?

A. String aaaa proves G is ambiguous
B. String baaa proves that G is ambiguous
C. String abab proves that G is ambiguous
D. Ambigous but not detectable from above given procedure
E. Not ambiguous

29. A computer represents floating point exponents using the excess 64 to base 10 form (using 7 bits). If two such exponents are added using a seven bit adder, what is the modification in the sum required to ensure that the resulting exponent is in excess 64 form?

A. Generate end around carry and add
B. Generate end around borrow and subtract
C. Complement MSB
D. No change is required

30. The following are coded representations of the numbers 2 ,4 , 6, 8. Which is not a Gray code?

A. 1110,1010,1111,0101 B. 1001,0011,1111,1100
C. 1101,1000,0010,0111 D. None

31. All the following statements are true except :

A. Concurrent processes must work on different processors.
B. If the result of two parallel tasks is independent of their speed then the result has a deterministic solution.
C. A deadlock is a circular wait where two or more processors are waitig for others to release resources.
D. None

32. Sometimes the object module produced by a compiler includes information (from the symbol table) mapping all source program names to their address. The most likely purpose of this information is

A. For use as input to a debugging aid.
B. To increase the run time efficiency of the program.
C. For the reduction of the symbol table space needed by the compiler.
D. To tell the loader where each variable belongs.
E. To tell the OS what to call the variables.

33. Given the grammer

$S \rightarrow S+S \mid S*S \mid a$

How many distinct parse trees exist for the expression a + a*a + a?

A. 4 B. 5
C. 6 D. 9
E. 8

34. If a,b,c occur with equal probability, which of the following is a valid Huffman coding of these symbols?

A. a=1, b=0, c= 11 B. a=0, b=111, c=000
C. a=0, b= 10, c= 11 D. None of the above

35. In an AVL tree with 1000 nodes, path length is the length of a path from root to a leaf node is

A. At least one path has path length >100.
B. All paths have path length >100.
C. No path has path length >100.
D. Cannot be determined from the above info.

36. The boolean expression xyz′ + xy′z + x′yz + xyz is equivalent to

A. xy + yz + zx
B. xy+ yz
C. xyz+xyy
D. (xy)′ + (yz)′ + (zx)′ + (xyz)′

37. Which of the following exhibit locality of reference?

I. Sequential processing of arrays
II. Symbol table using hashing
III. Collection of garbage in linked memory

A. None B. I only
C. I and II D. III
E. I, II and III

38. There is a relational schema which has k attributes. The domain of each attribute consists of exaclty 2 elements. A table is defined as subset of tuples where in each tuple, a value is defined for each of the k attributes. The minimum value of k needed for the number of distinct tables to exceed 10^9 is

A. 5 B. 9
C. 17 D. 512
E. 1024

39. Which of the following is not done when an interrupt occurs?

A. Save the starting address of the executing procedure
B. Save the address of the current instruction
C. Detect the cause of the interrupt
D. Save the values of the registers
E. Make a call to the kernel

40. If n is a power of 2, then the minimum number of multiplications needed to computer a^n.

A. lg n B. sqrt(n)
C. n–1 D. n

41. The language accepted by a pushdown automation in which the stack is limited to 10 items is best described as

A. Context free B. Regular
C. Deterministic context free
D. Recursive

42. According to 4th normal form
A. Make sure to have multivalued dependencies
B. Make sure that there are no hidden dependencies
C. Make sure to remove transitive dependencies
D. None

43. bplus tree preferred over hashing in
A. Read one record
B. Read next, read all, reorganize records
C. Modify records
D. Insert, delete records

44. At a particular time the value of the counting semaphore is 7. Then 20 P operations and n V operations are done which makes the final result of semaphore variable as 5. Then the value of n is
A. 15 B. 18
C. 22 D. 13

45. When will be the paths from one node to every other node will become V?
A. Directed
B. Only one cycle and all nodes are involved.
C. No self cycles
D. All

46. Adjacency matrix of a graph is having 1's in off diagonal while remaining all elements are same. Then the graph
A. Contains all isolated nodes
B. V/2 connected components
C. If odd no of nodes are there then V/2+1 connected components
D. If odd no of nodes are there then one isolated node will be available
E. B & C & D

47. If T is a full binary tree with r internal nodes then
A. then it will have r+1 terminals
B. 2r+1 total vertices
C. Both A & B D. None

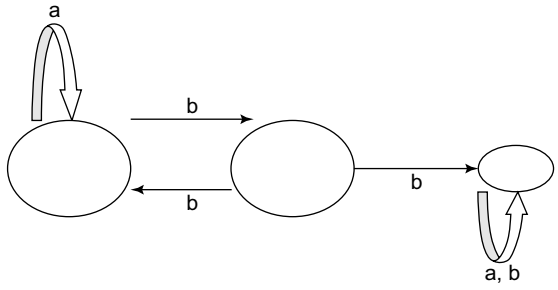
48. For a connected graph with e edges and n vertices
A. n cycles
B. n-1 branches in any spanning tree
C. n connected components
D. None

49. A graph is connected if it contains
A. One connected component
B. More than one connected component
C. If it has a spanning tree
D. Both A & B

50. A minimal DFA that is equavalent to an NDFA has
A. Always more states
B. Always lesser number of states
C. Always 2^n states
D. Sometime more states
E. None

Section B

51. According to the following transition diagram find whether strings aaab, bbbbaa are valid or not.



A. aaab B. bbbbaa
C. Both A & B D. None

52. The minimal finite automata accepting set of all strings over {0,1} ending 000 or 111 has
A. 5 states B. 6 states
C. 7 states D. 8 states
E. None

53. $E = \overline{Y} + \overline{X}\,\overline{Z}$ is a Boolean expression. It is equivalent to
A. $\Sigma(0,1,2,4,5)$ B. $\Pi(3,6,7)$
C. Both A & B D. None

54. The rank of a matrix

$$\begin{matrix} -1 & 0 & 0 \\ 2 & -3 & 0 \\ 1 & 4 & 2 \end{matrix}$$

A. 0 B. 1
C. 2 D. None

55. The n'th difference of polynomial of degree n is
A. Constant B. Zero
C. Variable D. None

56. A connected planar graph with n vertices and e edges has ___ regions.
A. n – e+2 B. e – n+2
C. n – e D. e – n

57. A covering of an n-vertex graph will have atleast __ edges.

A. n–2 B. [n/2]
C. [3n/2] E. 2n

58. A sender has a sliding window of size 15. The first 15 frames are sent. The receiver sends an ACK for 10. How many spaces does the receiver widnow expand?
A. 5 B. 9
C. 10 D. 15

59. Which of the following can be used for zeroing out alternate bits of a 16-bit number using the AND operation?
A. 0101 B. AAAA
C. EEEE D. FFFF
E. 1010

60. If n is even, and assuming that all A[i] are distinct, what does the execution of the code below result in?

```
for (i = 0; i < n; i++)
A[i] = A[n+1-i];
```

A. It results in 2 copies of each value data
B. The values remain unchanged.
C. The array reverses D. None of the above

61. In a packet delivery system, a packet is retransmitted when lost, till it is successfully transmitted. If the probability of loss is C, and retransmissions are independent, what is the expected number of unsuccessful transmission before a successful transmission?
A. 1/(1–C) B. C/(1–C)
C. 1/(1–C)^2 D. C
E. 1–C

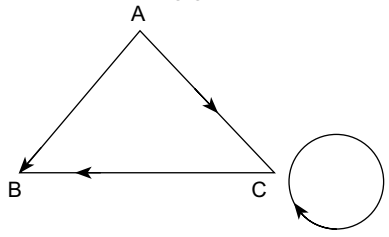
62. z = xyz + xyz + xyz
Which of the following is correct?
A. z is true if only 2 variables are true.
B. z is true if one or more variables are true
C. z is false if one or more variables is false
D. z is false is if 2 variables are false

63. Which string does a(b*c)* reject?
A. abbccc B. abcbcbc
C. abbcc D. acbc

64. S → A
A → AA | 0
How many derivations of 00000 are possible?
A. 5 B. 14
C. 24 D. 29
E. 54

65. Find the number of 10 digit sequences belonging to { 0,1}* such that each sequence starts with 01 and/or ends with 01.
A. 512 B. 448
C. 478 D. None

66. int a[10]; here a is __ pointer
A. Wild B. Constant
C. Dangling D. None
Consider the following graph



67. The third row in the transitive closure of the above graph is
A. 1,1,1 B. 1,1,0
C. 1,0,0 D. 0,1,1

68. A graph is having adjacency matrix of all 1's then
A. It is fully connected
B. Its path matrix is same as adjacency matrix
C. Contains cycles
D. All

69. Does C doesn't support
A. Call by value B. Call by reference
C. Call by copy D. Call by restore
E. C & D

70. Make uses
A. I-nodes information
B. Data blocks information
C. Time stamps information
D. None

71. The grammar S → SS|aa|e
A. is LR(0) B. is SLR(1)
C. LALR(1) D. None

72. What is the average time required to read or write 512-bte sector for a disk with 5400 RPM with the average seek time of 12ms, transfer rate of 5MB/sec. Assume controller overhead is 2ms and disk is idle initially.
A. 10 ms B. 12 ms
C. 19.7 ms D. 19.2 ms
E. None

73. A synchronous bus with clock cycle of 50 ns and 40ns per handshake can do one transmission per 1 clock cycle. Then its bandwidth ____. The data portion of bus is 32 bits wide and memory access time is 200 ns.
A. 10 Mbps B. 11 Mbps
C. 13.3 MB/sec D. None

74. A memory and bus system supports block access of 4 32-bit words. A 64-bit synchronous bus has clocked at 200MHz, with each 64-bit transfer taking 1-clock cycle, and 1 clock cycle to send and address memory.

Two clock cycles are needed between each bus operation. A memory access time for the first four words of 200ns; each additional set of four words can be read in 20ns. Assume that a bus transfer of the most recently read data and a read of the next four words can be overlapped. The bandwidth for a read of 256 words for is ___

A. 4 MB/sec B. 71.11 MB/sec
C. 177.11 MB/sec D. None

75. Complexity of generating all possible subsets of a set of data

A. O(n) B. $O(n^2)$
C. $O(2^n)$ D. O(n!)

76. Complexity of splitting a set of data in half repeatedly and traversing each half

A. O(lg n) B. O(n lg n)
C. $O(n^2 \lg n)$ D. None

77. Which is having highest complexity $O(n^2)$, $O(n^2 \lg n)$, O(n!), $O(2^n)$?

A. $O(2^n)$ B. $O(n^2)$
C. $O(n^2 \lg n)$ D. O(n!)

78. $T_1(n)=3n \lg n + \lg n$, $T_2(n)= 2^n+n^3+25$, $T_3(n,k)=k+n$, k<=n. The highest ordered one is

A. T_1 B. T_2
C. T_3 D. All are same order

79. A graph's adjacency matrix is as follows. Then the graph

0100
0010
0001
1000

A. Directed
B. All nodes are a cycle
C. One connected component
D. All

80. In a graph adjacency matrix i'th row contains all 1's and i'th column contains all zero's (except i'th row element in the column) then i'th node is

A. Sink
B. Not reachable from any node
C. Not reachable from any node except itself
D. None

81. What is the average read/write time for a 512 byte sector with 4 ms average seek time, transfer rate 4MB/s and 7200 RPM, controller overhead is negligible and queing delay is also negligible

A. 4 ms B. 7 ms
C. 8.3 ms D. 8.15 ms

82. For the following code fragment, 2 stage pipeline is proposed; 1st stage for multiplication (10 ns) and second 2nd stage for addition (10 ns) is required. Then how much time it takes to complete.

for I=1 to 100 do A[I]=B[I]*C[I]+D[I]

A. 2000 ns B. 1010 ns
C. 1020 ns D. 2010 ns

83. A system has 3 page frames in main memory and uses LRU replacement policy with the following reference string. What is the state of main memory (the pages existing) after the 5th page fault?

1223413121

A. 321 B. 124
C. 234 D. None

84. Imagine that the length of a 10Base5 LAN cable length is 2500m and it is working at 1Gbps. In worst case how many padding bits are needed? (assume addresses are 6 bytes long)

A. 46 B. 614
C. 6474 D. None

85. Finding whether a graph G(V,E) contains a sink (a vertex with in-degree of 1 and out-degree as zero) is

A. O(V) B. $O(V^2)$
C. O(1) D. None

86. A diagraph has the following adjacency matrix

1 1 0 0
0 1 0 1
0 1 1 1
1 0 0 1

The corresponding relation is

(1) refl. (2) symm.
(3) antisymm.

A. 1 B. 2
C. 3 D. 1 & 2
E. 1 & 3

87. Given a machine with only a stack whose top can be outputted and on which POP and PUSH are allowed. Which of the following strings can be sorted in ascending order?

A. 4312 B. 3421
C. 2134 D. 1243
E. 3142

88. Number of productions used to generate a string of length x for a grammer in CNF is

A. x+1 B. x
C. 2x D. 2x–1
E. x^2

89. A 2,3 tree is a tree in which every node has either 1,2 or 3 sons, except the leaves, and also has equel path length from root to any leaf. The number of leaves is 9. The number of internal nodes could be

A. 7 B. 8
C. 9 D. 6
E. 3

90. Solution of

T(n)= 2T(n/2) + n n>1
= 1 n=1

A. nlogn + n B. n^2
C. nlogn –n +2 D. n+1
E. n+3

91. Z → Ax|By
A → Ay|y
B → x|y
What is the regular expression for the set of strings generated by the grammer?

A. xy B. yy
C. xy+y+y(y)*x D. None

92. The number of distinct strings of length 3 that can be obtained using a,a,b,b,c is

A. 7 B. 6
C. 12 D. 15
E. 18

93. In connection of n processors find the minimum number of connections needed to provide two distinct paths between any two processors.

