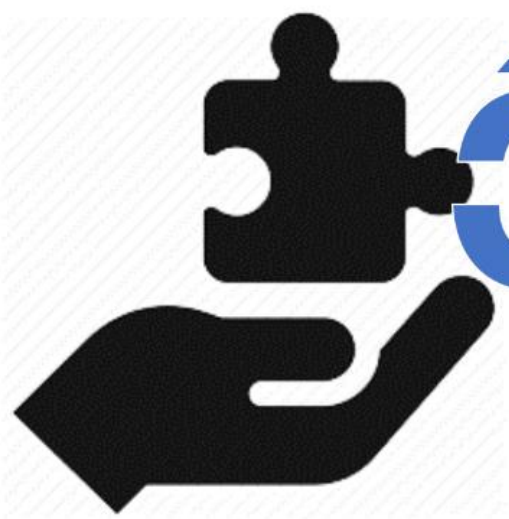
Soundarya Educational Trust

Soundarya Institute of Management & Science

Affiliated to Bangalore University || NAAC Accredited Institution || Approved by Govt. of Karnataka

Department of Computer Science



Ó uick reference

SOLUTION BANK

BACHELOUR OF COMPUTER APPLICATIONS (B.C.A)

VI Semester BCA

INDEX

SIXTH SEMESTER BCA

BCA601T : THEORY OF COMPUTATION

Total Teaching Hours : 60 No of Hours / Week : 04

Unit - I

Introduction to Finite Automata: The central concepts of Automata theory; Deterministic finite automata; Nondeterministic finite automata. An application of finite automata,

Finite automata with Epsilon transitions.

[12 Hours]

Unit - II

Regular Expressions: Finite Automata and Regular Expressions Applications of Regular Expressions. Regular languages; Proving languages not to be regular languages; Closure properties of regular languages; Decision properties of regular languages; Equivalence and minimization of automata.

[12 Hours]

Unit - III

Context–free grammars: Parse trees; Applications; Ambiguity in grammars and Languages. Definition of the Pushdown automata; the languages of a PDA; Equivalence of PDA's and CFG's.

[12 Hours]

Unit - IV

Deterministic Pushdown Automata:Normal forms for CFGs; The pumping lemma for CFGs; Closure properties of CFLs. Problems that Computers cannot solve.

[12 Hours]

Unit – V

The Turing machine:Programming techniques for Turing Machines. Undecidability, A Language that is not recursively enumerable; An Undecidable problem that is RE; Post's Correspondence problem.

[12 Hours]

Text Book:

1. John E. Hopcroft, Rajeev Motwani, Jeffrey D.Ullman: Introduction to Automata Theory, Languages and Computation, 3rd Edition, Pearson Education, 2011.

Reference Books:

1. John C Martin: Introduction to Languages and Automata Theory, 3rd Edition, Tata McGraw-Hill, 2007.
2. Daniel I.A. Cohen: Introduction to Computer Theory, 2nd Edition, John Wiley & Sons, 2009.
3. Thomas A. Sudkamp: An Introduction to the Theory of Computer Science, Languages and Machines, 3rd Edition,Pearson Education, 2006

BCA602T: SYSTEM PROGRAMMING

Total Teaching Hours : 60 No of Hours / Week : 04

Unit - I

Background: Machine Structure, Evolution of the Components of a Programming System, Assembler, Loaders, Macros, Compliers, Formal Systems. Machine Structure, Machine Language and assembly language: General Machine Structure, Machine Language, Assembly Language

[12 Hours]

Unit - II

Assemblers: General Design Procedure, Design of assembler, Statement of Problem, Data structure, Format of databases, algorithm, look for modularity, Table Processing: Searching and Sorting. The Problem, Searching a table, linear Search, binary Search, Sorting, interchange sort, Shell Sort, Bucket Sort, Radix Exchange Sort, address calculation sort, comparison of sorts, hash or random entry searching.

[12 Hours]

Unit - III

MACRO LANGUAGE AND THE MACRO PROCESSOR: Macroinstruction, Features of macro Facility, Macro instruction arguments, conditional macro Expansion, macro calls within macros, macro Instructions defining macros, Implementation, Statement of problem, implementation of a restricted facility, A two pass algorithm. A single pass algorithm, implementation of macro calls within macros. Implementation within an assembles.

[12 Hours]

Unit - IV

LOADERS: Loader schemes, Compile & go, General loading Scheme, absolute loaders, Subroutine Languages, Relocating loaders, Direct linking loaders, other loading Schemes – Binders, linking loaders, Overlays, Dynamic binders. Design of absolute loader, Design of a Direct linking loader Specification of problem, Specification of data structure, format of data bases algorithm.

[12 Hours]

Unit - V

COMPILERS: Statement of problem, Problem1: Recognizing basic Elements, Problem2: Recognizing Syntactic cutis & interpreting meaning, Problem3: Storage Allocation, Problem4: Code Generation. Optimization (machine independent) optimization (machine dependent), Assembly Phase, General Model of complier. PHASES OF COMPILERS: Simple Structure of Compiler, Brief introduction to 7 Phases of Compliers.

[12 Hours]

Text Books:

1. John J. Donowon, System Programming, TATA McGraw-Hill.

Reference Books:

1. Dhamdhere: System programming and Operating System TMH
2. Beck: System Software, 3/e Pearson Education.

BCA603T : CRYPTOGRAPHY AND NETWORK SECURITY

Total Teaching Hours : 60 No of Hours / Week : 04

Unit - I

Introduction: Security Goals, Cryptographic Attacks, Services and Mechanism, Techniques. Mathematics of Cryptography: Integer Arithmetic, Modular Arithmetic, Matrices, Linear Congruence.

[12 Hours]

Unit – II

Traditional Symmetric-Key Ciphers: Introduction, Substitution Ciphers, Transpositional Ciphers, Stream and Block Ciphers. Data Encryption Standard (DES): Introduction, DES Structure, DES Analysis, Security of DES, Multiple DES, Examples of Block Ciphers influenced by DES. Advanced Encryption Standard: Introduction, Transformations, Key Expansion, The AES Ciphers, Examples, Analysis of AES.

[12 Hours]

Unit III

Encipherment using Modern Symmetric-Key Ciphers: Use of Modern Block Ciphers, Use of Stream Ciphers, Other Issues. Mathematics of Asymmetric-Key Cryptography: Primes, Primality Testing, Factorization, Chinese Remainder Theorem, Quadratic Congruence, Exponentiation and Logarithm. Asymmetric Key Cryptography: Introduction, RSA Cryptosystem, Rabin Cryptosystem, Elgamal Cryptosystem, Elliptic Curve Cryptosystems.

[12 Hours]

Unit - IV

Cryptography Hash Functions: Introduction, Description of MD Hash Family, Whirlpool, SHA-512. Digital Signature: Comparison, Process, Services, Attacks on Digital Signature, Digital Signature Schemes, Variations and Applications. Key Management: Symmetric-Key Distribution, Kerberos, Symmetric-Key Agreement, Public-Key Distribution, Hijacking.

[12 Hours]

Unit - V

Security at the Application Layer: PGP and S/MIME: Email, PGP, S/MIME. Security at the Transport Layer: SSL and TLS: SSL Architecture, Four Protocols, SSL Message Formats, Transport Layer Security. Security at the Network Layer: IPSec: Two modes, Two security protocols, Security association, security policy, Internet Key exchange, ISAKMP. [12 Hours]

Text Book:

1. Behrouz A. Forouzan, Debdeep Mukhopadhyay: Cryptography and Network Security, 2nd Edition, Special Indian Edition, Tata McGraw-Hill, 2011.

Reference Books:

1. Michael E. Whitman and Herbert J. Mattord: Principles of Information Security, 2nd Edition, Thomson, Cengage Delmar Learning India Pvt., 2012.
2. William Stallings: Network Security Essentials: Applications and Standards, 4th Edition, Pearson Education, 2012.

BCA604T: WEB PROGRAMMING

Total Teaching Hours : 60 No of Hours / Week : 04

Unit - I

Fundamentals of Web: Internet, WWW, Web Browsers, and Web Servers, URLs, MIME, HTTP, Security, The Web Programmers Toolbox. XHTML: Origins and evolution of HTML and XHTML, Basic syntax, Standard XHTML document structure, Basic text markup, Images, Hypertext Links, Lists, Tables.

[12 Hours]

Unit - II

HTML and XHTML: Forms, Frames in HTML and XHTML, Syntactic differences between HTML and XHTML. CSS: Introduction, Levels of style sheets, Style specification formats, Selector forms, Property value forms, Font properties, List properties, Color, Alignment of text, The Box model, Background images, The and tags, Conflict resolution.

[12 Hours]

Unit -III

Java Script: Overview of JavaScript; Object orientation and JavaScript; General syntactic characteristics; Primitives, Operations, and expressions; Screen output and keyboard input; Control statements; Object creation and Modification; Arrays; Functions; Constructor; Pattern matching using expressions; Errors in scripts; Examples.

[12 Hours]

Unit - IV

Java Script and HTML Documents: The JavaScript execution environment; The Document Object Model; Element access in JavaScript; Events and event handling; Handling events from the Body elements, Button elements, Text box and Password elements; The DOM 2 event model; The navigator object; DOM tree traversal and modification.

Unit - V

Dynamic Documents with JavaScript: Introduction to dynamic documents; Positioning elements; Moving elements; Element visibility; Changing colors and fonts; Dynamic content; Stacking elements; Locating the mouse cursor; Reacting to a mouse click; Slow movement of elements; Dragging and dropping elements. XML: Introduction; Syntax; Document structure; Document Type definitions; Namespaces; XML schemas; Displaying raw XML documents; Displaying XML documents with CSS; XSLT style sheets; XML Processors; Web services.

[12 Hours]

Text Books

1. Robert W Sebesta, "Programming the World Wide Web", 4th Edition, Pearson Education, 2008.

Reference Books

1. M.Deitel, P.J.Deitel, A.B.Goldberg, "Internet & World Wide Web How to program", 3rd Edition, Pearson Education / PHI, 2004.
2. Chris Bates, "Web Programming Building Internet Applications", 3rd Edition, Wiley India, 2006.
3. Xue Bai et al, "The Web Warrior Guide to Web Programming", Thomson, 2003.
4. Sklar, "The Web Warrior Guide to Web Design Technologies", 1st Edition, Cengage Learning India.

BCA605P : PROJECT WORK

Students should individually develop a project. They should implement their project in college in any RDBMS package or any language available in the college. The project should web based. The students have to collect data outside practical hours. Project may be taken outside but must be implemented in the college. Internal marks can be awarded by the guide by evaluating the performance of the students during the course of project work. In viva-voce the questions must be directed only on the project work to access the involvement and understanding of the problem by the students.

The project carries 200 marks is distributed as follows:

Demonstration and Presentation	130 Marks
Viva-voce	50 Marks
Project Report	20 Marks

* * * * * * * * *

BCA604P : WEB PROGRAMMING LAB

PART -A

1. Write a program to find factorial of list of number reading input as command line argument.
2. Write a program to sort list of element in ascending and descending order and show the exception handling.
3. Write a program to implement all string operations.
4. Write a program to find area of geometrical figures using method overloading.
5. Write a program to implement constructor overloading by passing different number of parameter of different types.
6. Write a program to create student report using applet, read the input using text boxes and display the o/p using buttons.
7. Write a program to implement an apply by passing parameter to HTML.
8. Write a program to implement thread, applets and graphics by implementing animation of ball moving.
9. Write a program to implement mouse events.
10. Write a program to implement keyboard events.

PART – B

During practical examination the External and Internal examiners may prepare exam question paper related to theory syllabus apart from Part-A. (A minimum of 10 Programs has to be prepared).

Note :

a) The candidate has to write both the programs One from Part-A and other from Part-B and execute one program as of External examiner choice.

b) A minimum of 10 Programs has to be done in Part-B and has to be maintained in the Practical Record.

c) Scheme of Evaluation is as follows:

Writing two programs	- 10 Marks
Execution of one program	- 10 Marks
Formatting the Output	- 05 Marks
Viva	- 05 Marks
Record	- 05 Marks
Total	**- 35 Marks**

BCA604P : WEB PROGRAMMING LAB

PART - A

1. Create a form having number of elements (Textboxes, Radio buttons, Checkboxes, and so on). Write JavaScript code to count the number of elements in a form
2. Create a HTML form that has number of Textboxes. When the form runs in the Browser fill the textboxes with data. Write JavaScript code that verifies that all textboxes has been filled. If a textboxes has been left empty, popup an alert indicating which textbox has been left empty.
3. Develop a HTML Form, which accepts any Mathematical expression. Write JavaScript code to Evaluates the expression and Displays the result.
4. Create a page with dynamic effects. Write the code to include layers and basic animation.
5. Write a JavaScript code to find the sum of N natural Numbers. (Use user-defined function)
6. Write a JavaScript code block using arrays and generate the current date in words, this should include the day, month and year.
7. Create a form for Student information. Write JavaScript code to find Total, Average, Result and Grade.
8. Create a form for Employee information. Write JavaScript code to find DA, HRA, PF, TAX, Gross pay, Deduction and Net pay.
9. Create a form consists of a two Multiple choice lists and one single choice list

(a) The first multiple choice list, displays the Major dishes available

(b) The second multiple choice list, displays the Starters available.

(c)The single choice list, displays the Soft drinks available.

10. Create a web page using two image files, which switch between one another as the mouse pointer moves over the image. Use the on Mouse Over and on Mouse Out event handlers.

PART – B

During practical examination the External and Internal examiners may prepare exam question paper related to theory syllabus apart from Part-A. (A minimum of 10 Programs has to be prepared).

Note :

a) The candidate has to write both the programs One from Part-A and other from Part-B and execute one program as of External examiner choice.

b) A minimum of 10 Programs has to be done in Part-B and has to be maintained in the Practical Record.

c) Scheme of Evaluation is as follows:

Writing two programs	- 10 Marks
Execution of one program	- 10 Marks
Formatting the Output	- 05 Marks
Viva	- 05 Marks
Record	- 05 Marks
Total	- **35 Marks**

US – 644

VI Semester B.C.A. Examination, May 2017
(2016 – 17 & Onwards) (CBCS)
COMPUTER SCIENCE
BCA 601 : Theory of Computation

Time : 3 Hours | Max. Marks : 100

***Instruction** : Answer **all** Sections.*

SECTION – A

Answer **any ten** questions. **Each** question carries **two** marks. **(10×2=20)**

1. Define Finite Automata.
2. Define DFA. Mention the types of Finite Automata.
3. Build an regular expression that generates a string with even number of 0's followed by odd number of 1's.
4. What is Pumping Lemma ?
5. What are terminal and non-terminal symbols in grammar ?
6. What is left most derivation in CFG ?
7. What are the different types of grammar ?
8. Mention the 7 types of PDA.
9. Define GNF.
10. What are useful and useless symbols in grammar ?
11. What is Turing Machine ?
12. What are the different types of Turing Machine ?

SECTION – B

Answer **any five** questions. **Each** question carries **five** marks. **(5×5=25)**

13. Mention five differences between DFA and NFA.
14. Construct a DFA to accept the string 'abba'.

P.T.O.

15. Explain the various applications of Regular expressions.

16. Obtain the left most and right most derivations for the string 00112. The production rules are given by

$P = \{S \rightarrow AB$

$A \rightarrow 01 \mid 0A1$

$B \rightarrow \in \mid 2B$

17. Prove that $S \rightarrow aSbS/bSaS/\in$ is ambiguous.

18. Write a short note on Chomsky hierarchy of languages.

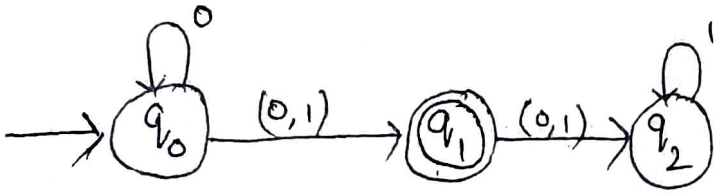
19. Write down the steps for conversion of DFA to CFG.

20. Explain halting problem of Turing Machine.

SECTION – C

Answer **any three** questions. **Each** question carries **fifteen** marks. **(15x3=45)**

21. Convert the following NFA to its equivalent DFA.



22. Construct a NFA with $\in$ for $(0 + 1)^* 1 (0 + 1)$.

23. Explain the block diagram of Pushdown automata with its components, specification, language and transition table.

24. Transform the CFG into GNF

$S \rightarrow AB$

$A \rightarrow BS \mid 1$

$B \rightarrow SA \mid 0$

25. a) Explain Post's Correspondence Problem (PCP). **8**

b) Explain intersection and homomorphism property of Regular languages. **7**

SECTION – D

Answer **any one** question.

26. Find the minimized DFA for the following transition table : **10**

	δ	a	b
→	A	B	A
	B	A	C
	C	D	B
	*D	D	A
	E	D	F
	F	G	E
	G	F	G
	H	G	D

27. Design a Turing Machine that accepts the language of all strings over the alphabet $\sum = \{a, b\}$ whose second letter is 'b'. **10**

SM – 623

VI Semester B.C.A. Examination, May/June 2018
(CBCS) (F +R)
(2016 – 17 & Onwards)
COMPUTER SCIENCE
BCA 601 : Theory of Computation

Time : 3 Hours Max. Marks : 100

Instruction : *Answer **all** Sections.*

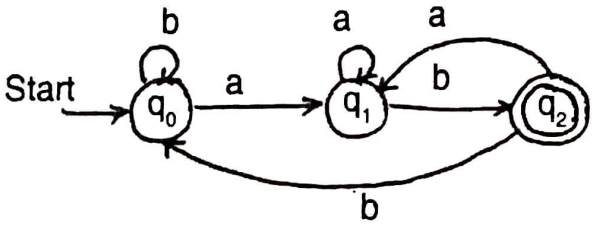
SECTION – A

Answer **any ten** questions. **Each** question carries **two** marks. **(10×2=20)**

1. What is finite automata ? Explain with block diagram.

2. What is trap state ? Explain with a simple example.

3. What are the moves made by the following DFA while processing the string abaab ? Find if the string is accepted or rejected by DFA.



4. Design a regular expression over $\Sigma = \{a, b\}$ for the language accepting string of exactly length 2.

5. State pumping Lemma for regular languages.

6. State Arden's theorem.

7. Define grammar. Give one example.

P.T.O.

8. Mention any two applications of context free grammar.

9. Define Nullable variable.

10. Define GNF.

11. Define turing machine.

12. Define recursively enumerable language.

SECTION – B

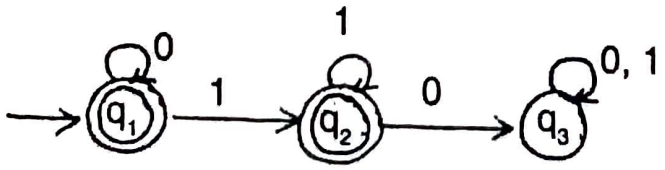
Answer **any five** questions. **Each** question carries **five** marks. **(5×5=25)**

13. Construct a DFA to accept string of 0's and 1's representing zero modulo five.

14. Define NFA. Obtain a NFA to accept the language L = {w/w∈ abab^n or aba^n where n ≥ 0}.

15. Using pumping Lemma prove the language L = {yy/y∈ (0.1)*} is not regular.

16. Convert the DFA to Regular Expression.



17. Define context free grammar.

    Consider a grammar G = (V, T, P, S) where V = {S} T = {a, b} S = S P = {S→aS|b}. Find the language accepted by G.

18. Explain Chomsky hierarchy of grammar.

19. Eliminate useless symbols from the following grammar

$S \rightarrow aAa$

$A \rightarrow Sb$

$A \rightarrow bcc$

$A \rightarrow DaA$

$C \rightarrow abb$

$C \rightarrow DD$

$E \rightarrow ac$

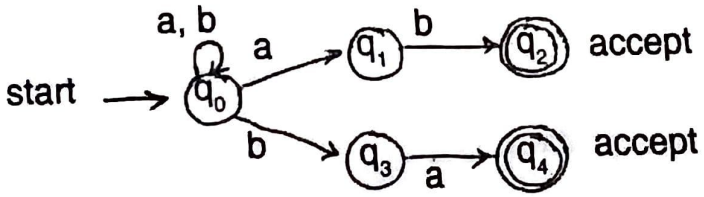
$D \rightarrow aDa$

20. What are the different types of turing machine ?

SECTION – C

Answer **any three** questions. **Each** question carries **fifteen** marks. **(15×3=45)**

21. Convert the following NFA to DFA using lazy evaluation method.



22. Minimize the following DFA using table filling algorithm.

	δ	a	b
→	A	B	C
	B	G	C
*	C	A	C
	D	C	G
	E	H	F
	F	C	G
	G	G	E
	H	G	C

No. of Printed Pages : 3

GS-642

VI Semester B.C.A. Examination, May/June 2019
(CBCS - F+R) (2016-17 & onwards)

COMPUTER SCIENCE

BCA 601 : Theory of Computation

Time : 3 Hours

Max. Marks : 100

Instruction : *Answer all sections.*

SECTION - A

Answer **any ten** questions. Each question carries **two** marks. **10x2=20**

1. Define DFA. With Mathematical Representation.

2. Define Alphabet and Symbol with example.

3. What is trap state ?

4. Define Regular Expression.

5. Design RE (Regular Expression) for the language containing any number of a's and b's ending with aa.

6. What is Pumping Lemma ?

7. Mention the types of chomsky hierarchy grammer.

8. Define PDA (push down Automata).

9. Define GNF (Greibach Normal Form).

10. What is turing machine ?

11. Define PCP (Post Correspondence Problem).

12. State Arden's Theorem.

P.T.O.

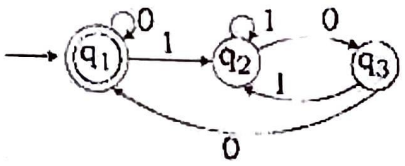
SECTION - B

Answer **any five** questions. Each question carries **five** marks. **5x5=25**

13. Construct a DFA to accept strings of O's & L's ending with 101.

14. Write difference between DFA and NFA.

15. Convert the DFA to Regular Expression.



16. State and Prove Pumping Lemma.

17. Obtain a CFG (Context free grammer) for the following Langauge $L = \{a^n b^n \mid n \geq 1\}$.

18. Explain Halting Problem of Turing machine.

19. Eliminate the unit production from the grammer.

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow c$

$B \rightarrow b$

$C \rightarrow D$

$D \rightarrow E$

$E \rightarrow a$

20. Show that the following grammer is ambiguous.

$E \rightarrow E + E$

$E \rightarrow E - E$

$E \rightarrow E * E$

$E \rightarrow E \mid E$

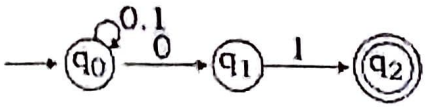
$E \rightarrow |E|$

$E \rightarrow id$

SECTION - C

Answer **any three** questions. Each question carries **fifteen** marks. **3x15=45**

21. Convert the following NFA to DFA.



22. Minimize the given DFA using table filling Algorithm.

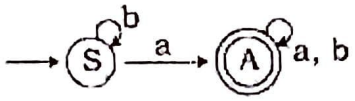
δ	0	1
→ A	B	D
B	C	E
C	B	E
D	C	E
* E	E	E

23. Construct a PDA to accept the language
$L(M) = \{ww^R | w \in (a+b)^*\}$ where w^R is the reverse of w by final state acceptance.

24. Find the language accepted by CFG .

(a) $G = \{V, T, P, S\}$
$V = \{s\}$
$T = \{a, b\}$
$S = S$
$P = \{S \rightarrow aS|b\}$

(b) Obtain a grammer to generate string $S = \{a, b\}$ having atleast one a.



(c) Obtain a CFG for the language.
$L = \{wcw^R | w \in \{a, b\}^*\}$

25. Obtain a turing machine to accept the language $L = \{a^n b^n | n \geq 1\}$.

SECTION - D

Answer **any one** questions. **1x10=10**

26. Contruct the NFA with E-moves for
$(0+1)^* \ 1(0+1)$

27. Explain the types of Turing Machine.

- o 0 o -

THEORY OF COMPUTATION

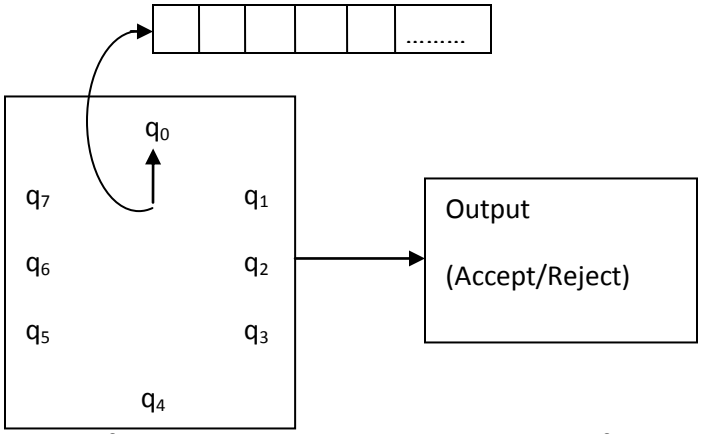
SOLUTION BANK

Unit - 1

1. Define finite automata.

Finite automata is a mathematical model which is used to study the abstract machines or abstract computing devices with the input chosen from ∑.

Block diagram:-



Input file:- Input file trope is divided into cells each of which can hold symbol. The string is processed and stored in these cells.

Control unit:- The machines has some states one of which is the start state designed as q_0 and at least one final state.

Output:- o/p may be accepted or rejected when end of the input is encountered. The control unit may be in accept or reject state.

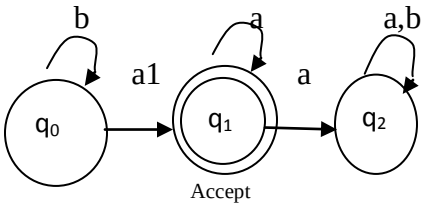
2. Define DFA. Mention the types of finite automata.

DFA is a finite automata which can have only one transition from a state on an input symbol. Types of finite automata:

- Deterministic finite automata(DFA)
- Non-Deterministic finite automata(NFA)
- Non-Deterministic finite automata with € moves(E-NFA)

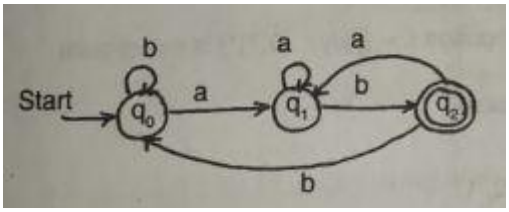
3. What is trap state? Explain with block diagram. [2018]

A state for which there exists transitions to itself for all the input symbols chosen from ∑.



trap state

4. **What are the moves made by the following DFA while processing the string abaab? Find if the string is accepted or rejected by DFA.**



The moves are:-

$\delta(q_0,a)=q_1$

$\delta(q_0,b)=q_0$

$\delta(q_1,a)=q_1$

$\delta(q_1,b)=q_2$

$\delta(q_2,a)=q_1$

$\delta(q_0,b)=q_0$

At the end of the string abaab the DFA will be in the state q_2 which is in the final state.

So the string abaab is accepted by the machine.

5. **Define alphabet and symbol with example.** **[2019]**

An alphabet is a finite nonempty set of symbols. Conventionally we use the symbol summession symbol for an alphabet

Example: If summession symbol = {0, 1}, then Power of 1 summession symbol = {0,1}

Power of 2 summession symbol = {00, 01, 10, 11}

Unit – 2

1. **Define regular expression.**

The language accepted by finite automata is called regular language. A regular language can be described using regular expressions, consisting of alphabets in ∑ and the operators '*','.','+'. The order of evaluation of regular expression is determined by parenthesis and the operator precedence '*','.' And '+' respectively.

2. **Build a regular expression that generates a string with even number of 0's followed by odd number of 1's.**

(00)*(11)*1

3. **What is pumping lemma?**

Pumping lemma is a method of pumping (generating) many input string from a given string it is used to show that certain languages are not regular.

4. **Design a regular expression over ∑=(a,b) for the language accepting string of exactly length 2.**

L={aa,ab,ba,bb} (abs)(a+b)

5. Define PDA (push down automata).

A pushdown automaton (PDA) is a finite state machine which has an additional stack storage. The transitions a machine makes are based not only on the input and current state, but also on the stack.

6. State pumping lemma for regular languages.

If A is a regular language then A has a pumping length 'p' such that any string's' where |s|>=p may be divided into 3 parts s=xyz such that the following conditions may be true.

I. $xy^iz \in A$

II. |y|>0

III. |xy|<=p

7. State Arden's theorem. (2019)

If P and Q are two regular expressions over , and if P does not contain , then the following equation in R given by R = Q + RP has an unique solution i.e., R = QP*." That means, whenever we get any equation in the form of R = Q + RP, then we can directly replaced by R = QP*. So, here first we will prove that R = QP* is the solution of this equation and then we will also prove that it is the unique solution of this equation.

Unit – 3

1. What are terminal and non terminal symbols in grammer?

Non-terminals are syntactic variables that denote sets of strings. The non-terminals define sets of strings that help define the language generated by the grammar. A set of tokens, known as **Terminal** symbols (Σ). Terminals are the basic symbols from which strings are formed.

2. What is left most derivation in CFG?

A derivation A^* => w is called left most derivation if we apply a production only to the left most variable at every step.

3. What are the different types of grammar?(2019)

There are 4 types of grammer:-

- Type 0 Grammer (Phrase structured/ unrestricted grammer)
- Type 1 Grammer(Context sensitive grammer)
- Type 2 Grammer(Context free grammer)
- Type 3 Grammer(Regular grammer)

4. Mention the 7 types of PDA.

A push down automata (PDA) is a seven tuple M=(Q, ∑, ~, δ, q_0 , z_0 , F)

Where Q= is a set finite sets

∑= set of input alphabets

~= set of stack alphabets

δ= transition from Q x ($\sum \cup \in$) x ~ to finite subset of QXI*.

$Q_0 \in Q$ is the start state of M

$z_0 \in$ ~ is the initial symbol on the stack

F⊆0 is set of initial states

5. Define grammar. Give one example

A grammer is a quad tuple G (V, T, P ,S) where, V is a finite set of variables or non terminals.

T is a finite set of terminals

P is a finite set of production rules. Each production is of the following from A→a where, A is a string of symbol from $(V \cup T)^*$

$\propto$ Is a string of symbol from $(V \cup T)^*$

S is the start symbol & S∈V

Example: S→aAb/∈

6. Mention any two applications of context free grammar.

- ✓ Parsers
- ✓ Markup language
- ✓ Finite automata
- ✓ Digital design

7. Define nullable variable.

Let G=(V,T,P,S) be a CFG. A nullable variable is defined as

a) If A → ∈ is a P, then A is a nullable variable.

b) If A→B_1,B_2…..B_n, is a production in P and if B_1, B_2……B_n are nullable variables, then A is also a nullable variable.

c)The variable for which these production of the form shown in a & b are nullable variables.

TMA Questions:

1. Define ID of PDA.

Unit – 4

1. Define GNF. (2019)

Let G= (V,T,P,S) be a CFG. The CFG is a said to be in GNF if all the production are of the form A→a$\propto$

Where a∈T$\propto$∈V^* i.e., the first symbol on the right hand side of the production must be a terminal & it can be followed by 0 or more variable.

2. What are useful and useless symbols in grammer?

In a CFG, G=(V,T,P,S), x is useless, if it does not satisfy either of the following condition

(a) $\propto^*$ => w, where w is in T^*.

(b)

TMA Questions:

1. What is left recursion.

2.What is parsing.