A. 2^n B. n
C. n+1 D. n(n+1)/2
E. n–1

94. The number of stack locations needed for evaluating (((a+b)*e)–d) using a stack is

A. Four B. Five
C. Six D. Three
E. Seven

95. S → aAa | BSB
A → aS | c
B → b | bS
What is the minimum length string full of terminals possible ?

A. 1 B. 2
C. 3 D. 4
E. 5

96. Newton Raphson method is used to calculate the square root of 2 using the iteration x(i+1) = (x(i) + 2/x(i))/2 if the intial value x(0) is chosen as 1.5, then the number of iterations needed to get an accuracy of 10^(–14) are

A. 2 B. 4
C. 8 D. 16
E. 32

97. We have the recurrence relation
A(n) = 2A(n–1) – 1
What is the order of A(n)?

A. Linearly
B. Quadratic
C. Cubic
D. Exponential
E. Logarithmic

98. In a pipelined architecture it is found that 20% of the instructions are branch instructions. Out of these 20% are unconditional branches. Out of the conditional branches about half the branches are taken. If each instruction takes one cycle. Each branch instruction causes 1 cycle delay if it is not taken and a 3 cycle delay if it is taken. What is the average number of cycles each instruction takes?

A. 1.68 B. 1.2
C. 1.4 D. 1.44
E. 1.32

99. Hashing table has size of 1400 records. Assume that not more than 1000 records are present at any time. There are 4 keys in each record a1, a2, a3, a4– each can take 36 values.
What will be the best hashing function?

a. a1 + a2 b. a1*a2*a3
c. a1*a2 d. None

100. Consider the join of a relation R with a relation S. If R has m tuples and S has n tuples, then the maximum and minimum sizes of the join, respectively are

A. m+n and 0 B. mn and 0
C. m+n and |m–n| D. mn and m+n

ANSWER KEY

Test 1

1. C	**2.** C	**3.** C	**4.** D
5. B	**6.** C	**7.** D	**8.** C
9. A	**10.** D	**11.** C	**12.** D
13. A	**14.** D	**15.** C	**16.** A
17. B	**18.** D	**19.** C	**20.** D

21. C **22.** A **23.** C **24.** C
25. D **26.** E **27.** D **28.** E
29. C **30.** A **31.** A **32.** A
33. C **34.** C **35.** C **36.** A
37. C **38.** D **39.** B **40.** A
41. C **42.** B **43.** B **44.** B
45. D **46.** E **47.** C **48.** B
49. D **50.** D **51.** A,D **52.** C
53. C **54.** D **55.** A **56.** B
57. B **58.** D **59.** B **60.** D
61. A **62.** C **63.** E (Option is not Given)
64. C **65.** A **66.** B **67.** D
68. D **69.** E **70.** C
71. D (Erroneous Question)
72. C **73.** C **74.** B **75.** D
76. B **77.** D **78.** B **79.** D
80. C **81.** C **82.** B **83.** D
84. B **85.** B **86.** E **87.** A,C
88. B **89.** E **90.** A **91.** D
92. E **93.** D **94.** D **95.** C
96. B **97.** C **98.** E **99.** C
100. C (mn,0)

Test 1

EXPLANATIONS FOR SELECTED QUESTIONS

9. p is not a pointer

12. Except all PC is little endian type

15. Most of the recent m/c has L1 cache inside and L2 outside (recent Itanium processor things are different)

19. Usually (say router m/c) may be having two network cards which belongs to two separate LAN's. Thus they should differ in IP addresses.

22. Max Data Rate = 2Hlog2V = 2*8*10^6*log2(4)= 32 Mbps.

26. Probably situation will be root node, two children will accommodate all 9 leaves.

28. Given strings in aaaa, baaa, abab can not be generated through different derivations

29. For example one 1000111(7), second is 1000111(7) then sum in seven bits is 0001110 (14). Thus by flipping MSB we can get in excess-64 form.

33. Possible derivations: (bold ones are derivations)

(i) S→S+**S** S→S+**S**+S S→S+S*S+S

(ii) S→**S**+S S→S+**S**+S S→S+S*S+S

(iii) S→S+**S** S→S+S***S** S→S+S*S+S

(iv) S→**S**+S S→S+S***S** S→S+S*S+S

(v) S→S***S** S→S+S***S** S→S+S*S+S

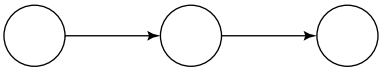
(vi) S→**S** *S S→S+S***S** S→S+S*S+S

36.

xyz	xyz′	xy′z	x′yz	xyz	F	xy	yz	zx	F
000	0	0	0	0	**0**	0	0	0	**0**
001	0	0	0	0	**0**	0	0	0	**0**
010	0	0	0	0	**0**	0	0	0	**0**
011	0	0	1	0	**1**	0	1	0	**1**
100	0	0	0	0	**0**	0	0	0	**0**
101	0	1	0	0	**1**	0	0	1	**1**
110	1	0	0	0	**1**	1	0	0	**1**
111	0	0	0	1	**1**	1	1	1	**1**

41. Check explanation from definitions

45.



51. aaab is valid string where as bbbbaa is not.

53.

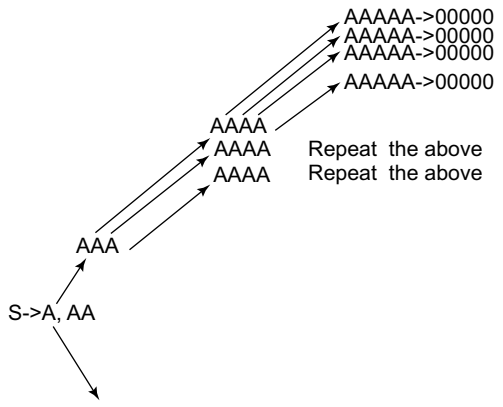
xyz	y′	x′z′	E	Term No
000	1	1	1 Maxterm	0
001	1	0	1 Maxterm	1
010	0	1	1 Maxterm	2
011	0	0	0 Minterm	3
100	1	0	1 Maxterm	4
101	1	0	1 Maxterm	5
110	0	0	0 Minterm	6
111	0	0	0 Minterm	7

56. See the theorem

57. See the theorem in Deo

61. Probability of loss = C, probability of success = 1–C
Probability of success in k'th attempt is $(1–C)C^{k–1}$
Expercted value = $\Sigma\, k\,(1– C)C^{k–1}$
$= (1–C)C^0 + 2(1–C)C^1 + 3(1–C)C^2 + 4(1–C)C^3$
$= (1–C)\{C^0 + 2C^1 + 3C^2 + C^{3\ldots}\}$
$= (1–C)/C\{C + 2C^2 + 3C^{3\ldots}\}$ (multiplying and devideing with C)
$= (1–C)/C^*\ C/(1–C)^2$
$= 1/(1–C)$

64.



Above tree will be exactly replicated

66. a is called as constant pointer. We can apply on them a++, a—etc.

72. See answer from question 80

73. 50 ns to send address to memory

200 ns read the memory

50 ns send the data to the device

Thus total 300 ns.

The maximum bus bandwidth = 4 bytes/300ns = 13.3 MB/sec

74. 1 clock cycle to send the address to memory

200ns/5ns=40 clock cycles to read

2 clock cycles to send the data from memory

2 idle clock cycles between this transfer and next

Thus total 45 cycles. There for 256/4 = 64 transactions are needed, so the entire transfer takes 45 × 64 = 2880 clock cycles. Thus the latency is 2880 cycles × 5ns = 14400ns.

The number of bus transactions per second is = 64*1second/14400 = 4.44M transactions/sec

The bus bandwidth is = (256*4) bytes*1 second/ 14400ns = 71.11MB/sec

80. 4+ 0.5/7200RPM+0.5KB/4MB/sec = 4+4.15+0.125 = 8.3ms

82. No of stages = k = 2

No of data elements = n = 100

Each stage requires = 10ns

Time required = (k+n–1)*10 = 1010

83.

Page No	Page Fault or Not	
1	PF	1
2	PF	1,2
2	No PF	1,2
3	PF	1,2,3
4	PF	2,3,4
1	PF	**3,4,1**

84. According to 802.3 specifications maximum length of cable is 2500 and the minimum allowed frame should takes 51.2 micro seconds. There fore minimum frame size is 6400 bytes = 1Gbps*51.2micro seconds. In frame addresses, and others occupies 26 bytes. Thus padding bytes 6400–26 = 6374.

92. Total sixteen

aab	aac	aba	abc	aca	acb	acb
bab	bac	bba	bbc	bca	bcb	
caa	cbb	cba	cab			

95.

$s \rightarrow aAa$ (by applying $A \rightarrow c$) $s \rightarrow aca$. Thus minimum 3 terminals

97. $A(n)=2A(n-1)-1 = 2\{2A(n-2)-1\}-1 = 2^2A(n-2) - 3 = 2^3A(n-3)-7 \ldots$

This may approach 2^n

This complex 2^n is about n^3. Thus it is third order complexity

98. 20% are branch (0.2)

80% are others (0.8)

% of conditional branches = 0.2*0.2 = 0.04 (they require 1+1 cycles)

% of not taken instructions =0.08 (they require 1+1 cycles)

% of taken instructions =0.08 (they require 1+3 cycles)

The avg no of cycles = 0.8*1 + 0.04*2 + 0.08*2 + 0.08*4 = 1.32

Test 2

Section A

1. When DMA is not preferred

A. I/O devices send interrupts but don't do any data movement

B. When data rate is so slow

C. To I/O copying take place in parallel with the CPU work

D. There is an overhead in terms of sending more complicated commands to the controller to inform it of the DMA operation.

E. All except C

2. Find odd one

A.  B. 

C.  D. 

E. 

3. Topological sort

A. Is not possible if the graph has cycles
B. Is used to sort numbers
C. Is applicable on DAG's
D. Is used in process management of OS
E. Ordering is not unique; any legal ordering will do

4. An array is having array containing indegrees of all nodes of a graph. There exists a function Find_New_Vertex_Of_Indegree_Zero() which when called it returns node index which is having indegree zero else it returns zero. If it returns zero then the graph is

A. DAG B. Connected
C. Contains cycle D. Acyclic

5. A graph with V vertices can have ___ minimum spanning trees.

A. V B. V–1
C. V^{V-2} D. V^{V}
E. None

6. Let A={a,b}, L(r) be the language where r = abb*a is

A. all words begins with a and ends with a
B. all words with exactly two a's
C. all words beginning and wnding in a and enclosing one or more b's
D. all words begining with ab and ending with a and having any no of b's
E. C & D

7. Find odd man out

A. L* contains all words over A*
B. L*, L+ are exactly same
C. L*, L + are same with the exception that L+ doesn't contain empty word
D. L* is called as Kleene Closure of language L

8. If A={a,b}, if r =____ is a regular expression then A*

A. a∨b* B. (a∨b)*
C. (a ∧ b)* D. None

9. In Databases ____ points are used with ROLLBACK statement to go back provided that the transaction has not been committed.

A. Synchronisation B. Marker
C. Savepoint D. None

10. Find odd man out

A. Readv system call
B. Writev system call
C. Writing into multiple buffers with single function call
D. Reading from multiple buffers with a single call
E. Read system call

11. Find odd one out

A. For a completely balanced binary search tree the worst case search complexity is $\log_2(N+1)$.
B. For an AVL tree the worst case search complexity is $1.44*\log_2(N+2)$.
C. AVL procedures guarantees atmost 5 pointer reassignments for a single reorganisation.
D. For a paged binary tree with k keys/page has worst case search complexity of $\log_{k+1}(N+1)$.
E. None

12. Ostrich Approach (Find odd one out)

A. Simplicity (for the OS designers),
B. Speed (no overhead),
C. Ease of use for the user – doesn't need to specify exact resources early.
D. Deadlocks can be prevented

13. Find odd one out

A. An access list is a list for each object consisting of the domains with a nonempty set of access rights for that object.
B. A capability list is a list of objects and the operations allowed on those objects for each domain.
C. Unix chmod command can extend per user basis permissions
D. In DR DOS 6.0 for each file one can assign passwd
E. None

14. What is the meaning of the "execute" permission bit for a directory in UNIX?

A. Allows execution of files in that directory
B. Allows access to a file in the directory if the file's name is known
C. Allows listing of directory contents
D. No meaning, as directories are not executable files

15. The performance of a particular hash function depends on

A. Distribution of key values that are currently in use

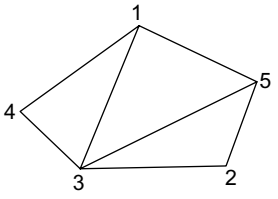
B. No. of key values that are actually in use relative to the size of the address space

C. No. of records that can be stored at a given address without causing any collision

D. The collision resolution technique used

E. None

16. No. of Hamiltonian paths in the following graph are —



A. 19 B. 10

C. 20 D. None

17. Find odd one out

A. A code is k-error-detecting iff its manimum distance is atleast k+1

B. A code is k-error-correcting iff its minimum distance is atleast 2k+1

C. Hamming codes are used for error detection and correction

D. Huffman codes are non uniform type

E. CRC codes can be used for error correction

18. Find odd man out

A. FORTRAN IV

B. FORTRAN 77

C. COBOL

D. C

19. Find odd one out

A. Language definition time binding

B. Language implementation time binding

C. Execution time binding

D. Compile time binding

20. Typeless language

A. C B. C++

C. Object oriented D. Assembly

E. C++

21. In the following function code if a,b are arguments passed in call by reference style, then find the wrong one.

```
swap(int a, int b)
{
x=x+y;
y=x-y;
x=x-y;
}
```

A. works for any two integers

B. works if we pass two different elements of an array, i.e swap(a[i], a[j]), i≠j

C. works if we pass two different elements of an array, i.e swap(a[i], a[j]), i≠j

D. works if we pass two different elements of an array, i.e swap(a[i], a[j]), i=j

E. None

22. Consider a buddy system in which a particular block under the current allocation has an address of 011011110000. If the block size is 4, what is the binary address of its buddy?