Unit – 5

1. What is Turing machine? (2019)

The Turing machine M = (Q, ∑, ~, δ, q_0 , B , F)

Where, Q is the set of finite states

∑ is the set of input alphabets

~ is the set of tape symbols

δ is the transition function Q x ~ to Q x ~ x (L,R)

q_0 is the start state

B is the special symbol indicating blank character.

F is the set of final state

2. What are the different types of Turing Machine?

- Multi tape Turing Machine
- Non-deterministic Turing Machine
- Multi-dimensional Turing Machine
- Multi Read Turing Machine

3. Define recursively enumerable language.

Recursively enumerable (RE) language are generated by type 0 grammer. A recursive enumerable language can be accepted or recognized by Turing machine which means it will enter into final state for the string of language and may or may not enter into rejecting state for the string which is not part of the language. It means Turing machine can loop forever for the string which are not a part of the language. RE language are also called Turing recognizable language.

4. **Define PCP (Post Correspondence Problem).**

The Post Correspondence Problem (PCP), introduced by Emil Post in 1946, is an undecidable decision problem. The PCP problem over an alphabet ∑ is stated as follows − Given the following two lists, **M** and **N** of non-empty strings over ∑ −

M = (x_1, x_2, x_3,……., x_n)

N = (y_1, y_2, y_3,……., y_n)

We can say that there is a Post Correspondence Solution, if for some i_1,i_2,…… i_k, where $1 \leq i_j \leq n$, the condition x_{i1} …….x_{ik} = y_{i1} …….y_{ik} satisfies.

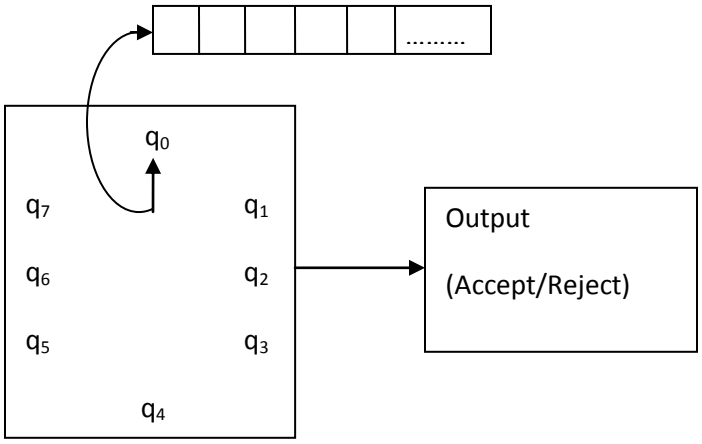
VI SEMESTER BCA SOLUTION BANK
THEORY OF COMPUTATION

@2 Marks

1. Define finite automata? Explain with the block diagram.

Finite automata are a mathematical model which is used to study the abstract machines or abstract computing devices with the input chosen from ∑.

Block diagram: -



Input file:- Input file trope is divided into cells each of which can hold symbol. The string is processed and stored in these cells.
Control unit:- The machines has some states one of which is the start state designed as q_0 and at least one final state.
Output:- o/p may be accepted or rejected when end of the input is encountered. The control unit may be in accept or reject state.

2. Define DFA. Mention the types of finite automata.

DFA is a finite automata which can have only one transition from a state on an input symbol. Types of finite automata:

- Deterministic finite automata(DFA)
- Non-Deterministic finite automata(NFA)
- Non-Deterministic finite automata with € moves(E-NFA)

3. Build a regular expression that generates a string with even number of 0's followed by odd number of 1's.

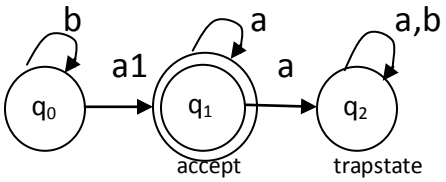
(00)*(11)*1

4. What is pumping lemma?

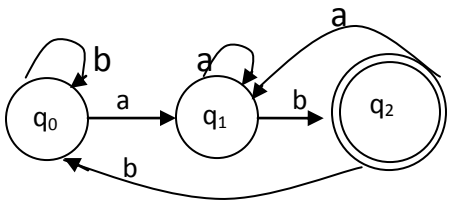
Pumping lemma is a method of pumping(generating) many input string from a given string it is used to show that certain languages are not regular.

5. What is trap state? Explain with block diagram.

A state for which there exists transitions to itself for all the input symbols chosen from ∑.



6. What are the moves made by the following DFA while processing the string abaab? Find if the string is accepted or rejected by DFA.



The moves are: -

$\delta(q_0,a)=q_1$

$\delta(q_0,b)=q_0$

$\delta(q_1,a)=q_1$

$\delta(q_1,b)=q_2$

$\delta(q_2,a)=q_1$

$\delta(q_0,b)=q_0$

At the end of the string abaab the DFA will be in the state q_2 which is in the final state.

So the string abaab is accepted by the machine.

7. Design a regular expression over ∑={a,b} for the language accepting string of exactly length 2.

L={aa,ab,ba,bb} (a+b)(a+b)

8. State pumping lemma for regular languages.

If A is a regular language, then A has a pumping length 'p' such that any string 's' where |s|>=p may be divided into 3 parts s=xyz such that the following conditions may be true.

I. $xy^iz \in A$

II. |y|>0

III. |xy|<=p

9. Define DFA with mathematical representation.

DFA is a finite automata which can have only 1 transition from a state on an input symbol.

Mathematical representation: -

DFA is a five tuple $(Q, \sum, \delta, q_0, F)$

Where Q- non empty finite set of states

$\sum$- non empty finite set of m/p symbols

δ: $Q \times \sum \rightarrow Q$ is a transition function

q_0,Q is start state

F is the final state

10. Define regular expression.

The language accepted by finite automata is called regular language. A regular language can be described using regular expressions, consisting of alphabets in $\sum$ and the operators '*','.','+'. The order of evaluation of regular expression is determined by parenthesis and the operator precedence '*','.' And '+' respectively.

@5 Marks

1. Mention 5 differences between DFA and NFA.

DFA	NFA
• DFA is a 5 tuple. D= $Q,\sum,\delta,q_0,F$ δ: $Q \times \sum \rightarrow Q$ • It can have only one transition	• NFA is a 5 tuple. N= $Q,\sum,\delta,q_0,F$ δ: $Q \times \sum \rightarrow 2^Q$ • It can have zero, one or more

from a state on an i/p symbol. • Difficult to construct • Less powerful since at any point of time it will be in only one state.	transitions from a state on an i/p. • Easy to construct • More powerful than DFA since at any point of time it will be in more than one state.

2. Explain the various applications of regular expressions.

- ✓ Design of compilers
- ✓ To define languages
- ✓ Declarative way to express set of strings
- ✓ Validation – i.e., checking the correction of i/p
- ✓ Tokenization– i.e., conversion of string of characters into a sequence of words for later interpretation in pattern matching.
- ✓ Test for a pattern within a string.
- ✓ Replace text in a document.
- ✓ Extract a substring from a string based upon a pattern match.
- ✓ Used in languages like JScript and e for string handling.
- ✓ Helps in implementing complex match logic in databases.

3. Construct a DFA to accept string of 0's and 1's representing zero modulo five.

Step 1: Identify radix, input alphabets and the divisor.

r=2 d={0,1} k=5

Step 2: Compute the possible remainders.

i = 0,1,2,3,4

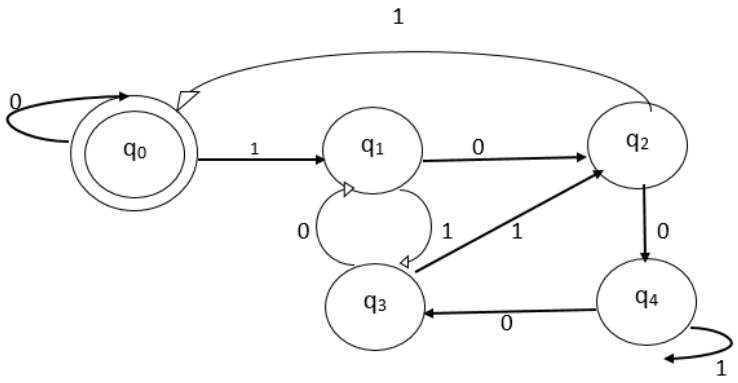
Step 3: Compute the transitions using the relation

$\delta(q_i,d)=q_j$ where $j=(r*i+d) \bmod k$

remainder	d	(2*i+d)mod5=j	$\delta(q_i,d)=q_j$
i=0	0 1	(2*0+0)mod5=0 (2*0+1)mod5=1	$\delta(q_i,0)= q_0$ $\delta(q_i,1)=q_1$
i=1	0 1	(2*1+0)mod5=2 (2*1+1)mod5=3	$\delta(q_i,0)=q_2$ $\delta(q_i,1)=q_3$
i=2	0 1	(2*2+0)mod5=4 (2*2+1)mod5=0	$\delta(q_i,0)=q_4$ $\delta(q_i,1)=q_0$

i=3	0 1	(2*3+1)mod5=1 (2*3+1)mod5=2	$\delta(q_i,0)=q_1$ $\delta(q_i,1)=q_2$
i=4	0 1	(2*4+1)mod5=3 (2*4+1)mod5=4	$\delta(q_i,d)=q_3$ $\delta(q_i,d)=q_4$

Step 4: Construct the DFA



The DFA D=(Q,∑,δ,q_0,F)

Where Q={ q_0 ,q_1 ,q_2 ,q_3 ,q_4}

∑={0,1}

q_0={ q_0}

F={q_0}

δ is shown using the transition table

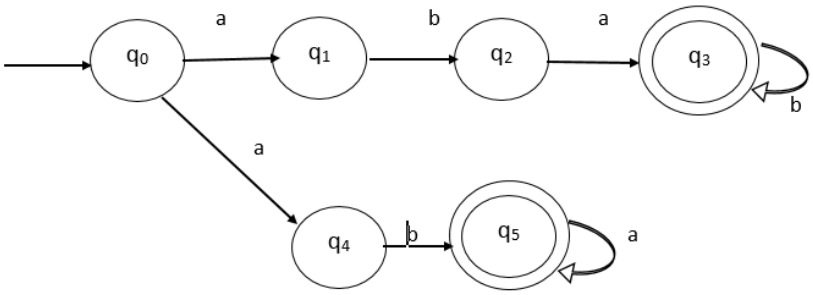
δ	0	1
q_0	{q_0}	{q_1}
q_1	{q_2}	{q_3}
q_2	{q_4}	{q_0}
q_3	{q_1}	{q_2}
q_4	{q_3}	{q_4}

4. Define NFA. Obtain a NFA to accept the language L={W/W € $abab^n$ or aba^n where>=0}.

The NFA N=(Q,∑,δ,q_0,F)

Where Q={ q_0 ,q_1 ,q_2 ,q_3 ,q_4 ,q_5}

∑={a,b}



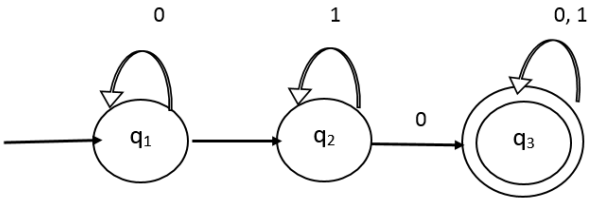
$q_0=\{q_0\}$

$F= \{q_3, q_5\}$

δ is shown using the transition table

δ	a	b
q_0	$\{q_1,q_4\}$	-
q_1	-	$\{q_2\}$
q_2	$\{q_3\}$	-
q_3	-	$\{q_3\}$
q_4	-	$\{q_5\}$
q_5	$\{q_5\}$	-

5. Convert the DFA to regular expression.



Step 1: q_1 is the start state.

Step 2: Calculate q_1

$q_1 = q_1\, 0 + \in$

By rearranging

$q_1 = \in + q_1\, 0$

$q_1 = 0^*$

Step 3: Since q_2 is the final state, calculate q_2,

$q_2 = q_2\ 1 + q_1\ 1$

$q_2 = q_2\ 1 + 0^*\ 1$

By rearranging

$q_2 = 0^*\ 1 + q_2\ 1$

$q_2 = 0^*\ 1\ 1$

$q_2 = 0^*\ 1^*$

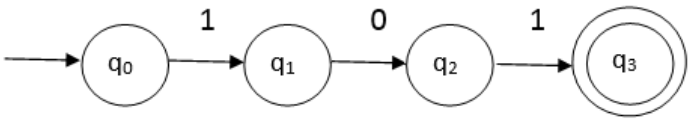
Since q_1 and q_2 are the final states the required regular expression is $0^* + 0^*\ 1^*$

6. Construct a DFA to accept strings of 0's and 1's ending with 101.

Step1: Minimum string=101

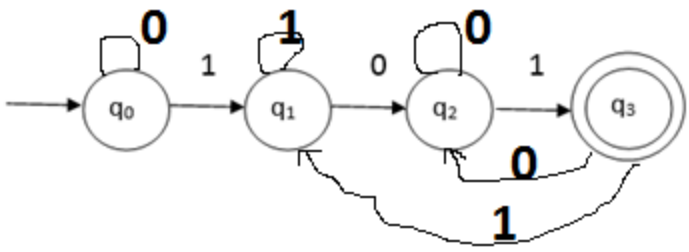
Step2: ∑={0,1}

Step3: Skeleton DFA



Step4: Identify the other undefined transition

δ	a	b
q_0	?	$\{q_1\}$
q_1	$\{q_2\}$	?
q_2	?	$\{q_3\}$
q_3	?	?



Step5: The DFA is defined as D=(Q,∑,δ,q_0,F)

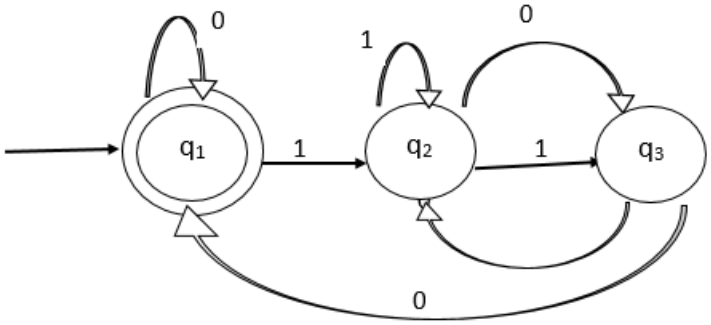
Q={ q_0 ,q_1 ,q_2 ,q_3}

∑={0,1}

q_0={ q_0}

F={q_3}

7. Convert the DFA to regular expression.



Step 1: q_1 is the start state

Step 2: Calculate q_1

$q_1 = q_1 0 + q_3 0 + \in$

Step 3: Calculate q_2, q_3

$q_2 = q_2 1 + q_3 1 + q_1 1$

$q_3 = q_2 0$

Substitute q_3 in q_2

$q_2 = q_2 1 + q_2 1\ 0 + q_1 1$

$q_2 = q_2 (1+01) + q_1 1$

$q_2 = q_1 1 + q_2 (1+01)$

$q_2 = q_1 1(1+01)$ *

Substitute in q_1

$q_1 = q_1 0 + q_3 0 + \in$

$q_1 = q_1 0 + q_2 00 + \in$

$q_1 = q_1 0 +[q_1 1(1+01) *] 00 + \in$

$q_1 = q_1[0+1(1+01) *00] + \in$

$q_1 = \in + q_1[0+1(1+01) *00]$

$q_1 = \in [0+1](1+01) *00]*$

$q_1 = [0+1] (1+01) *00] *$

Since q_1 is the final state the required regular expression [0+1] (1+01) *00] *

8. State and prove pumping lemma

Pumping lemma is used to prove that a language is not regular.

Theorem Statement: - if A is regular language then A has a pumping length 'p' such that any string 's' where |s|>=p may be divided into 3 parts s=xyz such that following conditions must be true.

i. xyz € A for every i>=0
ii. |y|>0
iii. |xy|<=p

Sol: - $L=\{a^n b^n\}$ p=7

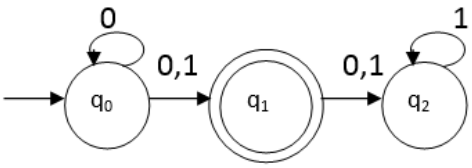
$S= a^n b^n$

$= a^7 b^7$

=aaaaaaabbbbbbb

x y z

9. Convert the following NFA to its equivalent DFA



The given NFA N=(Q, ∑,δ,q_0,F)

Q={q_0 ,q_1 ,q_2 }

∑= {0,1}

q_0={q_0}

F={q_1}

The equivalent DFA D = (Q_0 ,∑, δ_0, q_0, F_0)

∑ = {0, 1}

Step 01: q_0 = {q_0}, q_0 is the start state of DFA, D

Step 02: δ_0 from q_0 on ∑

$$\delta_0(q_0, 0) = \delta_N(q_0, 0)$$

$$= \{q_0, q_1\}$$

$$\delta_0(q_0, 1) = \delta_N(q_1, 0)$$

$$= \{q_1\}$$

δ_0 from {q_2} on ∑

$$\delta_0(\{q_2\}, 0) = \delta_N(q_2, 0)$$

$$= \emptyset$$

$$\delta_0(\{q_2\}, 1) = \delta_N(q_2, 1)$$

$=\{q_2\}$

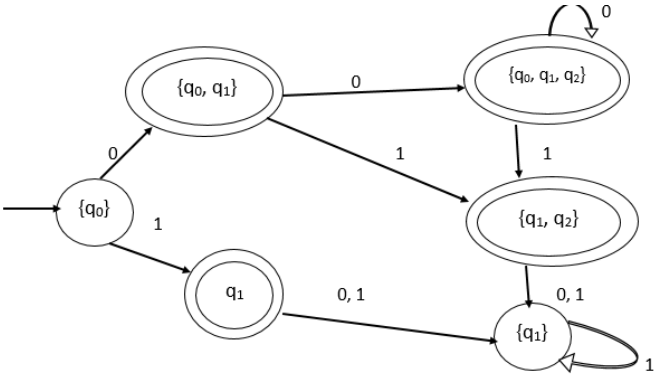
Step 03: The final state $F_0 = \{\{q_0, q_1\}, \{q_1\}, \{q_0, q_1, q_2\}, \{q_1, q_2\}\}$

Therefore, the equivalent DFD D = (Q_0 ,$\sum$, δ_0, q_0, F_0)

where $Q_0 = \{\{q_0\}\ \{q_0, q_1\}, \{q_1\}, \{q_0, q_1, q_2\}, \{q_1, q_2\}, \{q_2\}\}$

$\sum=\{0, 1\}$

$q_0 = q_0$

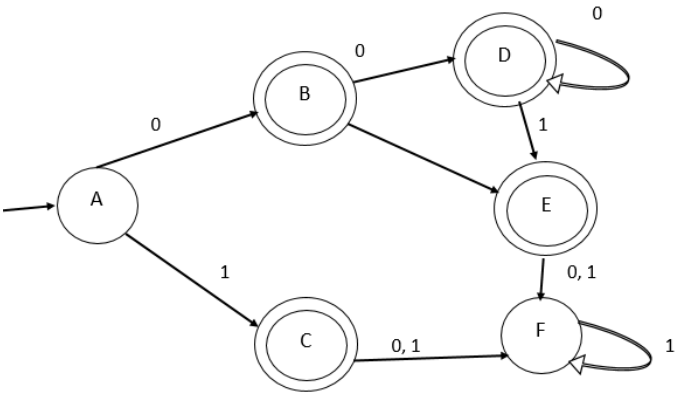


$\{\{q_0, q_1\}, \{q_1\}, \{q_0, q_1, q_2\}, \{q_1, q_2\}\}$

δ_0	0	1
→ A	{B}	{C}
*B	{D}	{E}
*C	{F}	{F}
*D	{D}	{F}
*E	{F}	{F}
F	Ø	{F}

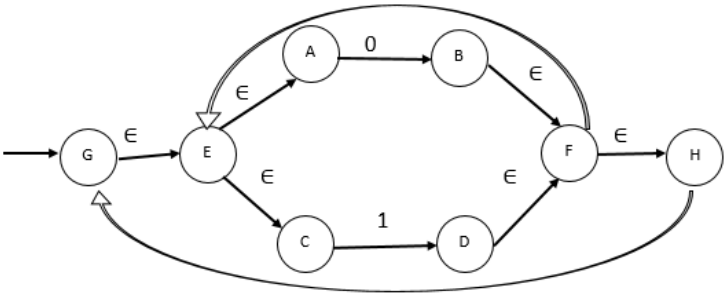
δ_0	0	1
→ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_1\}$
*$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
*$\{q_1\}$	$\{q_2\}$	$\{q_2\}$
*$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
*$\{q_1, q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_1\}$	Ø	$\{q_2\}$

The equivalent DFA

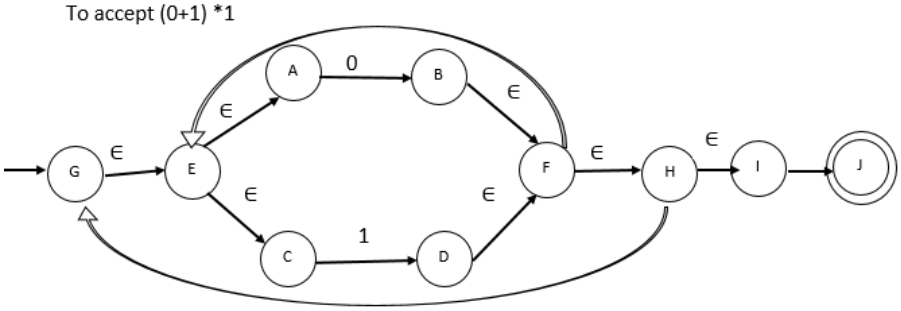


10. Construct a NFA with € for $(0+1)^*1(0+1)$.

To accept (0+1)*

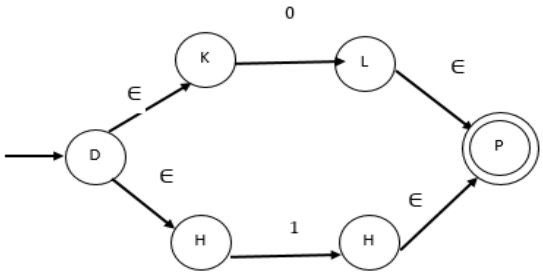


To accept (0+1) *1

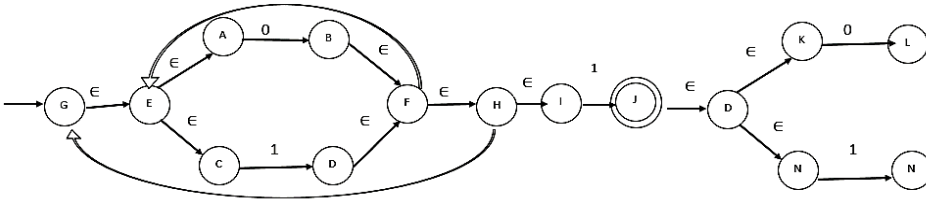


To accept (0+1)

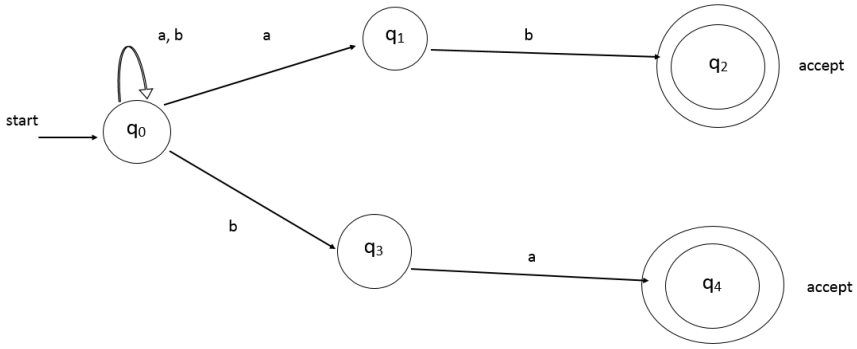
δ_0	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0, q_3\}$
q_1	∅	q_2
* q_2	∅	∅
q_3	q_4	∅
* q_4	∅	∅



To accept (0+1)*1 (0+1)



11. Convert the NFA to DFA using lazy evaluation method.



Step 1: $q_0=\{q_0\}$, q_0 is the start state

δ_0 from q_0 on $\sum$

$\delta_0(q_0, a) = \delta_N(q_0, a) = \{q_0, q_1\}$

$\delta_0(q_0, b) = \delta_N(q_0, b) = \{q_0, q_3\}$

δ_0 from $\{q_2\}$ on $\sum$

$\delta_0(\{q_0, q_3\}\ a) = \delta_N(q_0, a) \cup \delta_N(q_3, a)$

$=\{q_0, q_1\} \cup q_4$

$=\{q_0, q_1, q_4\}$

$\delta_0(\{q_0, q_3\}\ b) = \delta_N(q_0, b) \cup \delta_N(q_3, b)$

$=\{q_0, q_3\} \cup \emptyset$

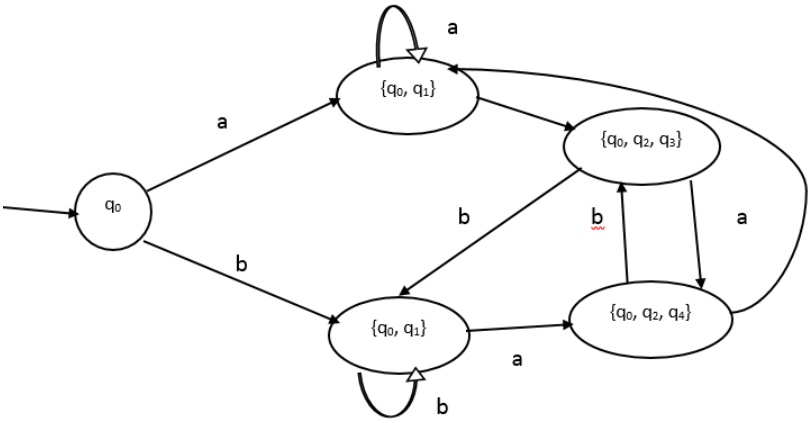
$=\{q_0, q_3\}$

δ_0 from $\{q_0, q_2, q_3\}$ on $\sum$

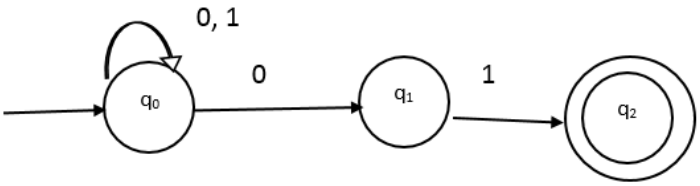
$\delta_0(\{q_0, q_2, q_3\}, a) = \delta_N(q_0, a) \cup \delta_N(q_2, a) \cup \delta_N(q_3, a)$

$=\{q_0, q_1\} \cup \emptyset \cup q_4$

$=\{q_0, q_1, q_4\}$



12. Convert the NFA to DFA



The given NFA, $N=(Q_N, \sum, \delta_N, q_0, F_N)$

where, $Q_N = \{q_0, q_1, q_2\}$

$\sum=\{0, 1\}$

$q_0 = q_0$

$F_N =\{q_2\}$

The equivalent DFA

$D=(Q_0, \sum, \delta_0, q_0, F_0)$

$\sum=\{0, 1\}$

Step 01: Start state of NFA

Therefore, $Q_0 =\{q_0\}$

δ_0 from q_0 on $\sum$

$\delta_0 (q_0, 0) = \delta_N (q_0, 0) = \{q_0, q_1\}$

$\delta_0 (q_0, 1) = \delta_N (q_0, 1) = \{q_0\}$

δ_0 from $\{q_0, q_1\}$ on $\sum$

$\delta_0(\{q_0, q_1\}\ 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0)$

$= \{q_0, q_1\} \cup \emptyset$

$= \{q_0, q_1\}$

$\delta_0(\{q_0, q_1\}\ 1) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$

$= \{q_0\} \cup \{q_2\}$

$= \{q_0, q_2\}$

δ_0 from $\{q_0, q_2\}$ on $\sum$

$\delta_0(\{q_0, q_2\}\ 0) = \delta_N(q_0, 0) \cup \delta_N(q_2, 0)$

$= \{q_0, q_1\} \cup \emptyset$

$= \{q_0, q_1\}$

$\delta_0(\{q_0, q_2\}\ 1) = \delta_N(q_0, 1) \cup \delta_N(q_2, 1)$

$= \{q_0\} \cup \emptyset$

$= \{q_0\}$

Step 3: The final state F_0 is the state in Q_0

$F_0 = \{q_0, q_2\}$

Therefore, The equivalent DFA $\{q_0\}$

$D = (Q_0, \sum, \delta_0, q_0, F_0)$

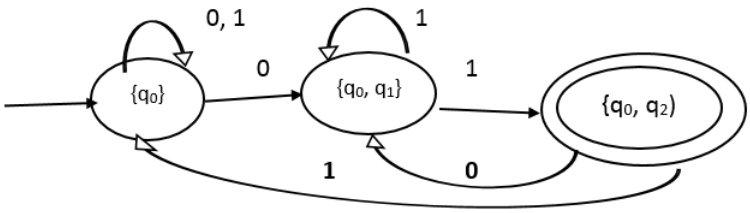
where $Q_0 = \{\{q_0\}, \{q_0, q_1\}, \{q_0, q_2\}\}$

$\sum = \{0, 1\}$

$q_0 = q_0$

$F_N = \{q_0, q2\}$

δ	0	1
-> $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$



16 obtain the left most and right most derivation for the string 00112. The production rules are given by

P= {S -> AB

A-> 01|0A1

B->∈|2B leftmost derivation

S -> AB

OA1B [Since A->OA1]

0011B [Since A>01]

00112B [since B->2B]

00112∈ [Since B->∈]

00112

Right most derivation

S-> AB

A2B [Since B->2B]

A2∈ [since B->∈]

A2

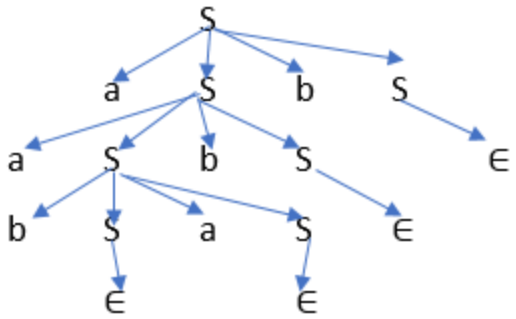
0A12 [Since A-> 0A]

00112[Since A->01]

17. Prove that S-> aSbS | bSaS | ∈ is ambiguous.

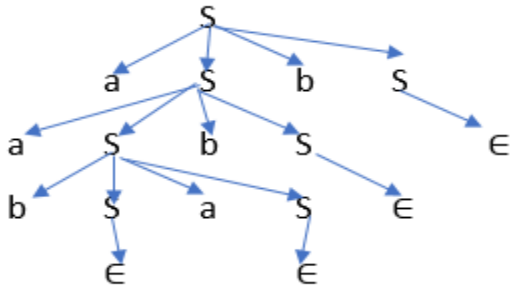
consider the left most derivation for the string aababb and the Corresponding parse tree

s->	asbs	by using S->aSbS
	aaSbSbS	by using S-> aSbS
	aabSaSbSbS	by using S-> bSaS
	aabaSbSbS	by using S-> ∈
	aababSbS	by using S-> ∈
	aababbS	by using S-> ∈
	aababb	by using S-> ∈



Consider the left most derivation again for the string aababb but using different set productions.

s->	asbs	by using S->aSbS
	aaSbSbS	by using S-> aSbS
	aabSbS	by using S-> ∈
	aabaSbSbS	by using S-> aSbS
	aababSbS	by using S-> ∈
	aababbS	by using S-> ∈
	aababb	by using S-> ∈



Since there are two parse tree for the string aababb by applying leftmost derivation the grammar is ambiguous.