A. 011011110100 B. 011011110000

C. 011011111111 D. 111011110100

E. None

23. Free block list in memory is maintained as

A. Push-down stack B. FIFO queue

C. Both D. None

24. Number of frames sent at a time in ARQ are

A. 1 B. w

C. 3 D. None

25.
```
#define<stdio.h>
int a[10];
int size=100;
static int b=15;
void main()
{
int sum=0;
}
```

In the above program which is stack variable?

A. a B. bb

C. size D. sum

E. None

26. In LAN terminology, 10 Base 5 signifies

A. Ethernet Bus 10 metres long, digital, 5Mbps

B. Token Bus, 10 Mbps, digital, 500 metre

C. Ethernet Bus 10 Mbps, analog, 500 metre

D. Ethernet Bus 10 Mbps, digital, 500 metre

27. A device has two IP addresses. This device can be

A. a computer B. a router

C. a gateway D. All

28. IPV6 has ___ bit address.

A. 16 B. 32
C. 128 D. Variable

29. Let v,e,f are no. of vertices, edges and faces. v+f = e+2 is valid. Find odd one out.

A. Tetrahedron B. Cube
C. Prism D. Octahedron
E. Circle with n points joined radially

30. What is the min. hardware required for a parity bit generator when the bits come in serial order?

A. EXOR gates : 1 for each pair of bits
B. 4 EXOR gates
C. AND gates
D. TOGGLE flip-flop which acts a mod 2 counter
E. A and D with some minor additions

31. What is the time about C, the cost of replacement policy (which is given as the no. of page faults for a particular reference string?

A. C(LRU) always < C(FIFO)
B. C(LRU) always > C(FIFO)
C. C(LRU) always = C(FIFO)
D. Can't be answered
E. A and C

32. In the following Fortran code fragment how many assignments taken place?

```
do 10 i= 1,N
    do 20 j= 1,N
    20 A=B
    do 30 k= 1,N
    30 B=C
    10 continue
```

A. 2N B. N(3N)
C. 4N D. N*N
E. None

33. In a memory scheme ,the address of a location is specified by a page address and a displacement within a page , in hexadecimal. # of pages = 16. # of words per page = 256. The address of the 11th page, 94th word is

A. B5E B. A5D
C. 5EB D. E9C

34. F(x)= if odd(x) then (x–1)/2
else F(F(x–1)).
Then Find F(16).

A. 9 B. 3
C. 10 D. None

35. Solution of :

$T(n) = 2T(n/2) + n \quad n>1$
$= 1 \quad n=1$

A. nlogn + n B. n^2
C. nlogn –n +2 D. n+1
E. n+3

36. The number of distinct strings of length 3 that can be obtained using a,a,b,b,c is

A. 7 B. 6
C. 12 D. 15
E. 18

37. Which of the following does an FSA accept?

A. 0^n1^n n>=1
B. a^n b^m m<n
C. Balanced paranthesis
D. A string of 0's and 1's and in the prefix the difference between #1's and #0's is 1.
E. Equal no. of 0's and 1's

38. CARRY, in a half-adder can be obtained using

A. EX-OR gate B. AND gate
C. DRAM D. EX-NOR gate

39. No. of comparisions needed to find the max. and the second max. of 4 numbers is

A. 3 B. 4
C. 5 D. None

40. Which is a tautology?

A. a or b ⇒ a and b B. a ⇒ a and b
C. a and b ⇒ a or b
D. (a or b) ⇒ (a equivalence b)

41. The minimum numbers of hardware required to construct 3-to-8 decoder is

A. two 2-to-4 decoders
B. two 2-to-4 decoders and 1 1-to-2 decoders
C. three 2-to-4 decoders
D. depends on the technology TTL, CMOS, etc

42. Every node in a binary tree has the form lptr|info|rptr. In a binary tree with n nodes , how many nodes will point to null?

A. (n + 1) B. n
C. 2n D. None

43. Programs are written with subroutines because of the following reasons except

A. Easy debugging
B. No repitition of code
C. Increase access speed from buffer to main memory

D. To exploit locality of reference
E. None

44. An array defined as A[1:8,-2:2,3:12]. A[1,-2,3] is at location 2000 where will be A[5,0,5] be.
A. 2020 B. 2222
C. 2022 D. None

45. Assuming there are no branch instructions or data hazards, the number of cycles required to process 100 instructions through a 3-stage instruction pipeline is
A. 300 B. 103
C. 102 D. 97

46. Stack based architectures supports ___ address instructions.
A. One B. Two
C. Zero D. None

47. If a = 0xaa, b = a << 1 then b =
A. a B. 2a
C. 2b D. a–1

48. In which collision processing method it is not needed to detect a given list position, if it is occupied or not?
A. Quadratic B. Linked
C. Rehashing D. None

49. The simplest way to organise the information about available memory block in a system is
A. AVL tree
B. Doubly linked circular list
C. Stack
D. None

50. The linear probing for collision resolution can lead to
A. Clustering B. Radix sort
C. Efficient storage D. Overflow

Section B

51. Given the functional dependencies F={A→BC, E→C, D → AEF, ABF → BD} the extraneous left attribute
A. B in A→BC
B. A in D→AEF
C. B in ABF→BD
D. None

52. Find out the Huffman code for symbol a from the following data

Symbol	a	b	c	d	e	f	g
Probability	0.4	0.1	0.1	0.1	0.1	0.1	0.1

A. 1 B. 0
C. 10 D. 11
E. Both A & B

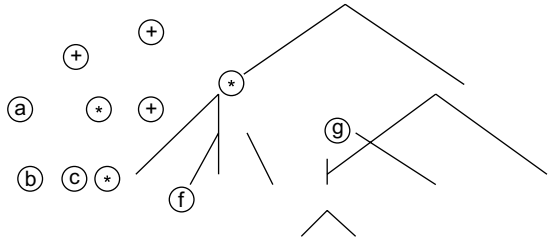
53. Solve the following traveling salesman problem

Station	A	B	C	D	E
A	∞	13	19	16	15
B	14	∞	18	15	16
C	14	18	∞	14	13
D	13	17	16	∞	18
E	19	17	16	17	∞

A. C→A→B→D→E B. C→B→A→E→D
C. B→E→A→C→D D. None

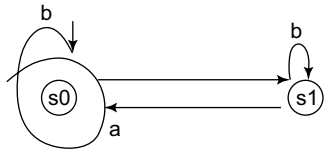
54. Disk requests come in to the disk driver for cylinders 12, 42, 106, 3, 55, 14 and 92, in that order. A seek takes 10 msec per cylinder moved. How much seek time is needed for SCAN algorithm. The arm is initially at cylinder 20, and the initial arm direction is ascending.
A. 1200 B. 1890
C. 1900 D. 3760
E. None

55. Find the odd thing about the following tree



A. (a+b*c)+((d*e+f)*g) B. abc*+de*f+g*+
C. abc*+def*+g* D. ++a*bc*+*defg

56. Find odd man out regarding the following transition diagram



A. aababbab B. aa
C. ababaa D. aaa
E. aa

57. Given the following data and for time slice value of 2 find out turn around time for process 3.

Process	Arrival	Service time	Priority
1	1	8	2
2	2	2	4
3	3	1	3
4	4	2	4
5	5	5	1

A. 9 B. 3
C. 10 D. None

58. A channel uses a 2400 hz carrier. The signal to noise ratio snr = 30db. The max bit rate __
A. 2400 B. 24000
C. 24Mbps D. None

59. A computer has the following page frames. The time of allocation, time of last access, and the accessed (*A*) and modified (*M*) bits for each page are as shown below (with times in clock ticks, lower is earlier in time). First please note that the times as given as milliseconds since some 'epoch'. In other words, time 200 is earlier than time 360. Then frames selected for replacement using LRU,NRU,FIFO and second chance approaches are:

Frame	Allocation	Last reference	A	M
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1
4	200	265	0	1
5	200	290	1	0
6	205	273	0	0

A. 1,0,2,0 B. 1,6,2,0
C. 1,2,6,0 D. A & B
E. None

60. Assume you have an inode-based filesystem. The filesystem has 512 byte blocks. Each i-node has 10 direct, 1 single indirect, 1 double indirect, and 1 triple indirect block pointer. Block pointers are 4 bytes each. Assume the inode and any block free list is always in memory. Blocks are not cached. What is the number of disk block reads and writes required to write 1 byte to a file 1.in the best case, 2.in the worst case?
A. 1w, 4r/1w B. 4w,1r/1w
C. 1w,1r/1w D. None

61. What are the mistakes in the following piece of code?

```
#define MAXARRAY 50008
int *cubed(void){
   int f[MAXARRAY];
   int i;
   while (i <= MAXARRAY){ f[i] = i * i * i; i ++;
   }
   return f;
}
```

A. i Uninitialised
B. Fence post (array index limit exceeded)
C. Overflow may occur
D. Returning pointer to automatic variable
E. All

62. The candidate keys for the relation r(X,Y,Z,W,P) if all FDs of the set F = {Y→Z, Z→Y, Z→W,Y→P} hold for all instances of r.
A. XY B. XZ
C. WP D. Both A & B
E. None

63. Given the relation r(A,B,C) and the set F={AB→C, B→D, D→B} of FDs then
A. AB, AD are the candidate keys
B. A, B, D are prime attributes
C. Both A & B
D. None

64. Find odd man regarding the following code fragment

```
#include<stdio.h>
int x,y,x;
void f()
 { int t,u;
   void ff()
    { int x,w;
          void fff()
           { int y,w,t;
                 }
          x=y+t+w+z;
          }
   }
void main()
 { int z,t;
   f();
   }
```

A. Only one activation record will be created
B. Program will be compiled successfully
C. 3 activation records will be created when f () is called
D. Program will be executed

65. What is the average read/write time for a 512 byte sector with 4 ms average seek time, transfer rate 4MB/s and 7200 RPM, controller overhead is negligible and queing delay is also negligible
A. 4 ms B. 7 ms
C. 8.3 ms D. 8.15 ms

66. For the following code fragment, 2 stage pipeline is proposed; 1st stage for multiplication (10 ns) and

second 2nd stage for addition (10 ns) is required. Then how much time it takes to complete?

for I=1 to 100 do A[I]=B[I]*C[I]+D[I]

A. 2000 ns B. 1010 ns

C. 1020 ns D. 2010 ns

67. A paging system is employing HW cache as TLB of 20ns access time (search time) and physical memory access time is 100 ns. It is observed that the hit ratio is 98%. The effective memory access time is

A. 120 ns B. 122 ns

C. 220 ns D. None

68. A Stop and Wait protocol has the following data: Frame size (L) = 1000 bits; Transmission Speed (R) = 1 Mps Distance (D) = 10 kms; Velocity of Propagation (V) = 2*10^8 m/sec. Calculate the utilization of the link, ignoring effects of ACK, and CPU.

A. 1 B. 0.9

C. 0.5 D. None

69. Procedure ZAP(n)

begin

if n $\Leftarrow$ 1 then zap = 1;

else zap = ZAP(n–1) + ZAP(n–3);

end

Find ZAP(6).

A. 6 B. 9

C. 10 D. None

70. What is the min. length string possible in the following grammar?

S $\rightarrow$ aAa | BDB

A $\rightarrow$ aS | c

B $\rightarrow$ b|bD

D $\rightarrow$ S

A. 4 B. 3

C. 7 D. None

71. Grammar

S $\rightarrow$ Aa | bB | Sa

A $\rightarrow$ b| a

B $\rightarrow$ Aa

Which of the following is true?

A. aaaa proves that grammar is ambiguous

B. baaa proves that the grammar is ambiguous

C. abab proves that the grammar is ambiguous

D. Ambiguous but not detectable from the above given productions

E. Not ambiguous

72. What is true about the language generated by the above rules?

A. Start(first) and end(last) symbols are the same for any string

B. Only odd length strings are generated

C. Consists of atmost one 'c' in any string

D. None

73. Consider the program fragments

```
Procedure;
var: x,y;
Procedure P(a,b,c)
Begin
b := b + 10;
a := b + c;
End;
y = 20; x = 10;
P(y,x,y);
Write(y,x)
End;
```

With respect to the above program fragment if the procedure P is call by reference what are the values of x,y printed out:

A. 40,20 B. 35,10

C. 30,20 D. 25,35

74. Five processes are in a queue. The time for completion of each are 6,3, 4,3 and 2, respectively. Find the minimum average turn around time

A. 18/5 B. 9

C. 62/5 D. 63/5

E. 18

75. On a Stack machine

Store pops one value and stores it;

MUL and ADD operate on the top two values of the stack

and the result is stored on to the stack

What is stored in T2 after the following sequence of statements

```
PUSH A
PUSH B
MUL
STORE T1
PUSH T1
PUSH T1
MUL
PUSH C
MUL
PUSH D
ADD
```

STORE T2
HALT

A. (AB)^2 * C * D
B. (AB)^2 * C + D
C. AB^2 * C + D
D. None

76.

A B C	f
0 0 0	1
0 0 1	1
0 1 0	1
0 1 1	X
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	X

Consider the function 'f' shown. Which of the following describes correctly the relation between the minimum sum and the minimum product form of 'f' ?

A. They are logically equivalent by definition.
B. They are logically equivalent because dont cares are used in the same way.
C. They are logically equivalent because don't cares don't matter.
D. They are logically not equivalent by definition.
E. They are logically not equivalent because don't cares are used in different ways.