18. Write a short note on Chomsky hierarchy of language.

Grammar type	Grammar accepted	Language accepted	Automation
Type 0	Unrestricted Grammar(Phrase structured grammar)	Recursively Enumerable language	Turing Machine
Type 1	Context sensitive grammar	Context sensitive Language	Linear bounded automation
Type 2	Context free grammar	Context free grammar	Pushdown automation
Type 3	Regular grammar	Regular Language	Finite state automata

19. write down the steps for conversion of DFA to CFG.

Let M=(Q, $\sum$, δ, q_0, F) be a FA,

1. A Grammar G =(V, T, P, S) can be constructed where

$V=\{q_0, q_1, q_2, \ldots\ldots\ldots q_n\}$

i.e., state of DFA will be Variable in the grammar.

2. T=$\sum$, i. e. ., input alphabets of DFA will be terminals in grammar.

3. S=q_0, i.e.., start state of DFA is the start symbol in grammar.

4. Production, 'P' can be obtained as:

(a) if $\delta(q_i, a)=q_j$ then

$q_i \rightarrow aq_j$

(b) if $q_i \in F$ (i.e.., if q_i is final) then $q \rightarrow \in$

Example:

To obtain grammar to generate string consisting of at least one a.

Transitions Grammar

δ(S,a) =A Therefore, S-> aA

δ(A,a)= A Therefore, A-> aA

A is a final state Therefore, A -> ∈

∴ the grammar G =(V, T, P, S)

where V = {S, A}

T = {a}

S = S

P= {S->Aa

A-> aA|∈ }

5 Marks 2018

1.Define Context free grammar. Consider a grammar G=(V, T, P, S)

where V = {S}

T = {a, b}

S -> aS | b }.

find the language accepted by G.

CFG in defined as 4 tuples

G = (V, T, P, S) where

V = set of variables/ Non-terminals

T = Set of terminals

S = start Symbol.

P = Set of Production of the form A = α where $\alpha = (VUT)^*$

S -> As [S -> ab] S -> b S -> ab

-> aaS -> ab[∵S->b]

-> aaaS

-> aaaa……b

Therefore, S = a*b

L(G) = a*b

5 Marks 2019

1. Obtain a. CFG (Context free Grammar) for the following language

L = {$a^n b^n$ | n >= 1 }.

In this any equal number of a is followed by b. So, we must have at least ab. This is achieved by replacing ∈ by ab. So,

S-> ab | aSb

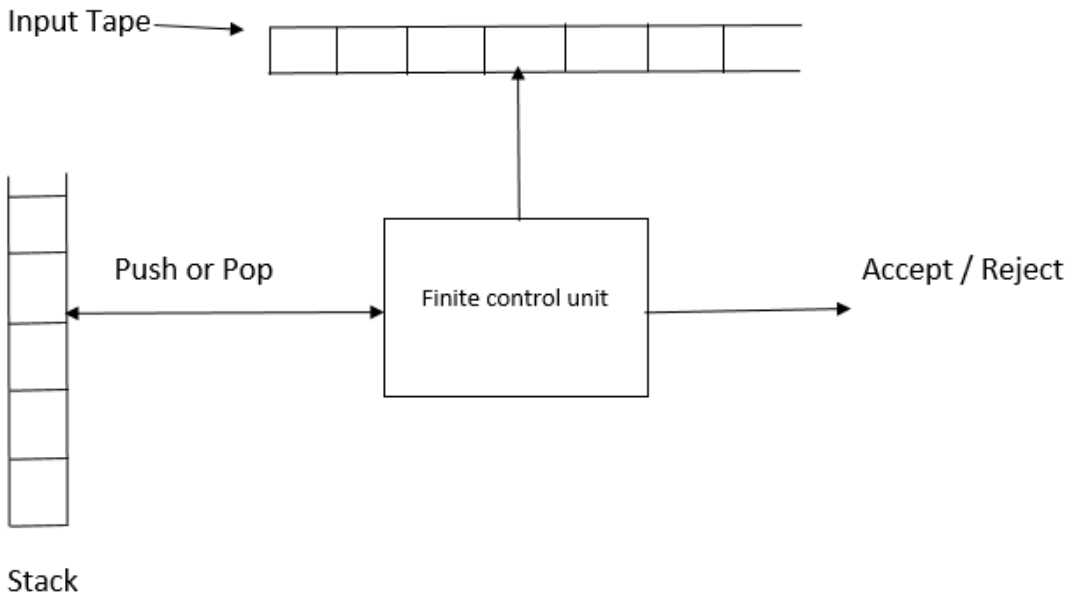
Unit 3 2017 (Section C)

23. Explain the block diagram of Pushdown automata with its components specification, language and transition table.

A DFA (or NFA) is not powerful enough to recognize many context free language.

A DFA (or NFA) has transition that it can't count and can't store the input for future reference, so need of new machine called Push Down Automation (PDA) to recognize CFL.

- PDA is a finite automaton with the addition of stack.
- A PDA has 3- Components: -
  - ➔ An input tape
  - ➔ A control unit
  - ➔ A stack with infinite Size



Specification

A Push Down automata (PDA) is a Seven table

$M=(Q, \sum, \Gamma, \delta, q_0, Z_0, F)$

where

Q- is a set of finite states

$\sum$- Set of input alphabet.

Γ - Set of stack alphabets

δ - transitions from $Q \times (\sum \cup \in) \times \Gamma$ to finite Subset of $Q \times \Gamma^*$

$q_0 \in Q$ is the start state of M

$Z_0 \in \Gamma$ is the Initial Symbol on the stack

$F \subseteq O$ is a set of final states

Transition

The transition function accepts three parameters namely a state, an input symbol & stack symbol and return a new state after changing the top of the stack.

δ (state, input symbol, stack symbol) = (next state, stack Symbol)

Example:

- The transition δ (P, a, Z) = (q, aZ)
- The transition δ (P, a, Z) =(q,$\in$)
- The transition δ (P,a, Z) = (q,r)
- The transition δ(P, $\in$, b) = (P,$\in$)

24. Transform the CFG info GNF. -

S -> AB

A -> BS | 1

B -> SA | 0

Let S= A, A= A_2, B = A_3 and the resulting grammar is

A_2 -> A_3 A_1 | 1

A_3 -> A_1 , A_2 |0

1st two productions are of the form

A_i -> A_j α for i < j

So, we consider A_3 production

Consider A_3 - Production:

Substituting for A_1 in A_3 production we get,

A_3 -> A_1 A_2 |0 = (A_2 A_3) A_2 | 0

Now again replacing the first A_2 in A_3 production we get,

A_3 -> A_2 A_3 A_2 | 0 = (A_3 A_1 |1) A_3 A_2 | 0

=A_3 A_1 A_3 A_2 | 1 A_3 A_2 | 0

we get the resulting A_3 production as

A_3 -> A_3 A_1, A_3 A_2 | 1 A_3 A_2 | 0

which is having left recursion. After eliminating left recursion, we get,

A_3 -> 1 A_3 A_2 |0|1A_3 A_2 Z | 0Z

Z -> A_1 A_3 A_2 | A_1 A_3 A_2 Z

Now, all A_3 production are in GNF.

Consider A_2 –production:

Since all A_3 -production is in GNF, Substituting A_2 -production we get,

A_2 -> (0 1 A_3 A_2 |0|1A_3 A_2 Z | 0Z) A_1 | 1

= 1 A_3 A_2 A_1|0 A_1|1A_3 A_2 Z A_1 | 0Z A_1 | 1

which is in GNF.

Now, all A_2- production are in GNF.

consider A_1 - productions:

Since all A_2 production are in GNF, Substituting A_2 production in A_1 - production we get,

A_1 -> A_2A_3 = (1 A_3 A_2 A_1 |0 A_1 | 1A_3 A_2 Z A_1| 0 Z A_1 | 1)A_3

Now, A_1 -production are also in GNF.

Consider Z – production:

Since A_1, is in GNF, Substitute A_1 - production in Z production, we get Z-Production in GNF below

Z -> A_1 A_3 A_2 | A_1 A_3 A_2 Z

-> (1 A_3 A_2 A_1 A_3|0 A_1 A_3 | 1A_3 A_2 Z A_1 A_3 | 0 Z A_1 A_3 | 1 A_3) A_3 A_2 |

(1 A_3 A_2 A_1 A_3|0 A_1 A_3 | 1A_3 A_2 Z A_1 A_3 | 0 Z A_1 A_3 | 1 A_3) A_3 A_2 Z

which can be written as

Z -> 1 A_3 A_2 A_1 A_3 A_3 A_2 | 0 A_1 A_3 A_3 A_2 | A_3 A_2 Z A_1 A_3 A_3 A_2 | 0 Z A_1 A_3A_3

A_2 | 1 A_3 A_3 A_2

Z -> 1 A_3 A_2 A_1 A_3 A_3 A_2 Z| 0 A_1 A_3 A_3 A_2 Z | A_3 A_2 Z A_1 A_3 A_3 A_2 Z| 0 Z A_1 A_3A_3

A_2 Z| 1 A_3 A_3 A_2 Z

Now, Since all productions are in GNF, the resulting grammar is also in GNF.

So, final grammar obtained in GNF is G=(V, T, P,S)

where V = { A_1 A_2 A_3 Z }

T = { 0, 1}

P = { A_1 -> 1 A_3 A_2 A_1 |0 A_1 A_3| 1 A_3 A_2 Z A_1 A_3| 0 Z A_1 A_3 | 1 A_3

A_2 -> 1 A_3 A_2 A_1|0 A_1|1A_3 A_2 Z A_1| 0ZA_1 | 1

A_3 -> 1 A_3 A_2 |0|1A_3 A_2 Z | 0Z

Z -> 1 A_3 A_2 A_1 A_3 A_3 A_2 | 0 A_1 A_3 A_3 A_2 Z| 1 A_3 A_2 Z A_1 A_3 A_3 A_2 | 0 Z A_1 $A_3$$A_3$ A_2 | 1 A_3 A_3 A_2

A_1 is the start Symbol

25 (a) Explain Post's correspondence Problem

Definition :

Given two Sequence of n strings on Same alphabet $\sum$ say

A = W_1,W_2,…W_n

B = V_1,V_2,…V_n

is says that there exists a post Correspondence solution for pair (A,B) if there is a non-empty sequence of integer i, j ...k such that W_i,W_j,…W_k = V_i,V_j,…V_k

The PCP in to device an algorithm that will tell us, for any (A,B)

whether or not there exists a solution

Example:

$\frac{B}{CA}$ $\frac{A}{AB}$ $\frac{CA}{A}$ $\frac{ABC}{C}$

$\frac{A}{AB}$ $\frac{B}{CA}$ $\frac{CA}{A}$ $\frac{A}{AB}$ $\frac{ABC}{C}$

b. Explain intersection and homomorphism property of Regular languages.

For L_1 and L_2 are Regular then it is closed under Intersection.

L_1 L_2 are regular language.

let $L = L_1 \cap L_2$

By applying Demorgans's theorem

$L_1 \cap L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$ [$\because \overline{\overline{L_1} \cap \overline{L_2}} = \overline{\overline{L_1} \cap \overline{L_2}}$]

$\overline{L_1}$, $\overline{L_2}$ is regular

$\overline{L_1} \cup \overline{L_2}$ is regular

$\overline{\overline{L_1} \cup \overline{L_2}}$ is regular

$L_1 \cap L_2$ is regular

so, regular language is closed under intersection

<u>closure under Homomorphism</u>:

let $\sum$ & Γ be the set of alphabets

The homomorphic function h : $\sum$ -> Γ * is called homomorphism.

$w = a_1 a_2 a_3 \ldots\ldots\ldots a_n$

$h(w) = h(a_1)\, h(a_2)\, h(a_3) \ldots\ldots\ldots h(a_n)$

If L is made of alphabet from $\sum$ then $h(L) = \{L(w) \mid w \in L\}$ is called homomorphic image.

19 Eliminate useless symbols from the following grammar.

S -> aAa

A -> Sb

A -> DaA

C -> abb

C -> DD

E -> ac

D -> aDa

V={S, A, C, D, E}

Identify non generating symbols

=V - set od generated variable

={ S, A, C, D } = { S, A, C, E}

={ D }

Remove all production having D Variable.

The resulting productions are :

S -> aAa

A -> Sb

A -> bCc

C=> abb

E => ac

Step 2: Identify unreachable symbol:

Unreachable symbol = (V∪T) – Reachable symbol

= { (S, A, C, E, a, b) – { S, A, C}}

= { E, a, b }

Remove all production having {E, a, b}

S -> aAa

A -> Sb | bCc

24.

a) Obtain a grammar to generate sling string consisting of any number of a's & b's with at least one or at least b.

Transition	Grammer
δ(S, a) = A	S -> aA
δ(S, b) = A	S -> bA
δ(A, a) = A	A -> aA
δ(A, b) = A	A -> bA

A is the final state [A -> E]

The equivalent grammar G=(V,T, P,S) where,

V = {S, A}

T = {a, b}

S = {S}

P = {S -> aA | bA A-> aA | bA | ∈ }

b) for the following production

S⟶AB

A -> aaA | ∈

B -> Bb | ∈

write the left most & Right most derivation for the string aab.

Leftmost derivation	Rightmost derivation
S -> AB	S -> AB
-> aaAB	-> ABb
-> aaB	-> Ab
-> aaab	-> aaAb
	-> aab

c) For the grammar G with production rules

E -> E + E

E -> E * E

E -> id

where V = {E}, T={id}, S={E}. Obtain the rightmost derivation and parse tree for the string w=id + id * id

Rightmost Derivation

E -> E + E

E -> E + E * E

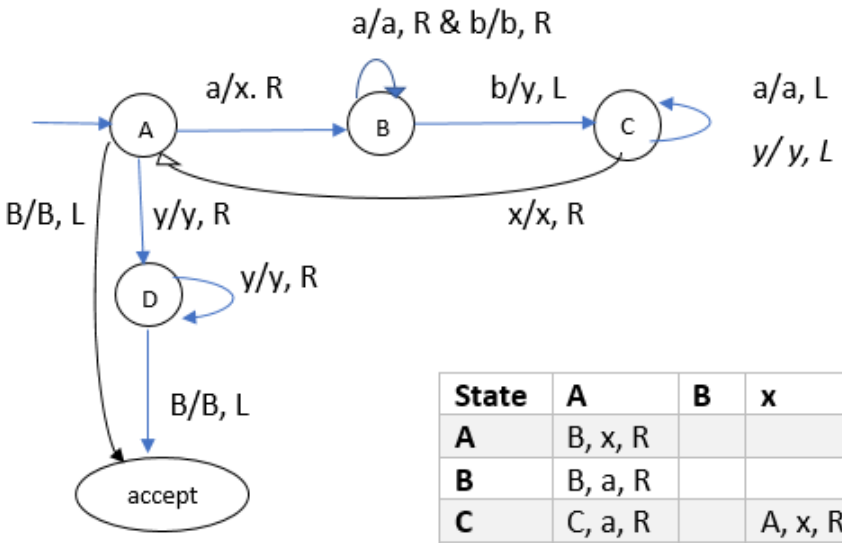
E -> E + E * id

E -> E + id * id

E -> id + id * id

Unit 5 2018

25. Obtain a Turing machine to accept the language L ={$a^n b^n$ | n>=1}



State	A	B	x	y	B
A	B, x, R			D, y, R	
B	B, a, R			B, y, R	
C	C, a, R		A, x, R	D, y, L	
D			D, y, R		
E	-	-			

2019 Unit 4

19. Eliminate the unit production from the grammar

S -> AA

A-> a

B -> c

B -> b

C -> D

D -> E

E -> a

There are 2-unit production:

C -> D and

D -> E

C -> D [not able to remove this unit production therefore no production is of form C -> α

D -> E [yes, we can remove this unit production because we have

E -> a

So, we can rewrite the production rules:

S -> AB

A -> a

B -> c/b

C -> D

D -> a

E -> a

Now for C -> D

[Now, we can remove this unit products because we have D -> a so, it can be rewrite the production rule

S -> AB

A -> a

B -> c/b

C -> a

D -> a

E -> a

Hence now in the resultant production rule we don't have any unit production.

20. Show that the following grammar is ambiguous.

E -> E + E
E -> E - E

E -> E * E

E -> E / E

E -> |E|

E -> id

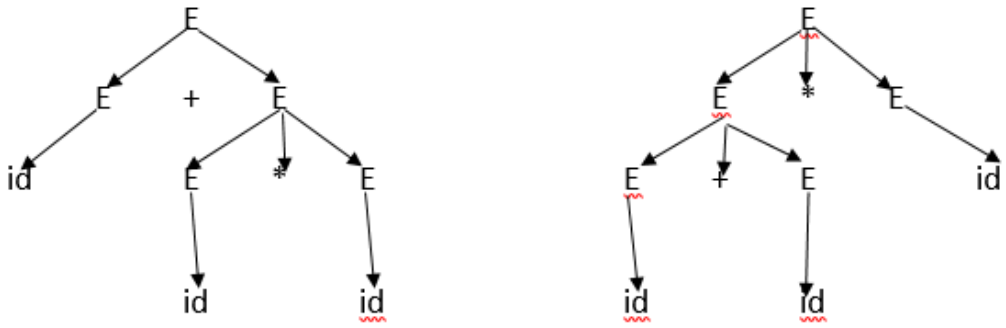
By using left most derivation:

E -> E + E

E -> E * E

-> id + E	-> E + E * E
-> id + E * E	-> id + E * E
-> id + id * E	-> id + id * E
-> id + id * id	-> id + id * id

Parse tree:



for the string id + id * id, we applied leftmost derivation & get two different parse tree so, the grammar is ambiguous.

Unit 3 2019

23. Construct a PDA to accept the language L(M)= {ww^R | w ∈ (a+b)* where w^R is the reverse of w by final state acceptance.

Given that L(M) = { ww^R }

if w= abb then reverse of w is denoted by w^R = bba so, the language L will be

ww^R = abbbba

Step 1:

Input Symbol $\sum$= {a,b} Let q_0 be the initial state and Z_0 to be initial symbol on the stack. in state q_0 with top of stack, Push the input symbols in stack 4 remain in q_0.

$\delta (q_0, a, Z_0) = (q_0, a, Z_0)$

$\delta (q_0, b, Z_0) = (q_0, b, Z_0)$

Now, in state q_0, Push input symbol a or b to the slack, The transition are

$\delta (q_0, a, a) = (q_0, a, a)$

$\delta (q_0, a, b) = (q_0, a, b)$

$\delta (q_0, b, a) = (q_0, b, a)$

$\delta (q_0, b, b) = (q_0, b, b)$

step 2:

once we reach the midpoint & if next Symbol is same. then pop the symbol from stack & move next to stage, q_1.

$\delta (q_0, a, a) = (q_1, \in)$

$\delta (q_0, b, b) = (q_1, \in)$

in the slate q_1, repeat the step 2 until we find empty input. The transitions are

$\delta (q_1, a, a) = (q_0, \in)$

$\delta (q_1, b, b) = (q_1, \in)$

step 4:

Finally, in state q_1 if a string is Palindrome, then there will be scanned & the stack should be empty.

$\delta (q_0, \in, Z_0) = (q_2, Z_0)$

Step 5:

The PDA, M to accept the language L(M)= $\{ww^R \mid w \in (a+b)^*\}$ is given by M=(Q, $\sum$, Γ, δ, q_0, Z_0, F)

where

Q- { q_0 , q_1, q_2 }

$\sum$- { a, b }

Γ - { a, b, Z_0 }

Z_0 – { Z_0 }

F – {q_2}

δ = $\delta (q_0, a, Z_0) = (q_0, a, Z_0)$

$\delta (q_0, b, Z_0) = (q_0, b, Z_0)$

$\delta (q_0, a, a) = (q_0, a, a)$

$\delta (q_0, a, b) = (q_0, a, b)$

$\delta (q_0, b, a) = (q_0, b, a)$

$\delta (q_0, b, b) = (q_0, b, b)$

$\delta (q_0, a, a) = (q_1, \in)$

$\delta (q_0, b, b) = (q_1, \in)$

$\delta (q_1, a, a) = (q_1, \in)$

$\delta (q_1, b, b) = (q_1, \in)$

δ (q_0, $\in$, Z_0) = (q_2, Z_0)

a, Z_0 / a Z_0

b, Z_0 / b Z_0

a, a / aa

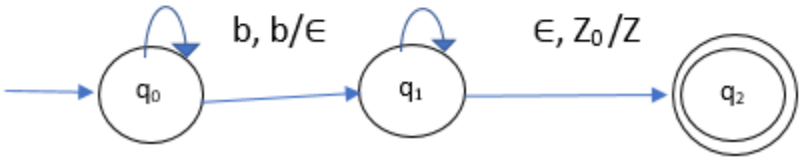
a, b / ab

b, a / ba

b, b / bb

a, a/$\in$

b, b/$\in$



24. Find the language accepted by CFG.

G= (V, T, P, S) where

V = {S}

T = {a, b}

S -> S

P -> {S -> aS | b }.

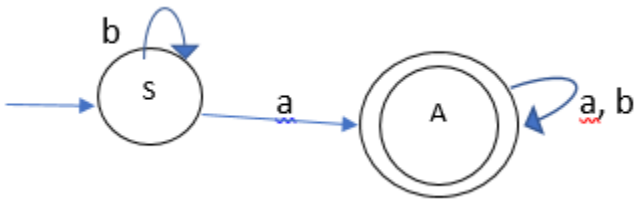
S -> aS
-> aaS
-> aaaS
-> aaa…….aS
-> aaa……. ab

S -> b

S -> aS
-> ab

i.e.., S -> a*b therefore L(G) = a*b

b) Obtain a grammar to generate string S = {a,b} having atleast one a



Transition	Grammar
δ(S, a) = A	S -> aA
δ(S, b) = S	S -> bS
δ(A, a) = A	A -> aA
δ(A, b) = A	A -> bA

A is the final state [A -> ∈]

The equivalent grammar G= (V, T, P, S) where,

V = {S, A}

T = {a, b}

S = {S}

P = {S -> aA | bAS A-> aA | bA | ∈ }

c) obtain a CFG for the language L = {wcw^R (W∈ {a, b}*}

The string that can be generated From this language are C, aca, bcb, abcba…..

S -> c

S -> aSa | bSb

The CFG, G = { V, T, P, S }

V = {S}

T = {a, b, c}

S = {S}

P = {S -> aSa | bSb| c }

Unit 5 2019

25. Obtain a Turing machine to accept the language L = {$a^n b^n$ | n>= 1}

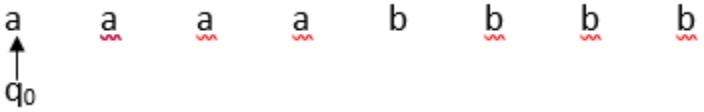
Given that,

TM should have n number of a's followed by 'n' number of b's.

Example,

aaaabbbb

Let, q_0 be the start of TM & read-write head prints to the 1st symbol of the string to be scanned.

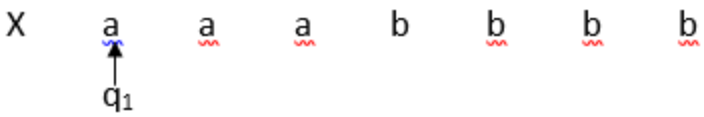


step 1:

In state q_0, replace a by X, change the state to q_1, & move pointer towards right.

$\delta(q_0, 0) = (q_1, X, R)$

The resulting Configuration is



Step 2:

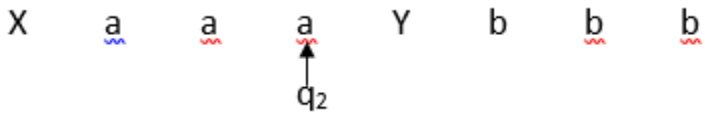
In state q_1, find the left most b and change the state to q_2. If we find any a's or Y's while moving right,

i.e.., $\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, Y) = (q_1, Y, R)$

$\delta(q_1, 1) = (q_1, Y, L)$

The resulting Configuration is



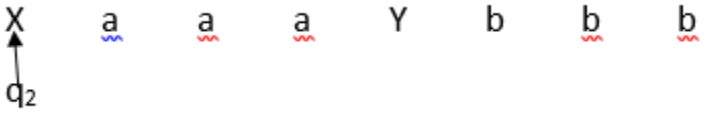
Step 3:

The read/write head has to move towards left to obtain left most a

$\delta(q_2, Y) = (q_2, Y, L)$

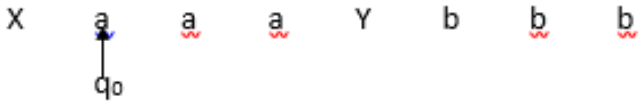
$\delta(q_2, a) = (q_1, a, L)$

The resulting configuration in



Step 4:

To get leftmost a, move pointer to right without changing X with other symbol, change state to q_0



Step 5:

Repeat step 1 to Step to get configuration :

Step 6:

In the state q_0, if the scanned Symbol is Y, it means there are no more a's. To check there are no more b's move the pointer towards night by changing the state to q_3

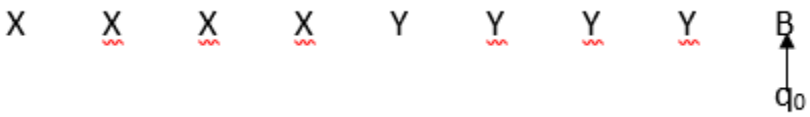
$$\delta(q_0, Y) = (q_3, Y, R)$$

Step 7:

In state, a there are only Y's and no more b's

$$\delta(q_3, Y) = (q_3, Y, R)$$

The resulting configuration:



Step 8:

Now, the string ends with infinite number blanks, change the state to q_4, which is a final state.

$\delta(q_3, B) = (q_4, B, R)$

Therefore, The TM to accept

$M=(Q, \sum, \Gamma, \delta, q_0, B, F)$

where

Q- $\{q_0, q_1, q_2, q_3, q_4,\}$

$\sum$- {0, 1}

Γ – {0, 1, X, Y, B}

$q_0 \in Q$ is the start state of Machine

$B \in \Gamma$ is the Blank Symbol

$F = \{q_4\}$

is the salary of machine BE N is the blank symbol?

$\delta(q_0, a) = (q_1, X, R)$

$\delta(q_1, a) = (q_1, 0, R)$

$\delta(q_1, Y) = (q_1, Y, R)$

$\delta(q_1, b) = (q_2, Y, L)$

$\delta(q_2, Y) = (q_2, Y, L)$

$\delta(q_2, a) = (q_2, a, L)$

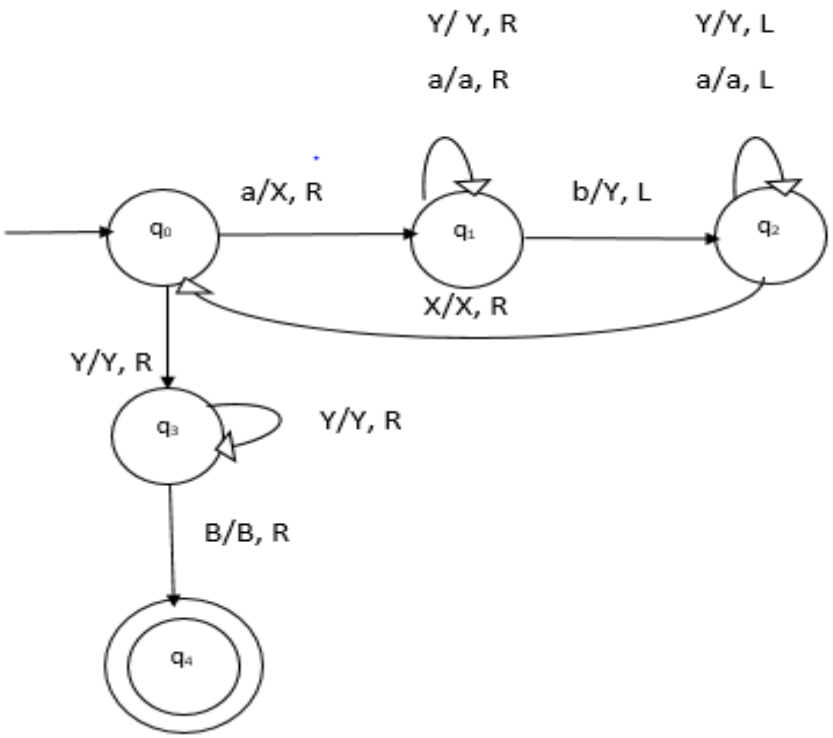
$\delta(q_2, X) = (q_0, X, R)$

$\delta(q_0, Y) = (q_3, Y, R)$

$\delta(q_3, Y) = (q_3, Y, R)$

$\delta(q_3, B) = (q_4, Y, R)$

Turing Machine diagram,



SA – 918

VI Semester B.C.A. Degree Examination, April/May 2015
(Y2K8 Scheme)
COMPUTER SCIENCE
BCA – 602 : System Programming
(100 mark – 2013-14 and Onwards/90 mark – Prior to 2013-14)

Time : 3 Hours Max. Marks : 90/100

Instructions : *1) Answer **all** questions.*
*2) Section – **D** is applicable for **100** marks student only*
(F+R – 100 : 2013 -14 and onwards R – 90 : Prior to 2013-14)

SECTION – A

I. Answer **any ten** questions. **(10×2 = 20)**

1) Define processor. List its types.
2) Differentiate between procedure and processor.
3) Explain program status word.
4) List different types of operands in an instruction format.
5) Mention the steps in designing of an Assembler.
6) What are the functions of an assembler ?
7) Expand MDI and MDLC.
8) Differentiate between macro and subroutine.
9) List four types of cards used in direct linking loader.
10) What is Binder ? What are the classes of Binders ?
11) Define Machine Dependent Optimization.
12) List the data bases of Lexical phase.

SECTION – B

II. Answer **any five** questions. **(5×5 = 25)**

13) Explain general machine structure with a neat diagram.
14) What is Pseudo-Op ? Explain its types.

P.T.O.

15) Explain databases used in Pass 1 and Pass 2 of Assembler.

16) Illustrate Binary Search with an example.

17) Explain macro facility with macro instruction argument with an example.

18) Explain Absolute Loader.

19) What are the functions of analysis and synthesis phases of compiler ?

20) Write a note on assembly phase of compiler.

SECTION – C

III. Answer **any three** questions. **(3×15=45)**

21) a) Explain different types of instruction formats with example. **7**

b) Explain address modification using instruction as data. **8**

22) a) Draw the detailed Pass 2 flow chart of an Assembler. **7**

b) List different types of sorting. Explain Interchange sort. **8**

23) a) Write an algorithm for pass 1 processing macro definitions. **7**

b) Explain implementation of macro processor within assembler. **8**

24) a) Write a note on direct linking loader. **7**

b) Draw a flow chart for pass 2 of loader. **8**

25) a) Explain Intermediate form of compiler. **7**

b) Explain code generation phase of compiler. **8**

SECTION – D

IV. Answer **any one**. **(1×10=10)**

26) a) Write a note on evolution of system software. **5**

b) What is an assembler ? Explain any four assembler directives. **5**

27) Explain the structure of a compiler with a neat diagram. **10**

MS – 583

VI Semester B.C.A. Examination, May 2016
(Y2K8 Scheme) (F + R)
Computer Science
BCA – 602 – SYSTEM PROGRAMMING
(100 marks – 2013-14 and Onwards
90 marks – Prior to 2013-14

Time : 3 Hours Max. Marks : 90/100

Instructions : *1) Answer **all** questions .*
*2) Section **D** is applicable for students from **2013-14** & Onwards.*

SECTION – A

I. Answer **any 10** questions. **Each** question carries **two** marks. **(10×2=20)**

1) List components of a system software.