77. Assume data transfer between disk and MM are interrupt driven, 1 byte at a time. if the instruction to accomplish a 1 byte transfer take 8 micro sec. and the interrupt overhead is 10 micro sec. then the time available in micro sec. for other computing between byte transfer is

A. 0
B. 8
C. 10
D. 14
E. 32

78. Consider N employee records to be solved in memory for online retrieval. Each employee received is uniquely identified by a social security number. Consider the following ways to store the N records

1. An array sorted by social security number
2. A link list sorted by social security number
3. A link not sorted.
4. Balanced binary search tree with social security number as key.

for the statements 1 to 4, respectively the average time for an efficient program to find out an employee received given social security number is

A. o(log n) o(n) o(n) o(log n)
B. o(n) o(log n) o(n) o(1)
C. o(log n) o(log n) o(n) o(1)
D. o(n) o(n) o(n) o(1)
E. o(n) o(log n) o(log n) o(1)

79. A stack can be defined abstractly by the following rules. Let S be a stack and let X be a symbol, then :

1. e,the empty stack is a stack
2. push(S,X) is a stack
3. pop(push(S,X)) = S
4. top(push(S,X)) = X

All the following are derivable from above except

A. top(push(push(e,X),Y)) = Y
B. pop(push(e,X)) = e
C. push(pop(push(e,X)),Y) is a stack
D. top(push)pop(push(e,X)),Y) = Y
E. pop(pop(push(e,X))) = X

80. There are 4 wagons standing on section A of the track. A move consists of moving any one wagon along the track without changing the direction from one straight section to the other. No vehicle can stop on any curved section. What is the minimum number of moves to invert the order of the 4 wagons.

A. 7
B. 8
C. 9
D. 10
E. 11

81.

State	Input Next	State
1	0	3
1	1	2
2	0	4
2	1	3
3	0	3
3	1	3
4	0	3
4	1	2

1 is start state and 4 is final state.

What is the regular expression accepted ?

A. 1(01)*0
B. 1110*1
C. (101)*0
D. None

82. Hashing table has size of 1400 records. Assume that not more than 1000 records are present at any time. There are 4 keys in each record a1, a2, a3, a4- each can take 36 values. what will be the best hashing function

A. a1 + a2
B. a1*a2*a3
C. a1*a2
D. All

83. If A={a,b,c}, B={1,2}, then the number of relations A to B is

A. 6 B. 9
C. 64 D. None

84. Find valid order

A. log n, n,n^2, nlogn
B. logn, n, nlogn, n^3
C. n^2, nlogn, 2^n, n^3
D. None

85. A pipeline has 4 stages with time delays 20 ns, 20 ns, 100 ns and 40 ns then in order to get maximum performance which stage can be devided?

A. Stage with 20 ns B. Stage with 100ns
C. Stage with 40 ns D. None

86. A pipeline has 4 stages with time delays 60 ns, 50 ns, 90 ns and 80 ns. The interface latch has a delay of 10 ns then clock frequency of the pipeline is

A. 10MHz B. 13MHz
C. 20MHz D. None

87. Consider a computer with 8M bytes of RAM and 128K cache with cache block size of 4K, and direct mapping is used. The no. of different memory block can map on to a given physical cache block are

A. 2048 B. 256
C. 64 D. None

88. A demand paging system with page table in registers takes 5 ms to service a page fault if an empty page is available else it takes 15 ms. Memory access time is 1 micro second. Assume we want an effective access time of 2 micro seconds and that the page to be replaced is dirty 60% of the time. What is the approximate maximum acceptable page fault rate to meet this access time requirement?

A. 0.1% B. 1.0%
C. 2.5% D. 0.01%

89. Parity bit stuffed frame for the data frame 01100111100 is 110011000111100

90. Checksum value for 01100111100 if CRC-12 is used is____

91. Given R(A,B,C,D) and FDs A→B, BC→D then find the wrong one

A. AC→D B. B→D
C. AD→B D. None

92. The Boolean expression A+BC

A. (A'+b)(A'+C) B. (A+B)(A'+C)
C. (A+B)(A+C) D. None

93. A shift reduce parser carries out the actions specified within braces immediately after reducing with the corresponding rule of grammar

S → xxW { print "1"}
S → y { print "2"}
W → Sz { print "3"}

What is the translation of xxxxyzz using the syntax directed translation scheme with the above rules

A. 23131 B. 11233
C. 11231 D. 33211

94. Rank of matrix

$$\begin{matrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 2 & 1 & -2 \end{matrix}$$

A. 0 B. 2
C. 1 D. None

95. A pipeline has k stages with time delays T_i, i=1,..,k and each stage has a latch with latch access time T_l then the clock period of this linear pipeline is given as

A. $\max\{T_1,T_2,\ldots.T_k\} + T_l$
B. $\min\{T_1,T_2,\ldots.T_k\} + T_l$
C. $\max\{T_1,T_2,\ldots.T_k\}$
D. None

96. Imagine that the length of a 10Base5 LAN cable length is 2500m and it is working at 1Gbps. In worst case how many padding bits are needed? (assume addresses are 6 bytes long)

A. 46 B. 614
C. 6474 D. None

97. What is the best method to sort if no of elements are less than 1

A. Bubble B. Quick
C. Merge D. None

98. R(A,B,C,D,E), FDs = {A→B, BC→D, D→BC, DE→ϕ}, What is the normal form of above relation?

A. 1NF B. 2NF
C. 3NF D. BNF
E. None

99. Eigen vector matrix is

A. determinant 1
B. rows are perpenticular to each other
C. columns are perpenticlar to each other
D. norm of rows are columns are one
E. All

100. Given 128×8 RAM chips, no of RAM chips needed for 2M RAM, no of address lines, chip select lines are:

A. 16, 11,4 B. 16,4,11
C. 11,4,16 D. None

ANSWER KEY

Test 2

1. E	**2.** E	**3.** B	**4.** C
5. C	**6.** E	**7.** B	**8.** B
9. C	**10.** E	**11.** E	**12.** D
13. C	**14.** B	**15.** E	**16.** C
17. E	**18.** C	**19.** C	**20.** D
21. D	**22.** A	**23.** C	**24.** A
25. D	**26.** D	**27.** D	**28.** C
29. E	**30.** E	**31.** E	**32.** E
33. A	**34.** B	**35.** A	**36.** E
37. D	**38.** B	**39.** B	**40.** C
41. B	**42.** A	**43.** C	**44.** B
45. C	**46.** C	**47.** B	**48.** D
49. B	**50.** A	**51.** C	**52.** E
53. A	**54.** B	**55.** C	**56.** D
57. B	**58.** B	**59.** D	**60.** A
61. E	**62.** D	**63.** C	**64.** A
65. C	**66.** B	**67.** B	**68.** B
69. B	**70.** B	**71.** E	**72.** C
73. A	**74.** B	**75.** B	**76.** C
77. B	**78.** A	**79.** E	**80.** C
81. A	**82.** C	**83.** C	**84.** B
85. B	**86.** A	**87.** C	**88.** A
89. 110011000111100		**90.** Refer text	
91. B	**92.** C	**93.** A	**94.** D
95. A	**96.** D	**97.** A	**98.** C
99. E	**100.** A		

Test 2

EXPLANATIONS FOR SELECTED QUESTIONS

35. T(n)= 2T(n/2) + n
=2{2T(n/4)+n/2) + n
= 4T(n/4) +n+n
=4{ 2T(n/8) + n/4} + n + n
$=8T(n/8)+\frac{n+n+n}{\text{logn terms}}$

In the last first term will be n. Therefore **nlogn+n**

45. See explanations for question 66. No of cycles required is (k + n – 1) = (100 + 3 – 1) = 102

58. **Step 1:** obtain absolute S/N ratio from the snr (decibels, db)

snr = 10 log10 (S/N)

30 = 10 log10 (S/N)

$\frac{30}{10}$ = log10 (S/N)

3 = log10 (S/N)

log10(S/N) = 3

S/N = 10^3 =1000

Step 2:

{Shannon's Law: C= W log2[1 + (S/N)] }

C = W log 2(1 + S/N)

= 2400* log2(1 + 1000)

= 2400 * log2 (1001)

= 2400 * 9.98 (2^9= 512; 2^10 = 1024;

log2(1001) in between 9 and 10 more towards 10)

= 23,99 bit/sec

56. String aaa is not acceptable whereas remaining are acceptable

65. 4+0.5/7200RPM+0.5KB/4MB/sec = 4+4.15+0.125 = 8.3 ms

66. No of stages = k = 2

No of data elements = n = 100

Each stage requires = 10ns

Time required = (k + n – 1)*10 = 1010

67. Time required to access memory if TLB hit occurs = 20 ns+100ns = 120ns

Time required to access memory if TLB miss occurs = 20 ns+100ns+100ns = 220ns

TLB hit = 98% = 0.98

Effective memory access time = 0.98*120 + 0.02*220 = 122ns

68. Tf = L/R = 1000 bits

---------- = 10^(–3) seconds

1*10^6 bits/sec

Tp = D/V = 10 kms * 10^3 m/km

---------- = 10^4

2*10^8 m/sec ------ seconds

2*10^8

= 0.5 * 10^(–4)

= 0.05 * 10^(–3) seconds

Tcycle = Tf + Tp + Tcpu + Tack + Tp + Tack, cpu

Tack, TCpu's are not given assumed negligible

= Tf + 2Tp

= 1*10^(–3) + 2*(0.05*10^(–3))

= 1*10^(–3) +0.1*10^(–3)

= 1.1*10^(–3) seconds

U = Tf/Tcycle = 1/1.1 = 0.9

[same result could have been obtained from U = 1/(1 + 2a)

a = Tp/Tf = 0.05*10^(–3)/1*10^(–3) = 0.05

1+2a = 1 + 2*0.05 = 1 + 0.1 = 1.1

U = 1/1.1 = 0.9

**caution, the formula was derived assuming Tack, Tcpu's negligible in this given situation, fits in, but not always true

70. See explanation for previous test

77. 18 ms

82. a1*a2 is best option as hash table size is 1400

85. Stage with 100 ns one has to be devided as it limits the pipeline throughput.

86. Here for every 100ns (90+10) one result will be coming. Therefore clock frequency 10MHz.

96. D (typing errors)

According 802.3 specifications maximum length of cable is 2500 and the minimum allowed frame should takes 51.2 micro seconds. There fore minimum frame size is 6400 bytes = 1Gbps*51.2micro seconds. In frame addresses, and others occupies 26 bytes. Thus padding bytes 6400–26 = 6374.

Test 3

Section A

1. All are attributes of static binding except
- A. Implicit type binding used in Fortran
- B. Read-only variables that are initialised with an expression
- C. Taking * as multiplication in C language
- D. Declaring a variable in C language

2. NOOP
- A. In RISC pipelines
- B. Is used to align instructions on word boundaries
- C. Is used to align subroutines on page boundaries
- D. Is used to introduce delays in the program
- E. Improves serial programs performance

3. How many instruction bits are required to specify two operand registers and one result register in a machine that has 16- general-purpose registers?
- A. If the "result" register can be different from the "source", 12 bits are needed.
- B. If the "result" register is always, say, the first destination register, then 8 bits are required.
- C. Information is insufficient
- D. A & B
- E. None

4. A Graphics application requires computations on vectors, and matrices. The best architecture is
- A. SISD -single instruction, single data path
- B. SIMD - single instruction, multiple data paths
- C. MISD - multiple instruction, single data path(s)
- D. MIMD - multiple instruction, multiple data paths(s)

5. An MCU can handle invalid op codes.
- A. Most MCU's use small table lookup to convert the op-code into a pointer into the microprogram memory.
- B. Op code itself indicates validity
- C. Ask the system administrator
- D. None

6. ROMs are used in computers for __ except
- A. Control memory for the CPU.
- B. Code to execute for exceptions.
- C. Routines to isolate faults in the computer hardware.
- D. Serial number of the computer.
- E. To store return addresses of system routines

7. Find odd one out about write through caching
- A. Increases memory bus traffic
- B. Increased likelihood of memory contention
- C. Increased likelihood of cache contention
- D. Cache consistency is guranteed

8. In C language a function can be passed as argument to a function. Really function name itself is pointer to the function. Thus this style can called as
- A. Call by value
- B. Call by name
- C. Call address
- D. Call by procedure
- E. Call by reference

9. The step which is taken when a context switching takes place is
- A. (H/W) Saves User SP and IP in reserved location
- B. (H/W) Loads Kernel SP and IP
- C. (S/W) Saves some working registers
- D. Allocates space for page table
- E. (S/W) Calls Scheduler

10. Long-term scheduler does not
- A. Selects processes from job queue
- B. Loads the selected processes into memory for execution
- C. Updates the ready queue

D. Controls the degree of multiprogramming (the number of processes in the main memory)
E. Select the process from ready queue

11. Priority of a process in a ready queue is not influenced by
A. Its memory(swapping) requirements
B. Attained service time
C. Real time spent in the system
D. Total time required for
E. Urgency, system load, external priorities

12. Where we can not save PSW value in
A. Registers B. Control stack
C. PCB D. TCB
E. Cache

13. Find odd one out of the following regarding DES
A. Weak keys is a major security issue
B. Semi weak keys comes in pairs
C. DES use 64 bit blocks
D. DES uses a key for encipher and another to decipher
E. DES uses a key of 56 bits

14. Complexity of Jarvis march to find convex hull of given set of points n and h as the number of points in the convex hull
A. O(n) B. O(n^2)
C. O(nh) D. None

15. Find odd one regarding inline functions
A. inline functions supports type checking.
B. inline functions if used excessively it may lead to reduced cache hit.
C. inline functions can be used to replace any function to get performance gain.
D. inline functions are very suitable for short functions such as accessors, mutators.

16. Name mangling
A. is also known as name decoration
B. can be also applicable to variables
C. is used by C++ compilers to generate unique names for identifiers in the program
D. All

17. Destination address of an IP packet which is to be broadcasted in remote LAN ___
A. All 1's in hostid B. All 1's in netid
C. All 0's in hostid D. All 0's in netid
E. All 32 bits to be 1's

18. A host with an IP address of 144.2.2.1 needs to test internal software. What is the destination address in the packet?
A. 127.1.1.1 B. 144.0.0.0
C. 144.255.255.255 D. 127.0.0.0
E. Both A & B

19. malloc, new etc. memory allocators use
A. Per-process memory
B. May call OS only if per-process memory is over or exhausted
C. Both A & B
D. None

20. Find odd man out
A. EXE B. COFF
C. ELF D. DLL

21. Which is not an advantage of large pages over small Page Sizes?
A. Better TLB coverage (fewer misses),
B. Larger I/O transfers (improves throughput),
C. Smaller page tables,
D. VPN has fewer bits
E. Copy-on-write costs more

22. The characteristics of rendezvous message passing are
A. Boundaries between messages, guaranteed message delivery
B. Either sender or receiver block until both are ready to exchange the message.
C. Rendezvous message passing is often use to synchronise two processes; the message forms a `token', and when either process has the token, it is permitted to access a shared resource.
D. None

23. Is it possible that two processes can read and write to location 15,000 but not be accessing the same RAM location? Where?
A. Single partition systems
B. Paged systems
C. Yes. In paged system every process may have their own page map and base address. Thus location 15000 may map to different RAM location.
D. None

24. Find the correct one
A. Every graph covering contains minimal covering
B. A covering of an n-vertex graph wil have at least $\lceil n/2 \rceil$ edges.
C. The minimal covering of an-vertex graph can contain no more than n–1 edges
D. All

25. Find the incorrect one regarding linkage editor
- A. Produces load module or an excutable image which is linked version of the program
- B. If a program which is to be executed many times without being reassembled, the linkage editor is best preferred
- C. External references and library searches are performed only once
- D. Linking loader is used with linkage editor while external references are resolved.

26. Find the correct answer regarding T(n)=T(n–1)+an if n>0 =b otherwise
- A. Order is linear
- B. Order is O(n^2)
- C. O(n^3) order
- D. It is running time of selection sort
- E. B & D

27. Running time of an algorithm is given as: T(n)=T(n/3)+T(2n/3)+n then the order of the algorithm.
- A. Linear
- B. Quadratic
- C. Cubic
- D. Exponential
- E. O(nlogn)

28. Find the correct regular expression for language of real numbers given +, –,., d, E as alphabet
- A. (+|–|e)dd*(.d*|e)(E(+|–|e)d*|e)
- B. (+|–)dd*(.d*)(E(+|–)d*)
- C. (+|–|e)dd*(d*|e)(E(+|–|e)d*|e)
- D. None

29. A connected undirected graph with n vertices has a min-cut of cardinality k
- A. then G has at least nk/2 edges
- B. then G will have n/2 edges
- C. then G will have nk edges
- D. None

30. In the expression a[I]= 4+2 no of identifiers are
- A. 8
- B. 4
- C. 2
- D. None

31. Commonly Intermediate code is
- A. Single address
- B. Two address
- C. Three address
- D. None

32. Commonly employed data structure for representing symbol table is
- A. Tree
- B. B-tree
- C. Hash table
- D. Linked list

33. Find odd one out
- A. Producing a good compiler from dirty compiler written in a assembly language is called as bootstrapping
- B. After bootstrapping we may have a compiler in both source and executing code
- C. Bootstrap loader causes the reading of other records
- D. Bootstrap loader is a special type of absolute loader
- E. None

34. Find odd one out
- A. lex
- B. flex
- C. yacc
- D. bison
- E. make

35. Find odd man out
- A. pipe()
- B. popen()
- C. msgget()
- D. fork()

36. What is the ratio of the number of characters needed for integer 'a' in bases 'b' and 'c' ?
- A. ceiling(log a (base b))/ceiling(log a (base c))
- B. (log a (base b))/(log a (base c))
- C. ceiling(log a (base b))/ceiling(log a (base c))
- D. None

37. [[set A XOR set B]-set C] = ?
- A. Only non_common elements.
- B. Common element
- C. Combined elements
- D. None

38. 84(base 16) +121(base 4) =?
- A. 2213(base 8)
- B. 2131(base 4)
- C. 2212(base 16)
- D. None

39. Give a regular expression containing exactly 2 consecutive 1s
- A. (0 + 10)*11(01 + 0)*
- B. {01}*11{01}*
- C. {11}*
- D. None

40. 5 processes are in a queue. The times for completion of each are 6, 3, 4, 3 and 2 respectively. Find the minimum average turn around time
- A. 18/5
- B. 9
- C. 62/5
- D. 63/5
- E. 18

41. When a fork() is executed the chiald process will inherit except
- A. Command line arguments
- B. Environment variables
- C. gid
- D. uid
- E. umask

42. Memory mapping of a file
- A. To share a part of a file by multiple process
- B. Lets a part of virtual address space to be associated with a section of a file

C. Writes by any of the process on mapped file's part/page can be seen by all processes
D. All

43. Interpolation search is a variant of
A. Bubble sort B. Binary search
C. Fibnocci search D. Linear search

44. Intel Pentium processor
A. 16K L1 cache
B. 8K Instruction cache (a part of L1 cache)
C. 8K data cache (a part of L1 cache)
D. All

45. Find odd one out
A. A good hash function reduces collisions
B. A good hash function uses entire key rather than just a part
C. A good hash function should distribute records uniformly
D. A good hash function is easily computable
E. None

46. Big:= A[1];
For i:= 2 to N do
if A[i] > Big then Big:= A[i];
If Big can be anywhere with equal probability in A[1:n] find the average number of executions of the statement Big := A[i]
A. (n/2) B. (n–1)
C. (n/4) D. None

47.
```
Begin
b:= a+b+c+d+e
if c>0 then d:= a+b+c+e;
else c:= a+b+d+e;
z:= a+b+c+d+e;
end
```
What can be taken out of the compound statement?
A. a+d+e B. a+e
C. a D. None

48. How more than one value can be returned from a function in C?
A. By sending a dummy array and storing the values to be returned
B. By creating a dynamic array and storing the values to be returned and returning its starting address
C. By passing addresses to scalar variables and storing in them the values to be returned
D. Through global variables
E. All

49. Find odd one out about Bit-fields of C language
A. Data members are specified in terms of bits
B. Bit-fields can be used as members of another structure
C. An array of bit field type of objects can not be declared
D. Really how much memory saved very much depends on the computer architecture
E. None

50. A pipeline has 4 stages with time delays 60 ns, 50 ns, 90 ns and 80 ns. The interface latch has a delay of 10 ns then cycle time of this pipeline is
A. 90 ns B. 100 ns
C. 60 ns D. None

Section B

51. A program runs in *10 seconds* on a computer (A), which has a 100*MHz* clock. A computer designer is trying to build a machine (B) that will run this program in *6 seconds*. It was determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing machine (B) to require *1.2* times as many clock cycles as machine (A) for this program. What clock rate should the designer target?
A. 150MHz B. 200 MHz
C. 220 MHz D. None

52. Consider a 32-bit microprocessor that has an on-chip 16-kbyte four-way set associative cache. Assume that the cache has a line size of four 32-bit words. Draw a block diagram of this cache, showing its organisation and how the different address fields are used to determine a cache hit/miss. Where in the cache is the word from memory location ABCDE8F8 mapped?
A. 140 B. 120
C. 143 D. None

53. A computer has a cache, main memory, and a disk for virtual memory. If a referenced word is in the cache, 20ns are required to access it. If it is in main memory but not in the cache, 60ns are needed to load it into the cache, and then the reference is started again. If the word is not in main memory, 12ms are required to fetch the word from disk, followed by 60ns to copy it to the cache, and then the reference is started again. The cache hit ratio is 0.9 and the main memory hit ratio is 0.6. What is the average time in nanoseconds required to access a referenced word on this system?
A. 48000 B. 1200008
C. 480026 D. None

54. Huffman coding is applied for the following data (72 symbols) then calculate no of bits the compressed data occupies

Symbol	A	E	P	R	U
Probability	12/72	18/72	7/72	15/72	20/72

A. 576 bits B. 300 bits
C. 163 bits D. None

55. The data bits are 11010110111110, generator polynomial is 10111 then checksum is

A. 1001 B. 1110
C. 0000 D. None

56. From the following data find out when process 3 is completed if shortest job next is employed.

Process	Arrival Time	Expected CPU Time
1	0	14
2	3	12
3	5	7
4	7	4
5	19	7

A. 7 B. 21
C. 25 D. None

57. Interrupt is used for a byte transfer request from the above disk. If the interrupt overhead is 10 microsecs, 4 byte transfer time is 8 microsecs, how much time is available during byte transfers for other work?

A. 13ms B. 14micro seconds
C. 33ms D. None

58. The set of base & limit registers for all of the processes is shown in the following diagram. Assume RAM has 200ns access time. How much time it requires for compaction?

A. 10minutes B. 11 minutes
C. 11.81 minutes D. None

59. Consider a demand-paging system with a paging disk that has an average access and transfer time of 5 milliseconds for a single page. Addresses are translated through a page table in main memory, with an access time of 100 nanoseconds per memory access. Thus, each memory reference through the page table takes two accesses. The system has a 48-entry TLB (with access time 10 ns) to speed up memory accesses. Assume that 99% of memory accesses result in a TLB hit, and of the remaining 1%, 5 percent (or 0.05% of the total) cause page faults. What is the effective memory access time?

A. 10 ns B. 11.8ns
C. 2500 ns(app) D. 40

60. The following are the preorder and inorder traversals of a tree

Preorder: GDBFEJMLP

Inorder: BDEFGJLMP

A. Balanced tree
B. Unbalanced tree
C. J is unbalanced node
D. If H is inserted tree becomes balanced
E. B, C, D are true

61. Complexity of Interpolation search assuming data is evenly distributed is

A. logn B. O(loglogn)
C. O(nlogn) D. None

62. Out of the following functions to calculate 2^5 which is efficient? Assume argument x is 5.

```
int F1(int x) { return (2<<x); }
int F2(int x) { int I, s=1; for(I=1;I<=x;I++)s*=2; return s; }
int F3(int x) { if(x==0) return 1; else return 2*F3(n–1) ; }
```

A. F1() B. F2()
C. F3() D. None

63. Find the square root of 2 after third iteration using the following recursive procedure

$x_1=1$

$x_{n+1}= 0.5*(x_n+2/x_n)$

A. 1.5 B. 1.414
C. 1.4142 D. 1.4167
E. None

64. Let a double precision number be represented as concatanating two single precision numbers. For example (a, b) and (c, d) are such a two numbers, then the product can be calculated through (a+b)(c+d) = ac+ad+bc+bd (appropriate shifting is done). Minimum how many times this double precision operation costlier compared to single precison

A. 2 B. 4
C. 3 D. None

65. What is the minimum frame size of 250m Ethernet Lan with 51.2 micro second propagation delay which is working at 1Gbps?

A. 64bytes B. 6300bytes
C. 640 bytes D. None

66. In unix system, fsck command is used to correct file system inconsistency.

A. In the first phase it identifies missing data blocks
B. In the second phase it identifies link mistmatches

C. It has total 5 passes out of which first two are major ones
D. All

67. What is the largest single file supported by a I-node based file system which has block size of 1K and block addresses 2 bytes?

A. 17GB B. 128GB
C. 64MB D. None

68. Find odd one out

A. NULL pointer of C is analogous to nil pointer in Pascal and LISP
B. NULL pointer doesn't point to any object or function
C. When malloc() fails it returns NULL indicating not allocated
D. Null pointer constant is used for representing null pointers in source code
E. None

69. Find the incorrect one

A. Prime area is an area in which the records are written when an indexed sequential file is created.
B. Prime area is a sequential file as it follows strict key sequence
C. When an indexed file is created there will not be any records placed in overflow area
D. Prime area is the super block of disk partition

70. In heap file organisation the average no. of blocks that may have to be read for a data item if it has equal probability of appearance anywhere in the file is (n is total no. of blocks, N is total no. of records)

A. n/2 B. N/2
C. N/n D. None

71. If no of records are 10^6, record size (including linkage pointer0 is 100 bytes, key is 10 bytes, block is 2000 bytes, pointer size 10 bytes then

A. 50000 first level index entries occuping 500 blocks
B. 500 second level indexes
C. Third level indexes are also needed
D. All
E. None

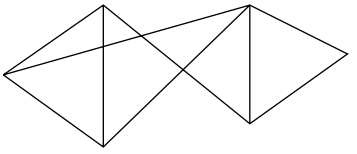
72. Consider the relation BOOK(CALL-NO, TITLE, AUTHOR-NAME, PUBLISHER, YEAR, PRICE) and the functional dependencies

AUTHOR-NAME TITLE → CALL-NO

CALL-NO → AUTHOR-NAME TITLE PUBLISHER YEAR.

A. AUTHOR-NAME TITLE PUBLISHER can be super key
B. CALL-NO is a key
C. AUTHOR-NAME TITLE is a key
D. All are valid

73. Find the wrong spanning trees for the following figure



A. Minimal spanning tree BE, CE, AE, DF, BD
B. Minimal spanning tree can be BD, AE, DF, CE, AF
C. Minimal spanning tree can be AC, AE, AF, BF, BD, BE
D. None

74. Packet broadcasting

A. Is employed by LAN's
B. When all stations share common channel this is used
C. No switching devices are available
D. Each station checks whether the packet whether it is intended for it or not
E. All

75. Erlangs is unit for

A. for data rate
B. throughput
C. traffic intensity ratio
D. no of processes completed by a multicomputer
E. None

76. At a server the mean arrival rate of client request is 0.05 requests/sec and mean service time is 0.1 requests/sec then mean service time is equal to

A. 1 sec B. 10 ms
C. 10sec D. None

77. Consider a terminal concentrator with 4800 bps input lines and one 9600 bps output line. The mean packet size is 1000 bits. Each of four lines delivers poisson traffic on average of 2 packets/sec.