2) Differentiate between processor and I/O channel.

3) Explain PSW.

4) Explain the advantages of using a base register.

5) What is LTORG ?

6) List all the variable tables and fixed tables used by an assembler.

7) What is ALA ? Explain its format.

8) Define AIF and AGO.

9) List the functions of a loader.

10) Mention the four types of cards used in a direct linking loader.

11) What is intermediate form ?

12) List the three tasks of lexical phase.

P.T.O.

SECTION – B

II. Answer **any five** questions. **Each** question carries **five** marks. **(5×5=25)**

13) Explain different data formats used in IBM 360 with an example.

14) Explain Long-way-no-looping.

15) Draw an overview flowchart for Pass I of an assembler.

16) Explain radix sort with a suitable example.

17) Explain macro definition with arguments with an example.

18) Explain compile-and-go loader with a diagram.

19) Explain the databases used in lexical analysis phase of a compiler.

20) Explain machine dependent optimisation.

SECTION – C

III. Answer **any three** questions. **Each** question carries **fifteen** marks. **(3×15=45)**

21) a) Draw the general machine structure of IBM 360 and explain. **8**

b) Explain various instruction formats used in IBM 360. **7**

22) a) Draw the detailed Pass 2 flowchart of an assembler. **8**

b) Explain databases used by Pass I and Pass II of an assembler. **7**

23) a) Explain the terms macro definition, macro call and macro expansion with an example. **7**

b) Explain with a flowchart pass 2 of a macroprocessor. **8**

24) a) Explain the design of an absolute loader. 7

b) Explain the databases used by Pass I and Pass II of a direct linking loader. 8

25) a) Explain structure of a compiler with a diagram. 10

b) Explain syntax phase of a compiler. 5

SECTION – D

Answer **any one** question. Question carries **ten** marks. **(1×10=10)**

26) a) Sort the following numbers using bubble sort technique :

55, 53, 45, 48, 39. 5

b) Explain any 5 pseudo ops. used in an assembly language program. 5

27) a) Explain dynamic loading. 5

b) List the databases used by Pass I and Pass II of a macroprocessor. 5

US – 645

VI Semester B.C.A. Examination, May 2017
(2016-17 and Onwards) (CBCS)
COMPUTER SCIENCE
BCA 602 : System Programming

Time : 3 Hours Max. Marks : 100

Instruction : *Answer* ***all*** *Sections.*

SECTION – A

I. Answer **any ten** questions, **each** question carries **two** marks : **(2×10=20)**

1) Define compiler, assembler.
2) What are the functions of a Loader ?
3) Explain PSW.
4) What is Instruction Interpreter ?
5) Write the format of POT.
6) What is a symbol table ? Give its format.
7) Differentiate between a macro and subroutine.
8) What is an argument list array ?
9) What are overlays ?
10) What is dynamic loading ?
11) What are the three classes of uniform symbols ?
12) Define local and global optimization.

SECTION – B

II. Answer **any five** questions, **each** question carries **five** marks. **(5×5=25)**

13) Explain open subroutine and closed subroutine with an example.
14) Explain different instruction formats of IBM 360/370 machine.
15) Explain address modification using instruction as data.
16) Explain shell sort with an example.

P.T.O.

17) Explain pass-2 overview of an assembler with flow-chart.

18) Explain macro definitions with an example.

19) Describe four types of cards used in direct linking loader.

20) Explain intermediate phase with an example.

SECTION – C

III. Answer **any three** questions, **each** question carries **fifteen** marks. **(3×15=45)**

21) a) Explain the general machine structure of IBM 360/370 with a neat diagram. **7**

b) Draw the detailed PASS-1 flow-chart of an assembler. **8**

22) a) Explain databases used in PASS-1 and PASS-2 of assembler. **8**

b) Explain different data formats used in IBM 360/370 with an example. **7**

23) a) Explain simple one pass macro processor. **10**

b) Explain conditional macro expansion. **5**

24) a) Explain design of absolute loader with a neat diagram. **8**

b) Explain direct-linking loaders. **7**

25) a) Explain the passes of compiler with neat diagram. **10**

b) Discuss briefly about lexical phase of compiler. **5**

SECTION – D

IV. Answer **any one** question, **each** question carries **ten** marks. **(1×10=10)**

26) With a neat diagram explain the structure of compiler. **10**

27) Write short note on :

a) Relocating loaders. **5**

b) Draw the micro flow-chart of ADD instruction. **5**

SM – 624

VI Semester B.C.A. Examination, May/June 2018
(CBCS) (F+R) (2016-17 and Onwards)
COMPUTER SCIENCE
BCA 602 : System Programming

Time : 3 Hours Max. Marks : 100

***Instruction** : Answer **all** Sections.*

SECTION – A

I. Answer **any ten** questions. **Each** question carries **two** marks. **(10×2=20)**

1) What is system software ?
2) What is location counter ? What is its purpose ?
3) List any two advantages of assembly language.
4) What is Declaration Statement ? Give example.
5) Mention any two disadvantages of Radix Sort.
6) What is Macro call ?
7) Define Macro definition table.
8) Write the four basic task that can be performed by macro-instruction processor.
9) What are the functions of loader ?
10) Define Relocation factor.
11) What is intermediate form ?
12) What is a token ? Give example.

SECTION – B

II. Answer **any five** questions. **Each** question carries **five** marks. **(5×5=25)**

13) Explain the general machine structures with neat diagram.
14) What is sorting ? Explain briefly about Bubble sort.
15) Explain databases used in Pass 1 and Pass 2 assemblers.
16) Explain the features of Macro facility in detail.
17) Explain macro instructions defining macros.
18) Explain compile-and-go loader with a neat diagram.
19) Define binder. What are the classes of binders ? Explain.
20) What are the functions of analysis and synthesis phases of compiler ?

P.T.O.

SECTION – C

III. Answer **any three** questions. **Each** question carries **fifteen** marks. **(3×15=45)**

21) a) Explain various instruction formats used in IBM 360. **8**

b) Explain the use of literals in assembly language programs using example. **7**

22) a) Draw the detailed pass 2 flowchart of an assembler. **8**

b) What is an assembler directive ? Explain any five assembler directives with an example. **7**

23) a) Give the database specifications for pass 1 and pass 2 of macro processor. **8**

b) Explain the four basic tasks of macroprocessor. **7**

24) a) Explain design of absolute loader with a neat diagram. **8**

b) Explain the overlay structures for linking. **7**

25) a) Explain structure of compiler with a diagram. **8**

b) Explain identifier table for the phases of compiler. **7**

SECTION – D

IV. Answer **any one** question. **Each** question carries **ten** marks. **(1×10=10)**

26) a) Differentiate between Pseudo-op and machine-op with example. **5**

b) Draw the micro-flow chart for ADD instruction. **5**

27) a) Explain Relocatable, non-relocatable and self relocatable programs. **5**

b) Explain the use of EXTERN and ENTRY statements. **5**

100409

No. of Printed Pages : **2**

GS-643

VI Semester B.C.A. Examination, May/June 2019
(CBCS - F+R) (2016-17 & onwards)

COMPUTER SCIENCE

BCA 602 : System Programming

Time : 3 Hours Max. Marks : 100

Instruction : *Answer* ***all*** *sections.*

SECTION - A

I. Answer **any ten** questions. Each question carries **two** marks. **10x2=20**

1. Define System Programming.
2. Mention the functions of loader.
3. Explain BCT and LTDRG pseudo-ops.
4. Define DC and DS.
5. What is Macro ? Write down its syntax.
6. Define open and closed subroutine.
7. Define PSW.
8. Define local and global Optimization.
9. What is MDT and MNT ?
10. What is sorting ?
11. Differentiate between compiler and assembler.
12. Mention any 4 components of SP.

SECTION - B

II. Answer **any five** questions, Each question carries **five** marks. **5x5=25**

13. Explain Data formats of IBM 360/370 machine.
14. Explain Long-Way-No-Looping with example.
15. Sort following array using Radix exchange Sort
    21, 14, 19, 11, 7, 12
16. Explain Macro definition with arguments with an example.
17. Write specification of databases used in pass 1 and pass 2 Assembler.
18. Explain General Loader Scheme.
19. Explain Machine independent optimisation.
20. Explain databases used in lexical analysis phase of a compiler.

P.T.O.

SECTION - C

III. Answer **any three** questions. Each question carries **fifteen** marks. **3x15=45**

21. (a) Explain General Machine Structure of IBM 360/370 with a neat diagram.
(b) Explain instruction formats of IBM 360/370 with syntax and example.

22. (a) Draw detailed pass 2 flow chart of an assembler.
(b) Give format of all five tables used in assembler.

23. (a) Give specification of databases used in pass 1 and pass 2 of Macroprocessor.
(b) Explain Macro instructions defining Macros with an example.

24. (a) Explain design of an absolute loader with a neat diagram.
(b) Give specification of databases used in pass 1 and pass 2 of Direct Linking Loader.

25. (a) Explain different phases of a Compiler with a neat diagram.
(b) Explain syntax Phase of a Compiler.

SECTION - D

IV. Answer **any one** question. Each question carries **ten** marks. **1x10=10**

26. (a) Explain the terms macro definition, macro call, macro expansion with syntax and example. **5**
(b) Draw Micro-flow chart for ADD instruction. **5**

27. (a) Explain different types of cards used in Direct Linking loader. **5**
(b) Write a note on compile and go loader. **5**

- o 0 o -

SOUNDARYA EDUCATIONAL TRUST (R)

SOUNDARYA INSTITUTE OF MANAGEMENT & SCIENCE

Soundarya Nagar, Sidedahalli, Hessaraghatta Main Road, Bangalore – 73

Department of Computer Science

SOLUTION BANK

BCA602T – System Programming

2 MARKS QUESTIONS AND ANSWERS

UNIT - 1

1. List components of a system software. (Repeated twice)

Components of system software are:

- Assemblers.
- Compilers.
- Loaders.
- Macro processor.
- Operating System.

2. Differentiate between processor and I/O channel.

Processor	I/O Channel
A processor is a device that performs a sequence of operations specified by instructions in memory.	I/O channels need processor in order to execute I/O instructions that are stored in memory.
It is the internal components of the system.	It is the external components of the system.
Through I/O channels processor retrieves and stores the data.	I/O channels will act as inter mediator between I/O devices and the CPU.

3. Explain PSW. (Repeated four times)

PSW stands for Program Status Word contains the information required for proper execution of a given program. It contains the value of the location, protection information and interrupt status. The size of a PSW is 8 bytes.

4. Explain the advantages of using a base register.

Advantages of base register:

1. It helps in the process of relocation of a program.
2. Efficient addressing of core.
3. Saves 8 bits per address reference.

5. What is LTORG?

LTORG is a pseudo-op which tells the assembler to place the encountered literals at an earlier location.

6. Define complier, assembler.

Compiler: Compiler is a program that accepts a source program in high level language and produces its corresponding object program.

Assembler: Assembler is a program that translates the assembly language program (Source Code) into machine language program (Object Code).

7. What is instruction interpreter?

Interpreter is a translator used for translating programs in high level language into machine language. It interrupts the instruction line by line.

8. What is system software?

System Software is computer software designed to operate the computer hardware and provides and maintains a platform for running application software.

9. What is location counter? What is its purpose?

Location counter is also known as Program counter(PC) or Instruction Counter is a hardware memory device which holds the location of the current instructions being executed.

10. List any two advantages of assembly language.

Advantages of Assembly Language:

1. It used mnemonic code.
2. Address are symbolic not absolute.
3. Reading is easier.
4. Introduction of data to program is easier.
5. It is not required to keep track of memory location.

11. Define system programming.

System programming involves designing and writing computer programs that allow the computer hardware to interface with the **programmer** and the user, leading to the effective execution of application software on the computer **system**.

12. Define open and closed subroutine.

Open Subroutine: An open subroutine is one whose code will be inserted at the point of function call within the main definition.

Closed Subroutine: A closed subroutine will be stored outside the main routine and there is a transfer in control to the subroutine for processing it.

13. Explain BCT and LTORG pseudo-ops (Refer. 5).

BCT: Indicates branch on count. It is a RX type instruction, whose size is 4 bytes.

14. Define DC and DS.

DC: DD stands for Define Constant, is a declarative pseudo-op used to create a memory area to hold a constant value. That is it constructs memory words containing constants.

DS: DS stands for Define Storage, is a declarative pseudo-op that reserves some storage for the data and gives them a name.

UNIT - 2

1. **List all the variable tables and fixed tables used by an assembler.**

   Variable Tables: Literal Table(LT), Symbol Table(ST) and Base Table(BT).
   Fixed Tables: Machine Op Table(MOT) and Pseudo Op Table(POT).

2. **Write the format of POT.**

8- bytes per entry

Pseudo-op(5-bytes) Character	Address of routine to process pseudo-op (3-bytes = 24 bits address)
"DROPb"	P1DROP
"ENDbb"	P1END
"EQUbb"	P1EQU
"START"	P1START
"USING"	P1USING

3. **What is a symbol table? Give its format.**

   Symbol Table: It is a variable table that contains labels and its value. ST for Pass1 and Pass2 are the same and contains symbol, value, length and relocation fields. Size of the table is 14 bytes per entry.

14 - bytes per entry

Symbol (8-bytes) character	Value (4-bytes) Hexadecimal	Length (1-byte) Hexadecimal	Relocation(1-byte) character
"JHONbbbb"	0000	01	"R"
"FOURbbbb"	000C	04	"R"
"FIVEbbbb"	0010	04	"R"
"TEMPbbbb"	0014	04	"R"

4. **What is Declaration Statement? Give example.**

   In programming, a declaration is a statement describing an identifier, such as the name of a variable or a function

5. **Mention any two disadvantages of Radix Sort.**

   Disadvantages:
   1. Radix Sort that can make it less preferable than other sorts.
   2. The speed of Radix Sort largely depends on the inner basic operations.

6. **What is sorting?**

   Sorting is the process of arranging items of the table in some sequence, either in ascending or descending order.

UNIT - 3

1. **What is ALA?(Repeated twice) Explain its format.**

ALA is a Argument List Array maintains the details about the parameters. ALA is used during pass1 and pass2, but the functions are reverse in both the process.

Argument List Array	**8 bytes per entry**
Index	**Argument**
0	"LOOP1bbb"
1	"DATA1bbb"
2	"DATA2bbb"
3	"DATA3bbb"

2. **Define AIF and AGO.**

AIF:AIF is a conditional branch pseudo-op. The format is as follows: AIF<expression> <sequencing symbol>. It does not appear in the expanded source code.

AGO: AGO is an unconditional branch pseudo-op. The format is as follows: AGO<sequencing symbol>. It does not appear in the expanded source code.

3. **Differentiate between a macro and subroutine.**

Macro	Subroutine
Macro can be called only in the program it is defined.	Subroutine can be called from other programs also.
Macro can have maximum 9 parameters.	Can have any number of parameters.
Macro can be called only after its definition.	This is not true for Subroutine.
A macro is defined inside: DEFINE … …. END-OF-DEFINITION.	Subroutine is defined inside: FORM ….. ….. ENDFORM.
Macro is used when same thing is to be done in a program a number of times.	Subroutine is used for modularization.

4. **What is Macro call?**

A **macro call** consists of a name optionally followed by an actual parameter list. The number of parameters in the actual parameter list must be the same as the number of formal parameters specified in the definition of the **macro**.

5. **Define Macro definition table.(MDT)**

MDT is used to store the body of the macro definitions. It contains text lines. The size of macro definition table is 80 bytes per entry. Every line in the macro definition, except MACRO is stored in the MDT.