A. The mean delay experienced by a packet from the moment the last bit arrives at the concentrator until the moment that bit is retransmitted to the output line is 625 msec
B. The mean no. of packets in the concentrator including the one in service is 5

C. Both A & B
D. None

78. Consider a slotted ALOHA system with four stations with the following offered loads 0.1, 0.5, 0.2 and 0.2. Then the station 1's throughput is
A. 0.1 B. 0.5
C. 0.2 D. 1
E. 0.032

79. Consider the following production rules
$S \rightarrow aS$ (1)
$S \rightarrow aB$ (2)
$B \rightarrow bC$ (3)
$C \rightarrow aC$ (4)
$C \rightarrow a$ (5)
A. The above production rules represents the language a^nba^m
B. aabaaa is valid according the above production rules
C. Production rules 1,2,3,4,4,5 are used to produce aabaaa
D. All

80. Consider the following production rules
$S \rightarrow aSBC$ (1)
$S \rightarrow aBC$ (2)
$CB \rightarrow BC$ (3)
$aB \rightarrow ab$ (4)
$bB \rightarrow bb$ (5)
$bC \rightarrow bc$ (6)
$cC \rightarrow cc$ (7)
A. Represents $a^nb^nc^n$
B. Represents $a^nb^mc^n$
C. Represents a^nbc^n
D. Represents $a^nb^nc^m$

81. Does the following function works? If not what is the problem?

```
Char* ITOA(int n)
  { char buf[20];
        spintf(buf,"%d", n);
        return buf;
  }
```

A. Yes.
B. No. Returning automatic array is not acceptable
C. Compiler dependent behaviour
D. None

82. The Boolean expression (X+Y)(X+Y')(X'+Z)
A. equal to XZ B. $\Sigma(4,5,7)$
C. $\Pi(0,1,2,3,6)$ D. All

83. Find the Boolean expression which has max terms as $\Sigma(0,4,5,7,8,9,13,15)$ and the variables are {w,x,y,z} then the minimal expression is
A. x'y'z' + w'xy'+wy'z+xz
B. w'y'z' + wx'y' + xz
C. x'y'z' + w'xy'+wy'z+xz + y'z'w'
D. None

84. Which of the following is not correct?
A. External fragmentation can occur in a paged virtual memory system.
B. External fragmentation can be prevented (almost completely) by frequent use of compaction, but the cost would be too high for most systems.
C. A page frame is a portion of main memory.
D. Once a virtual memory page is locked into main memory, it cannot be written to the disk is a wrong statement.

85. Find the correct one
A. Pages that are shared between two or more processes can never be swapped out to the disk.
B. The allocated portions of memory using a buddy system are all the same size.
C. Demand paging requires the programmer to take specific action to force the operating system to load a particular virtual memory page.
D. Prepaging is one possibility for the fetch policy in a virtual memory system.

86. Find the correct one
A. The translation lookaside buffer is a software data structure that supports the virtual memory address translation operation.
B. In a symmetric multiprocessor, threads can always be run on any processor.
C. Hrashing will never be a problem if the system has 1 GB of real memory.
D. The resident set of a process can be changed in response to actions by other processes

87. This function is proposed for use in an operating system, with the definitions of
Process, Process_Set and other functions given elsewhere.

```
Process next_process(Process_Set available_processes) {
  Process_Set A = highest_valuation(available_processes); /* priority ranking */
  Process_Set B = earliest(A); /* actual arrival time */
  Process c = random_selection(B); /* tie-breaker */
```

```
    return c; /* run this process next */
}
```

Does this function could lead to proc. starvation among the available processes.

A. Yes B. No
C. Yes. Because of ranking
D. None

88. Find the in correct one

A. Control and status registers are usually not visible to user programs.
B. An interrupt will always lead to an operating system action that corrects a problem and allows the interrupted process to continue running.
C. An interrupt handler will usually disable further interrupts temporarily.
D. Management of a process runtime stack usually requires a stack pointer, a stack base, and a stack limit.

89. Which of the following functions can be reentrant type? When?

```
int prob5(int A) { static int B = 0; B = B + A;
return B; }
int prob5_r(int A, int *B) { *B += A; return *B;}
```

A. prob5() B. prob5_r()
C. Not applicable
D. prob5_r() if each thread has their own B (properly initialised)

90. Given the relation R(ABCDE) with F = {A → BCDE, B → ACDE, C → ABDE}. Which of the following is the loss less BCNF decomposition?

A. (ABCD,AE) B. (ABC,DE)
C. (AB,CDE) D. None

91. The set of functional dependices F={A → BC, B → C, A → B, AB → C} then canonical cover for F is

A. A → B, A → C B. A → B, B → C
C. A → B, A → C D. None

92. Which of the following is not correct

A. Heap allocated records that are not reachable by any chain of pointers from program variable are garbage.
B. Garbage collection will be done runtime system
C. Mark-and-sweep is one way of garbage collection
D. Reference counts is another way of dealing garbage collection
E. C language has in built garbage collector like Java

93. Equivalent one for copying collection

A. Memory compaction
B. Disk defragmentation
C. Buddy system D. Fragmentation
E. Both A & B

94. What should be the order of loops to exploit the advantage of cache?

```
for I=0 to N-1
for J=0 to M-1
for K=0 to P-1
A[I,J,K=(B[I,J-1,K] + B[I,J,K] + B[I,J+1,K] )/3
```

A. I,K,J B. I,J,K
C. K,I,J D. None

95. Let F={A → B, AB → C, D-.AC, D-.E} and G={A → BC, D-.AB}

A. F covers G and G covers F
B. F covers G and G not covers F
C. G covers F and F not covers G
D. None

96. ___ is used to catch dangling references

A. make and sweep B. reference counts
C. tombstones D. None

97. Find odd one out of the following when passing arguments to functions.

A. When 1-D array is passed formal parameter can be declared as int a[] or int *a
B. When 2-D array with contiguous layout is passed formal parameter can be declared as int a[][n] or int (*a) [n]
C. When 2-D array is passed with row-pointer layout as formal parameter can be declared as int *a[] or int **a
D. None

98. Dope vector

A. Vector with same elements
B. Vector with mirror elements
C. Compiler maintains details of array variables
D. Is used to check array bounds
E. Both C & D

99. Register windows are used in

A. RISC machines B. Normal processors
C. Used to exploit pipelining benefits
D. Both A & C

100. Which of the following is not correct ?

A. Premature free or dangling pointer problems are same
B. Some programs continually allocate memory without ever giving it up and eventually run out of memory. This is known as memory leak.

C. A poor allocator may be the reason for external fragmentation in memory management.

D. None

ANSWER KEY

Test 3

1. B	**2.** E	**3.** D	**4.** B
5. A	**6.** E	**7.** D	**8.** D
9. D	**10.** E	**11.** D	**12.** E
13. D	**14.** C	**15.** C	**16.** D
17. A	**18.** E	**19.** C	**20.** D
21. E	**22.** D	**23.** C	**24.** D
25. D	**26.** E	**27.** E	**28.** A
29. A	**30.** B	**31.** C	**32.** C
33. E	**34.** E	**35.** D	**36.** A
37. A	**38.** B	**39.** A	**40.** B
41. D	**42.** D	**43.** B	**44.** D
45. E	**46.** A	**47.** B	**48.** E
49. E	**50.** B	**51.** B	**52.** C
53. C	**54.** C	**55.** C	**56.** C
57. B	**58.** C	**59.** C	**60.** E
61. B	**62.** A	**63.** C	**64.** B
65. D	**66.** D	**67.** B	**68.** E
69. D	**70.** A	**71.** D	**72.** D
73. C	**74.** E	**75.** C	**76.** C
77. C	**78.** E	**79.** D	**80.** A
81. B	**82.** D	**83.** B	**84.** A
85. A	**86.** D	**87.** C	**88.** B
89. D	**90.** D	**91.** B	**92.** E
93. E	**94.** A	**95.** B	**96.** C
97. D	**98.** E	**99.** D	**100.** D

Test 3

EXPLANATIONS FOR SELECTED QUESTIONS

51. A program runs in *10* seconds on a computer (A), which has a 100*MHz* clock. A computer designer is trying to build a machine (B) that will run this program in *6* seconds. It was determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing machine (B) to require *1.2* times as many clock cycles as machine (A) for this program. What clock rate should the designer target?

CPU TimeA= CPU clock cycles/clock rate

CPU clock cycles = Instrcutions for a programs X Avergae clock cycles per instruction (CPI)

CPU time for B can be used to find the same formula.

CPU timeb = 6 seconds = (1.2* CPU clock cycles of A) /clock rate A

Clock rate B= 2000 × 10^6 cycles/seconds = 200MHz

Therefor machine (B) must have the twice the clock rate of A o run the program in 6 seconds.

51. For example, location ABCDE8F8 is mapped onto:

Set 143, any line, byte 8.

Tag	set		Offset
A B C D E	**8**	**F**	8
	(1000)	**(1111)**	

52.

Location	Probability(hit)	Access time
In cache	0.9	20ns
In main memory	0.1*0.6	60+20 = 80ns
Neither in memory	0.1*0.4	12ms+60ns+20ns
or cache		=12000080

So, the average access time would be:

(0.9)(20) + (.06)(80) + (0.04)(12000080)

= 480026 ns.

58. Total memory occupied = 192K

= 192* 1024 word

During compaction one read/write operations will be done.

Thus no. of RAM operations = 192*1024*2

Total time for compaction = $192*1024*2*200*10^{-9}$ sec

59. If hit occurs time required is 10ns

If miss it may be checked in page table, thus access time = 10 + 100 + 100ns

If page fault occurs time required = 5ms+10ms+100

Effective access time = 0.99*10ns + 210*0.0095 + 0.0005*(5000110) = 2511.95ns

62. F1() As bitwise operators take less time

65. See Tenenbaum 640 bytes

67. N= datablock size/block addresses = 1KB/2byte = 512

There fore largest single file size using second level indirction is

10+ (1+512) + (1+512+512^2) +(1 + 512 + 512^2 + 512^3) blocks

= 512^3 blocks (approximately)= 512^3 * 1KB

= 128 GB.

76. Mean delay is needed. Not service time.
Mean delay= 1/(mean service time – mean arrivals)
= 1/(0.1-0.05)= 20 sec

77. Mean service time T= $1/(\mu C_i - \lambda_i)$
Where $1/\mu$ is mean packet size, λ_I is average packets per second, C_i is data rate.
$\mu C_i = 9600/1000 = 9.6 \qquad \lambda = 4 \times 2 = 8$
$T = 1/(9.6 - 8) = 625$ msec
Mean No of Packets $N = \rho/(1-\rho)$, ρ is traffic intensity
$= \lambda/\mu = 8/9.6$
$N = 5$

78. Thoughput of station 1 is = 0.1*(1–0.5)*(1–0.2)*(1–0.2) = 0.032

94. Spatial locality is exploited if I,K,J is employed. When loaded into cache a group of instructions are loaded such that cache hitting increases.

Test 4

Section A

Each question carries 1 mark for correct answer and –1/4 for incorrect answer.

1. The complexity of the best possible algorithm to determine if there exists an integer i such that $a_i = i$ in an array of integers $a_1 < a_2 < a_3 < \ldots, < a_n$.
A. O(1) B. O(n)
C. O(log n) D. O(n log n)

2. Number of multiplications needed to calculate x^{62} is
A. 62 B. 31
C. 5 D. 8

3. The time complexity of the following code fragment

```
Int I=n;
While(I>=1)
{
   X=X+1;
   I/=2;
}
```

A. O(1) B. O(n)
C. O(log n) D. O(n)

4. Find incorrect one regarding linkage editor.
A. Produces load module or an executable image
B. Loading can be done with one pass
C. Searches libraries and resolves external references every time the program is executed
D. No external symbol table is needed during loading

5. Find the in-correct one regarding relocation register.
A. It is directly available to user program
B. It is under the control of OS
C. It is saved during context switching
D. It is different from programmer-defined base registers of PowerPC

6. Running time of calculating n'th Fibnocci number can be represented as
T(n)= T(n–1) + T(n–2) +2
It's order of complexity is
A. Linear B. Quadratic
C. Cubic D. Exponential

7. A path matrix has all elements as 1's except diagonal elements whose values are 0's. Then, find the false one
A. Connected & complete
B. Its path matrix will be having all 1's
C. Its adjacency matrix square may have its diagonal elements as V-1, where V is no. of vertexes.
D. All vertexes are articulation points
E. None

8. Given a machine with only a stack whose top can be outputted and on which POP and PUSH are allowed, which of the following strings can be sorted in ascending order?
A. 4312 B. 3421
C. 2134 D. 1243
E. 3142

9. A modulo-20 counter can be designed using
A. 20 flipflops B. 4 flipflops
C. 5 flipflops D. None

10. The Hamming distance between two n-bit binary numbers is defined as the number of bit positions they differ. If we have a function called onescount(x); which returns the number of 1's in the number x, then a correct way to computer the Hamming distance between two numbers x and y is
A. Onescount(x) + onescount(y) -n
B. Onescount(x&y), where & is bitwise AND operator
C. Onescount(x|y), where & is bitwise OR operator
D. Onescount(x^y), where ^ is bitwise XOR operator

11. In 8-bit Booth's multiplication algorithm, the largest number of additions that will be ever required is
A. 8 B. 4
C. 3 D. 2
E. None

12. The width of the program counter of a CPU, which can address 200MB of main memory, is at least

A. 16 B. 25
C. 32 D. 28
E. None

13. Find the incorrect one out of the following.

A. Insertion sort performs on average $n^2/4$ comparisions
B. Insertion sorting algorithms time complexity is $O(n^2)$
C. Insertion sorting algorithms expected ruuning time is linear if the array elements are almost sorted
D. For large data insertion sorting is very preferable
E. None

14. If the array $A(b_1:e_1, b_2:e_2, \ldots\ldots, b_n:e_n)$ is stored in row major order, in the language, array name A itself pointer to the array, then $A(i_1,i_2,\ldots i_n)$ can be located at:

A. $A+ i_1*(e_1- b_1) + i_2 *(e_2- b_2) + \ldots\ldots + i_n*(e_n- b_n)$
B. $A+ (i_1-b_1)* (e_1-b_1) + (i_2-b_2)*(e_2-b_2) + \ldots\ldots + (i_n - b_n*)(e_n - b_n)$
C. $A + \Sigma_{i=1}^{n} (i_j - b_j)a_j$ where $aj = \Pi_{k=j+1}^{n} (e_k - b_k + 1)$, $1 \Leftarrow j < n$ and $a_n=1$
D. $A + \Sigma_{i=1}^{n} (i_j-b_j-1)a_j$ where $aj = \Pi_{k=j+1}^{n} (e_k- b_k + 1)$, $1 \Leftarrow j < n$ and $a_n= 1$
E. None

15. Let p: A byte has 7 bits

q: A word has 2 bytes

r: A bit is a 0 or 1

If, p is false and q, r are true then find out the valid statements

A. $p\wedge q$ B. $\sim p\vee\sim q$
C. $[(p\wedge q)\vee r]\wedge[\sim(p\wedge q)]$ D. $[(p\wedge q) \vee r] \wedge [(p \wedge q)]$
E. None

16. Find the odd one

A. $(p\wedge q)\vee(\sim p\vee\sim q)$
B. $\sim(p\wedge q)\vee(\sim p\vee\sim q)$
C. $[(p\vee r) \wedge (q\vee r)] \wedge [\sim p\vee \sim r]$
D. $[(p\wedge q)\vee r] \wedge [\sim(p\wedge r)]$
E. None

17. Find the unrelated one

A. Returning a reference to a local object
B. Dangling reference
C. A parameter passed by const reference be returned by const reference
D. Min(const int & a, const int &y) {return x<y ? x: y;}
E. Dangling memory

18. Locking the data before the beginning of transaction execution in order to prevent deadlock may lead to

A. Program abortion B. Starvation
C. Delay D. poor performance
E. Deadlock

19. Find the incorrect one

A. If a function argument is a base type object then while calling we can send its derived class type object without any problem.
B. If a function argument is a reference to a base type object then while calling we can send its derived class type object without any problem.
C. If a function argument is a pointer to a base type object then while calling we can send address of its derived class type object address without any problem.
D. If a function argument is pointer to a derived type object then while calling we can send address of its base class type object without any problem.
E. None

20. The primary key indexing technique does not allow

A. Duplicate data in a field
B. Multiple attributes
C. Sets of relations
D. Many-to-many relation

21. The downlink and uplink channels of a satellite are separated in

A. Time B. Space
C. Frequency
D. Not seperated in any domain
E. None

22. Sliding window protocols are very much (find the incorrect one)

A. Used in satellite channels where propagation delay is higher
B. Used in HDLC
C. USB
D. Used in channels for which signalling propagation is very high and data rate is also high such as fiber
E. Used in MAC protocols to share channel

23. A 30-channel PCM signal in digital telephony will have data rate of __

A. 64 Kbps B. 240 Kbps
C. 1 Mbps D. 2.048 Mbps
E. None

24. In a I-node based file system which uses 1KB blocks, 2 byte block addresses, with third level indirection can support theoretically a maximum file size of

A. 512MB B. 1GB
C. 128GB D. 17GB
E. None

25. Comment about the graph whose adjacency matrix is given as

0111
1010
0000
0010

A. Strongly connected B. Connected
C. bi-connected D. bi-patrite
E. None

26. Find the incorrect one

A. A language is regular if there is a finite automaton which accepts it
B. Union of two regular languages is regular
C. Concatenation of regular languages is a regular language
D. L={a, ab, ba, b} defined over {a, b} is regular.
E. None

27. Suppose X, Y are sets. It is given that there are exactly 97 functions from X to Y. Then,

A. |X|=1, |Y|=97 B. |X|=97, |Y|=1
C. |X|=97, |Y|=97 D. None

28. Suppose 2 process share variables and the final result depends upon the order of execution the processes. Which of the following describe the situation best?

A. Mutual exclusion B. Busy waiting
C. Race condition D. Deadlock

29. Belody Anomaly is applicable for

A. FIFO B. LRU
C. NRU D. Clock
E. Next fit

30. Heap allocation is required for the languages

A. That support recursion
B. That support dynamic data structures
C. That use dynamic scope rules
D. None

Section B

Each question carries 2 marks for correct answer and -1/2 for incorrect answer

31. Find the time complexity of the code fragment

```
integer iSum,i,j,k
For i=1 to n
 For j=1 to n
iSum = iSum + 1
Next j
For k=1 to 2*n
iSum = iSum + 1
For m=1 to 4*n step 2
iSum = iSum + 1
Next m
Next k
Next I
```

A. O(n) B. $O(n^3)$
C. $O(n^2)$ D. $O(n^2 \log n)$

32. In a system, the frame pointer %fp contains the stack pointer. When a function is called the following things happen. The %fp is set to the updated value. The following information is stored at the fp.

-4 return address

-8 static link

-12 any passed parameters followed by local variables.

In pass by reference the address of the passed variable is stored.

There is a instruction on the system

load %r2, offset(%r1).

This loads into r2, the value at the address pointed to by r1+offset. Consider the following fragment of code:

```
Procedure P
a:integer
Procedure Q (var x: integer) (call by reference)
x:= x+a (0)
end
begin
QA.
end
```

In this code how is the value of x read in line (0)?

A. load %t0, –12(%fp)
B. load %t0, –8(%fp)
C. load %t0, –12(%fp)
   load %t0, –8(%t0)
D. load %t0, –12(%fp)
   load %t0, 0(%t0)
E. load %t0, –8(%fp)
   load %t0, –8(%t0)

33. The time complexity of the following recurrence relations is:

$T(n) = 2T(n-1) + 1 \; n>1$

$T(0) = 1$

A. $O(n)$ B. $O(n^2)$

C. $\Theta(2^n)$ D. None

34. Adjacency matrix of a graph is given below. Comment about the graph

01000

00110

10000

00001

00100

A. Strongly connected B. Not connected

C. Has one sink D. Has two sources

E. None

35. The lower bound on the time complexity to found maximum and minimum of an array is

A. 2n –3 B. 3n/2 –3

C. 3n/2 –2 D. None

36. Which of the following exhibit locality of reference

I. Sequential processing of arrays

II. Symbol table using hashing

III. Collection of garbage in a linked memory

A. None B. I only

C. I and II D. I, II and III

37. Suppose there is a following scheme where, from the input string, remove the first 2 bits from left, xor them and then place it to the left and input and right of output. Continue this till you have only 1 bit in the input.

For example,

input output

1110 $

010 0

10 01

1 011

If the output is 101, then what is the input

A. 0011 B. 01110

C. 0111 D. 1011

E. None

38. Two processes have serial execution if instructions are executed in some order and instructions of a particular process are in order. Two processes share variable r1 and r2 and have local Variable's x,y in both the processes whose initial values are 0. r1 = r2 = 0 process 1 process 2 --------- -------- X = 1 Y = 1 r1 = Y r2 = X. Which of the following is not possible after execution of above process?

A. r1 = 0, r2 = 0 B. r1 = 1, r2 = 0

C. r1 = 1, r2 = 1 D. r1 = 0, r2 = 1

E. All of the above are possible

39. Consider the following piece of code where N is a power of 2 and all the logarithms are to the base 2

```
for (i = 1; i <= log(N); i++)
for (j = 1; j < N/(2^i); j++)
A[j] = A[2*j - 1] + A[2*j];
```

The above code has a running time

A. O(N) B. O(N*log(N))

C. O(N*N) D. O(log(N))

E. O(N/log(N))

40. Assume that there are N/2 processors, so that each iteration of the inner loop is run in parallel on these processors.i.e. the ith processor performs the ith iteration of the inner loop. Now the running time is

A. O(N) B. O(N*log(N))

C. O(N*N) D. O(log(N))

E. O(N/log(N))

41. In a pipelined architecture it is found that 20% of the instructions are branch instructions. Out of these 20% are unconditional branches. Out of the conditional branches about half the branches are taken. If each instruction takes one cycle. Each branch instruction causes 1 cycle delay if it is not taken and a 3 cycle delay if it is taken. What is the avg. no. of cycles each instruction takes.?

A. 1.68 B. 1.2

C. 1.4 D. 1.44

E. 1.32

42. What is the execution time of this sequence on a 7-stage pipeline with a 2-cycle instruction latency for non-branch instructions, but a 5-cycle branch instruction latency? Assume the branch is not taken, so the DIV is the next instruction executed after it.

BNE r4, #0, r5

DIV r2, r1, r7

ADD r8, r9, r10

SUB r5, r2, r9

MUL r10, r5, r8

A. 13 cycles B. 35 cycles

C. 16 cycles D. 15 cycles

E. None

43. A uniprocessor system uses separate instruction and data caches with hit ratios h_i and h_d, respectively. The access time from the processor to either cache is c clock cycles, and the block transfer time between the caches and main memory is b clock cycles.

Among all memory references made by the CPU, f_i is the percentage of references to instructions. Among blocks replaced in the data cache, f_{dir} is the percentage of dirty blocks. (Dirty means that the cache copy is different from the memory copy).

Assuming a write-back policy, determine the effective memory-access time in terms of h_i, h_d, c, b, f_i and f_{dir} for this system.

A. $f_i(h_i\, c + (1-h_i)(b+c) + (1-f_i)(h_d)c + (1-h_d)(\ (b+c)(1-f_{dir}) + (2b+c)f_{dir})\ )$

B. $f_i(h_i\ c + (1-h_i)(b+c)) + (1-f_i)\ ((h_d)c + (1-h_d)((b+c)(1-f_{dir}) + (2b+c)f_{dir})\ )$

C. $f_i(\ h_i\, c + (1-h_i)(b+c) + (1-f_i)(h_d)c + (1-h_d)(\ (b+c)(1-f_{dir}) + (2b+c)f_{dir})\ )$

D. None

44. Consider the concurrent execution of two programs by two processors with a shared memory. Assume that A,B,C, and D are initialized to 0 and that a print statement prints both the arguments indivisibly at the same cycle. The output forms a 4-tuple as either ADBC or BCAD.

Process P0		Process P1
A. A = 1	D.	C = 1
B. B = 1	E.	D = 1
C. Print A, D	F.	Print B, C

Then the number of all execution interleaving orders of six statements which will be preserve individual program order are

A. 10 B. 20

C. 40 D. 5

E. None

45. From the above question assume orders are preserved and all memory accesses are atomic; i.e a store by one processor is immediately seen by all the remaining processors. Then the possible 4-tuple output combinations is

A. 0111 B. 0000

C. 1111 D. None

46. For the same question 44, assume program orders are preserved but memory accesses are nonatomic; i.e a store by one processor may be buffered such that some other processors may not immediately observe the update. Then the not possible 4-tuple output combination is

A. 0111 B. 0000

C. 1001 D. 1011

E. None

47. User 1 is using seven printers and will need at most a total of 10 printers. User 2 is using one printer and will need at most four printers. User 3 is using two printers and will need at most 4 printers. Each user is currently requesting one more printer. Which of the following is true? With the system 12 printers are available in total.

A. The OS will grant a printer to User 1

B. The OS will grant a printer to User 2

C. The OS will grant a printer to User 3

D. The OS will not grant any more till some are relinquished

E. Bankers algorithm will prevent such a situation to arise.

48. User 1 is using x printers and will need a total of y printers. User 2 is using m printers and will need a total of n printers. This state is safe iff ($y \Leftarrow 12$, $n \Leftarrow 12$, $x \Leftarrow y$, $m \Leftarrow n$).

A. $x+n \Leftarrow 12$ and $y+m \Leftarrow 12$ and $x+m \Leftarrow 12$

B. $x+n \Leftarrow 12$, and $y+m < 12$ and $x+m \Leftarrow 12$

C. $x+n \Leftarrow 12$, or $y+m \Leftarrow 12$ and $x+m <\backslash 12$

D. $x+m \Leftarrow 12$

E. None

49. Given the symbols and frequencies of occurrences shown below, how many bits are necessary for the message "ABDEE" if Huffman coding is used to compress the data?

Symbol	Frequency
A	30
B	40
C	70
D	90
E	110
F	10

A. 11 B. 13

C. 15 D. 12

E. 14

50. A computer communication system uses a unique 8-bit pattern 01111110 to mark the beginning and end of frame. Suppose that this pattern appears in the information part of the frame and is subjected to bit studding; to what would the sequence be changed?

A. 001111110 B. 011111100

C. 011111010 D. 010111110

E. None

51. A a 2400 Hz carrier on the PSTN is reported to be having an SNR value of 30db. The maximum possible data rate is

A. 4800 bps B. 19200 bps
C. 23.99kbps D. 64kbps
E. None

52. ARQ protocol uses the following: Frame size=1000 bits, Transmission speed = 1Mbps, Distance =10Km, Velocity of propagation =2*10^8 m/sec, probability of error = 02. Then utilisation is

A. 0.9 B. 0.8
C. 0.72 D. None

53. Repeat question 52 for selective repeat problem.

A. 0.9 B. 0.8
C. 0.72 D. 0.51
E. None

54. A 6-MHz channel is used by a digital signalling system utilising four-level signals. What is the maximum possible transmission rate?

A. 6 Mbps B. 12 Mbps
C. 24 Mbps D. 3 Mbps
E. None

55. An eleven-bit word is constructed according to Hamming error correcting code scheme. Parity bits are: 1^{st}, 2^{nd}, 4^{th}, 8^{th} (denoted as p1,p2,p3,p4) and where as data bits are: 3^{rd}, 5^{th}, 6^{th}, 7^{th}, 9^{th}, 10^{th}, and 11^{th} (denoted as d1, d2, d3, d4, d5, d6, d7).

P1 is calculated to be even parity over d1,d2,d4,d5,d7

P2 is calculated to be even parity over d1, d3, d4, d6, d7

P3 is calculated to be even parity over d2, d3, d4

P4 is calculated to be even parity over d5, d6, d7

The code word 01010100110 is received. Where, if any where, has a bit been reversed during transmission?

A. p1 B. p2
C. d1 D. d5
E. None

56. A node in a network forwards incoming packets by placing them on its shortest output queue. What routing algorithm is in operation?

A. Hot potato B. Flooding
C. Static D. Delta
E. Hierarchical

57. $S \rightarrow AaaB$, $A \rightarrow BB|b$, $B \rightarrow AbA|a$

Which of the following cannot be generated by the above CFG?

A. aaaaa B. baaa
C. baabbb D. baabbbbb
E. baab

58. A polynomial has the following values in the range 0 to 5.

0, 3, 8, 15, 24, 35

What is the lowest degree possible for a polynomial that takes these values?

A. 3 B. 4
C. 5 D. 2
E. None

59. A 4^{th} degree polynomial is having the following values: 0,0,1,0,0. Third element after the last zero will be

A. 10 B. 45
C. 126 D. 8
E. None

60. Which of the following is LR(1) grammar?

I. $A \rightarrow A+A$, $A \rightarrow a$
II. $A \rightarrow aAa|bAb|a|b$
III. $A \rightarrow aAa|aAb|c$

A. I only B. II only
C. III only D. II and III
E. I and III

61. Rank of the following matrix is

$$\begin{matrix} 2 & 1 & 4 \\ 3 & -1 & 3 \\ 8 & -1 & 10 \\ 1 & 3 & 5 \end{matrix}$$

A. 4 B. 3
C. 2 D. Null
E. 1

62. A car registration system employs registration numbers to have either 1 or 2 or 3 letters, followed by a number with same number of digits and does not start with zero. The number of possible registrations numbers are

A. 36^3 B. 36!
C. 2^{36} D. 61080
E. 15879474

63. In IEEE 754 float representation, sign, exponent bits and fractional part bits are used. The number –0.0009002685546875 is stored as:

A. 1 0000 0000 1101 1000 0000 0000 0000 000
B. 1 0111 0100 1101 1000 0000 0000 0000 000
C. 0 0111 0100 1101 1000 0000 0000 0000 000
D. 1 0000 0000 1100 1010 1000 0000 0000 000
E. None

64. Which of the following are contradictory?

A. $p \vee (\sim p \rightarrow (p \vee \sim q))$
B. $(p \rightarrow q) \rightarrow ((p \rightarrow (q \rightarrow r)) \rightarrow (p \rightarrow r)$

C. $(p\rightarrow(q\rightarrow(r\rightarrow s)))\rightarrow(((p\rightarrow q)\rightarrow r)\rightarrow s)$

D. $\sim(p\vee q\vee\sim r)\wedge((r\rightarrow p)\vee(r\rightarrow q))$

E. None

65. The best way time complexity of Towers of Hanoi problem can be represented using recursive manner using divide conquer policy is

A. $T(n)=2T(n-1)+1$ given $T(1)=1$

B. $T(n)=2T(n-1)+1$ given $T(0)=1$

C. $T(n)=2T(n/2)+1$ given $T(1)=1$

D. $T(n)=2T(n/2)+n$ given $T(1)=1$

E. None

66. To sort (c, p, n, d, a, g) number of comparisons needed are

A. 9 B. 10

C. 11 D. 100

E. None

67. For the following code fragment, 2 stage pipeline is proposed; 1st stage for multiplication (10 ns) and second 2nd stage for addition (10 ns) is required. Then how much time it takes to complete.

for I=1 to 100 do A[I]=B[I]*C[I]+D[I]

A. 2000 ns B. 1010 ns

C. 1020 ns D. 2010 ns

68. Repeat the above problem assuming latching in pipeline require 2 ns and fetching time is ignored.

A. 2000 ns B. 1012 ns

C. 1212 ns D. None

69. A pipeline has 4 stages with time delays 60 ns, 50 ns, 90 ns and 80 ns. The interface latch has a delay of 10 ns then this pipeline clock frequency is

A. 10MHz B. 13MHz

C. 20MHz D. None

70. A CPU has a clock period of 20ns. It is possible to remove some (2%) instructions to make clock period as 18ns. This 2% instructions can be realized with 3 left over instructions in assembly. What will be the clock period of the second CPU?

A. 21 B. 18.72

C. 15 D. 20

E. None

71. A CPU has a clock period of 25ns. Some instructions can be removed from its instruction set to form a second CPU with a clock period of 24ns. These instructions comprise 1% of typical code and must be replaced by four instructions each.What percentage of typical code would be removed instruction have to comprise in order to for the two CPU's to have same performance?

A. 1 B. 1.388

C. 1.5 D. 1.1

E. None

72. Calculate execution time of the following set of instructions assuming a 5-stage instruction pipeline (all operands are registers).

ADD r3,r4,r5

SUB r7,r3,r9

MUL r8,r9,r10

ASH r4,r8,r12

A. 10 B. 15

C. 12 D. 20

E. None

73. An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R1 contains the number 200. Evaluate effective address if the addressing mode is register with R1 as index register.

A. 400 B. 702

C. 600 D. 602

E. None

74. If h_1, h_2, h_3 are cache hit ratio's in a 3 level cache schema and T_{L1}, T_{L2}, T_{L3} are their access times then the average memory access time is given as assuming T_{main} is the main memory access time and hit rates are till that level.

A. $h_1*T_{L1}+(h_2-h_1)*(T_{L1}+T_{L2})+(h_3-h_2-h_1)*(T_{L1}+T_{L2}+T_{L3})+(1-h_1-h_2-h_3)*(T_{main}+T_{L1}+T_{L2}+T_{L3})$

B. $h_1*T_{L1}+(1-h_2-h_1)*(T_{L1}+T_{L2})+(1-h_3-h_2-h_1)*(T_{L1}+T_{L2}+T_{L3})+(1-h_1-h_2-h_3)*(T_{main}+T_{L1}+T_{L2}+T_{L3})$

C. $(1-h_1)*T_{L1}+(1-h_2-h_1)*(T_{L1}+T_{L2})+(1-h_3-h_2-h_1)*(T_{L1}+T_{L2}+T_{L3})+(1-h_1-h_2-h_3)*(T_{main}+T_{L1}+T_{L2}+T_{L3})$

D. $h_1*T_{L1}+(h3-h_1)*(T_{L1}+T_{L2})+(1-h_3-h_2-h_1)*(T_{L1}+T_{L2}+T_{L3})+(1-h_1-h_2-h_3)*(T_{main}+T_{L1}+T_{L2}+T_{L3})$

E. None

75. If we want an average memory access time of 6.5ns, our cache access time is 5ns, and our main memory access time is 80ns, what cache hit rate must we achieve?

A. 97% B. 90%

C. 98.12% D. 99.12%

E. None

76. A certain memory has four levels with hit ratios 0.8, 0.95, 0.99 and 1.0, respectively. A program makes

3000 references to this memory system. Calculate the exact number of references that are satisfied at third level cache.

A. 2400 B. 540
C. 29.7 D. 30
E. None

77. What is the order of the following complexity equations?

T1=3n lg n + lg n

T2(n,k)=k+n, where k<=n

$T3=2^n+n^3+25$

A. T1<=T2<=T3 B. T2 <= T1 <= T3
C. T3<=T1<=T2 D. T2<=T3<=T1
E. None

78. Find the time complexity of the code fragment

```
integer iSum,i,j,k
For i=1 to n
 For j=1 to n
iSum = iSum + 1
Next j
For k=1 to 2*n
iSum = iSum + 1
iSum = iSum + 1
iSum = iSum +1
Next k
Next i
```

A. O(n) B. $O(n^3)$
C. $O(n^2)$ D. $O(n^2 \log n)$

79. Consider a system with 500 MHz clock. The HDD which transfers 4-word (of 4 bytes each) chunks at 4MB/sec. No transfer can be missed. Interrupt driven I/O is used. The overhead for each transfer including the interrupt is 500 clock cycles. The fraction of the processor time consumed if the HDD is only transferring 5% of the time.

A. 5% B. 2.5%
C. 2% D. 1.25%
E. None

80. Suppose we have processor with 500MHz clock rate and HD as above. Assume that the initial setup of a DMA transfer takes 1000 clock cycles for the processor and assume the handling of the interrupt at DMA completion takes 500 clock cycles for the processor. The HD transfer rate is 4MB/sec and uses DMA. If the average transfer from the disk is 8KB what fraction of the 500MHz processor is consumed if the disk is actively transferring 100% of the time? Ignore any impact from bus contention between the processor and DMA.

A. 1% B. 1.25%
C. 0.2% D. 0.5%
E. None

81. In a C program, n times fork() system call is made. Then, total number of processes which does not have child processes are

A. 2^n B. n
C. 2^{n-1} D. $2^{n/2}$
E. None

82. How are user-level threads treated differently from kernel-level threads?

A. Scheduling by process vs. by OS
B. Scheduled on one processor vs. any processor
C. Kernel-level requires mode switch to run the scheduler
D. In kernel-level threads when one blocks, they all block
E. All are true

82.

Process ID	Arrival time	Execution time
1	0	10
2	1	2
3	2	3
4	3	1
5	4	5

Calculate when process 4 is completed if we employ round robin algorithm with time slice value as 2.

A. 21 B. 4
C. 12 D. 9
E. 19

83. The RC 4000 system (and other systems) have defined a tree of processes (called a process tree) such that all the descendants of a process are given resources (objects) and access rights by their ancestors only. Thus, a descendant can never have the ability to do anything that its ancestors cannot do. The root of the tree is the operating system, which has the ability to do anything. Assume the set of access rights was represented by an access matrix, A. A(x,y) defines the access rights of process x to object y. If x is a descendant of z, what is the relationship between A(x,y) and A(z,y) for an arbitrary object y?

A. A(x,y) is superset of A(z,y)
B. A(x,y) is a subset of A(z,y).
C. A(x,y) is same as A(z,y)
D. None

84. What are the mistakes in the following C code fragment?

```
#define MAXARRAY 5000
int *cubed(void){
   int f[MAXARRAY]; int i;
 while (i <= MAXARRAY){
          f[i] = i * i * i;
          i ++;}
 return f; }
```

A. returning pointer to an automatic variable
B. overflow
C. use of unitialised variable
D. trying to access memory other than allocated
E. All

85. Consider a buddy system in which a particular block under the current allocation has an address of 011011110000. If the block size is 4, what is the binary address of its buddy?

A. 011011110100 B. 011011110001
C. 100011011110 D. None

86. There are 4 processes in the system and 4 resources types R1, R2, R3, R4 .There are 2 instances of R1 are available and remaining all are only 1 instance . currently R1 is allocated to P1 , another R1 is allocated to P2 , R3 to P3 , R4 to P4 ,R2 to P3.

The following requests are made

R1 ← P3
R1 ← P4
R3 ← P1

A. System is not deadlocked
B. System is deadlocked
C. System is not deadlocked because of multiple instances of R1
D. None

87. The following grammar is

S → x Y
X → x Y
Y → y X | y

A. type 0 B. type 1
C. regular D. type 2
E. None

88. The language recognised by the following grammar

S → a S | a B
B → b C
C → a C | a

A. L = { a^m b a^n , m,n >=1 }
B. L = { a^n b a^n , n >=1 }
C. L = { a^m b a^{2n} , m,n >=1 }
D. None

89. A → a B
B → b B | a C | D
C → a C | D
D → b

Which is acceptable by the above grammar ?

A. baaa B. aaaaab
C. aaa D. abbba
E. ababbbb

90. S → a A
A → a A B | a
B → b

Which is acceptable by the above grammar ?

A. aab B. abb
C. aaaabb D. ab
E. None

ANSWER KEY

Test 4

1. C	**2.** D	**3.** C	**4.** C
5. A	**6.** D	**7.** D	**8. B**
9. C	**10.** D	**11.** B	**12.** D
13. D	**14.** C	**15.** B,C	**16.** A
17. E	**18.** B	**19.** D	**20.** A
21. C	**22.** E	**23.** D	**24.** C
25. B	**26.** E	**27.** A	**28.** C
29. A	**30.** B	**31.** B	
32. Incomplete question			**33.** C
34. A	**35.** C	**36.** C	**37.** C, D
38. A	**39.** A	**40.** D	
41. Incomplete question			**42.** C
43. B	**44.** B	**45.** A,**C**	**46.** B
47. C	**48.** C	**49.** B	**50.** C
51. C	**52.** C	**53.** D	**54.** C
55. C	**56.** A	**57.** E	**58.** D
59. C	**60.** C	**61.** C	**62.** E
63. B	**64.** D	**65.** A	**66.** B
67. B	**68.** C	**69.** A	**70.** B
71. B	**72.** C	**73.** C	**74.** A
75. C	**76.** C	**77.** B	**78.** C
79. D	**80.** C	**81.** C	**82.** D
83. B	**84.** C	**85.** A	**86.** C
87. C	**88.** C	**89.** B	**90.** C