6. **Write the four-basic task that can be performed by macro-instruction processor.**

Four basic task that can be performed by macro-instruction processor:

1. Define macro.
2. Include macro name in the source program.

3. Expand Macro definition.
4. Compile or Assemble the Mcro.

7. What is Macro? Write down its syntax.

Macros allow a programmer to define pseudo operations, typically operations that are generally desirable, are not implemented as part of the processor instruction, and can be implemented as a sequence of instructions. Each use of a macro generates new program instructions, the macro has the effect of automating writing of the program.

Syntax:

MACRO	Start definition
[ ]	Macro name
_____ _____ _____ }	Sequence to be abbreviated
MEND	End of definition

8. What is MDT (Ref. 5) and MNT?

MNT stands for Macro Name Table is used to store the names of the defined macros. It is similar to machine op-table(MOT) and POT of the assembler. Each MNT entry consists of a character string whose size is 8 bytes and a pointer to the entry in the MDT that corresponds to the beginning of the macro definition. The size of he MDT is 4 bytes. Therefore, the size of the MNT is 12bytes per entry.

UNIT - 4

1. List the functions of a loader.(Repeated four times

Functions of Loader:

I. Allocation: Allocate space in memory for the programs.
II. Linking: Resolve symbolic references between object decks.
III. Relocation: Adjust all address dependent locations, such as address constants, to correspond to the allocated space.
IV. Loading: physically place the machine instructions and data into memory.

2. Mention the four types of cards used in a direct linking loader.

Four types of cards used in a direct linking loader.

1. External Symbol Dictionary(ESD).
2. The Text Card(TXT).
3. The Relocation and Linkage Directory(RLD).
4. The END card.

3. What are overlays?

Overlaying is a programming method that allows programs to be larger than the computer's main memory. An embedded system would normally use overlays because of the limitation of physical memory, which is internal memory for a system-on-chip, and the lack of virtual memory facilities.

4. **What is dynamic loading?**

   Dynamic loading is a mechanism by which a computer program can, at run time, load a library (or other binary) into memory, retrieve the addresses of functions and variables contained in the library, execute those functions or access those variables, and unload the library from memory.

5. **Define Relocation factor.**

   The translated and linked origins of program P be t_origin (p) and l_origin (p), respectively. Consider a symbol symb in P. Let its translation time be t (symb) and link time address be l (symb). The relocation factor of P is defined as relocation_factor = l_origin (p) – t_origin (p). This value can be positive, negative or zero.

UNIT - 5

1. **What is intermediate form?**

   The process of generating the object code for each construction after determining the syntactic construction is known as intermediate form.

2. **List the three tasks of lexical phase.**

   The three tasks of Lexical phase:

   1. To parse the source program into the basic elements or tokens of the language.
   2. To build a literal table and an identifier table.
   3. To build a uniform symbol table.

3. **What are the three classes of uniform symbols?**

   The three classes of uniform symbols are:

   1. Identifier(IDN).
   2. Terminal Symbol(TRM).
   3. Literal(LIT).

4. **Define local and global optimization (Repeated twice).**

   Global optimization: Removing or deleting the duplicate entries in the matrix and modifying all references to the deleted entries is one type of machine independent optimization.

5. **What is a compiler?**

   A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses.

6. **What are tokens? Give an example.**

   Tokens in C. Tokens are the smallest elements of a program, which are meaningful to the compiler. The following are the types of tokens: Keywords, Identifiers, Constant, Strings, Operators, etc.

7. **What is code generation phase?**

   In computing, code generation is the process by which a compiler's code generator converts some intermediate representation of source code into a form (e.g., machine code) that can be readily executed by a machine. Sophisticated compilers typically perform multiple passes over various intermediate forms.

*****

SOUNDARYA EDUCATIONAL TRUST (R)

SOUNDARYA INSTITUTE OF MANAGEMENT & SCIENCE

Soundarya Nagar, Sidedahalli, Hessaraghatta Main Road, Bangalore – 73

Department of Computer Science

SOLUTION BANK

BCA602T – System Programming

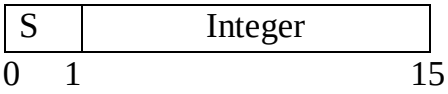
5 MARKS QUESTIONS AND ANSWERS

UNIT – 1

1. Explain different data formats used in IBM 360 with an example.

Data is storing in memory as group of bits. This group of bits can be characters, numbers or special characters. The different formats present in IBM 360 are:

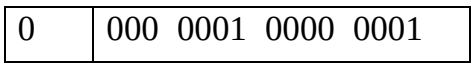
a. **Short form fixed point numbers**: in short form fixed point 16 bits is allocated to represent an integer number, out of this the first 1 bit is used the sign 0 for + and 1 for – negative. The machine interprets the contents of these two bytes as an integer. It represents the first bit as sign and the remaining 15 bits as binary number.



S is Sign bit

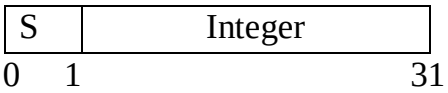
Ex: Decimal number +257 is represented as

----------16 bits ----------



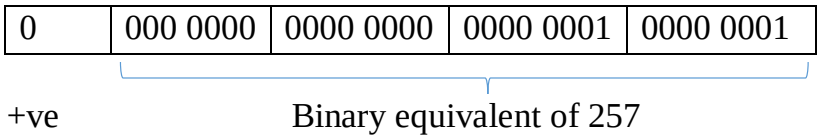
+ve Binary equivalent of 257

b. **Long form fixed point numbers**: In long form fixed point 32 bits are allocated, out of this first one bit is used for sign bit(+ or -). The m/c interprets the contents of this four bytes as an integer. It interprets the first bit as sign and the remaining 31 bits are as a binary number.

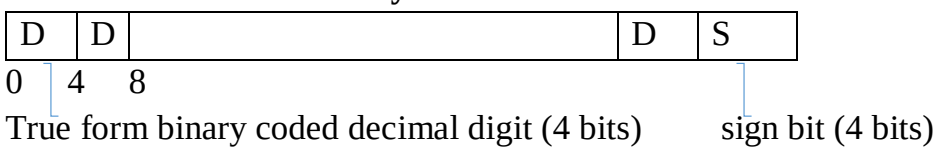


S is Sign bit

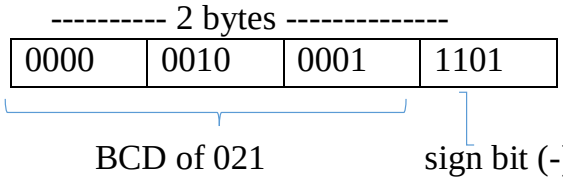
Ex: Decimal number +257 is represented as



c. **Packed decimal numbers**: Decimal digits, packed two to a byte, appear in fields of variable length and are accompanied by a sign in the right most four bits. Instead of representing the binary numbers, numbers are represented in binary coded decimal format.

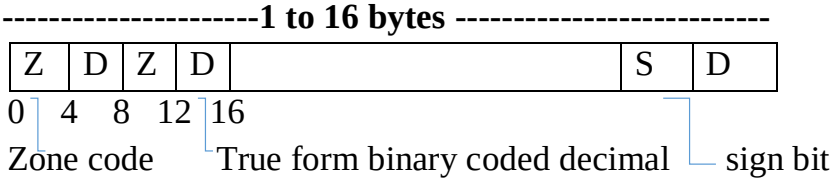


Ex: Decimal Number -021 is represented as

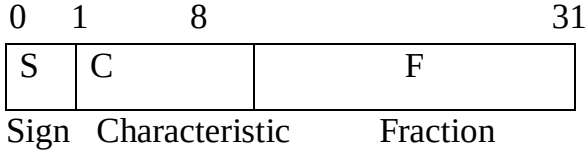


Note: sign bit contains the BCD of hexadecimal digits A, B, C, D, E and F. The hex digits C, A, F and E indicate a positive number while D and B indicates a negative number.

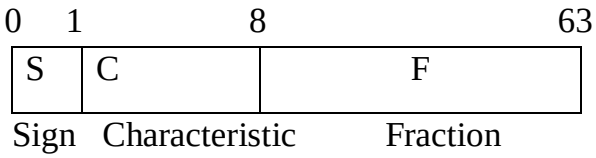
d. **Unpacked decimal numbers**: The size of the unpacked numbers varies from 1 to 16 bytes out of which the last 1 byte is reserved for sign and data. Instead of representing the binary number, numbers are represented in binary coded decimal.



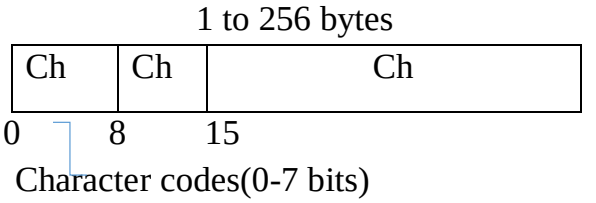
e. **Short form floating point numbers**: In short form floating point 32 bits are allocated. Out of this 1 bit is reserved for sign (+ or -). The floating-point number is divided into exponential and fractional part. This representation gives a precision of seven decimal places.



f. **Long form floating point numbers**: In long form floating point 64 bits are allocated for the floating-point number which contains the exponential and fractional part. The long length of the fractional part gives up to 17 decimals of precision, thus eliminating most requirements for double precision arithmetic.

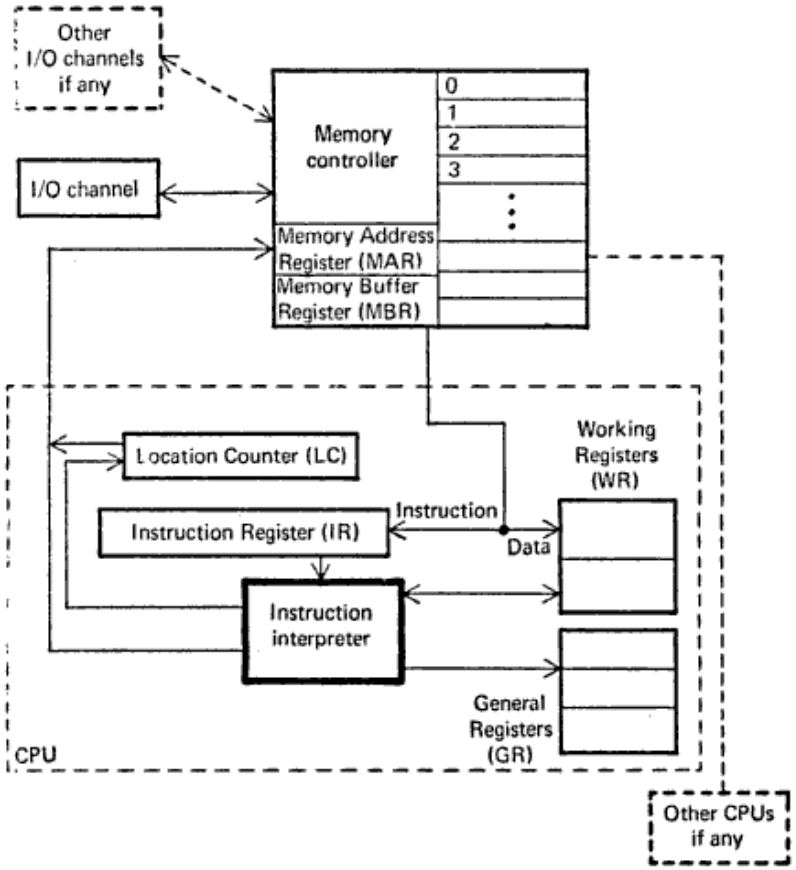


g. **Logical data**: Characters are stored as 8-bit ASCII code each character occupies 1 byte. The length of character data, varies from 1 to 256 bytes.



2. Draw the general machine structure of IBM 360/370 with a neat diagram.

The structure of CPU for a typical Von Neumann Machine is as follows:



The structure above consists of :

1. Instruction Interpreter
2. Location Counter
3. Instruction Register
4. Working Registers
5. General Register

The Instruction Interpreter Hardware is basically a group of circuits that perform the operation specified by the instructions fetched from the memory. The Location Counter can also be called as Program/Instruction Counter simply points to the current instruction being executed. The working registers are often called as the "scratch pads" because they are used to store temporary values while calculation is in progress.

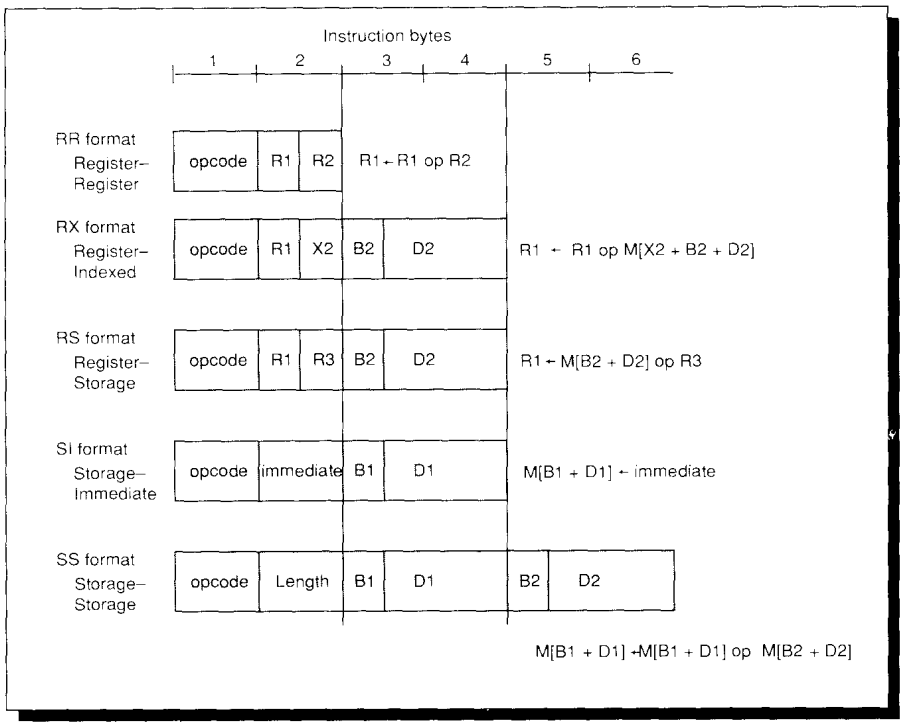
This CPU interfaces with Memory through MAR & MBR. MAR (Memory Address Register) - contains address of memory location (to be read from or stored into). MBR (Memory Buffer Register) - contains copy of address specified by MAR. Memory controller is used to transfer data between MBR & the memory location specified by MAR. The role of I/O Channels is to input or output information from memory.

3. Explain various instruction formats used in IBM 360/370 machine.

The 360/370 has five instruction formats. Each format is associated with a single addressing mode and has a set of operations defined for that format. While some operations are defined in multiple formats, most are not.

- **RR** (register-register): It denotes register to register operation. That is both the operands are registers.
- **RX** (register-indexed: It denotes a register and indexed storage operations. That is one operand is a register and another one storage operand.
- **RS** (register-storage): It denotes register and storage operation.
- **SI** (storage-immediate): It denotes a storage and immediate operand operation.
- **SS** (storage-storage): It denotes a storage operation.

The general formats are given in below figure.



4. Explain any 5 pseudo ops, used in an assembly language program.

Pseudo op is an assemble language instruction that specifies an operation of the assembler.

1. **USING**: It is a Pseudo-op that indicates to the assembler which general purpose register to use as a base register and what its contents will be in order to convert the addresses to base-displacement format. Since no special register is set aside for addressing, the programmer must inform the assembler which register to use as a base register and what value it contains at execution time.

   Syntax: USING <contents of base register> <GPR to be used as base register >

   EX: USING *, 5
2. **START**: It is Pseudo-op that tells the assembler where the beginning of the program and allow the user to give a name to the program.

   Ex: TEST START {TEST is the name of the program}
3. **END**: It is Pseudo-op that tells the assembler that the last statement of the program has been reached.

   Ex: END
4. **EQU**: It is Pseudo-op which allows the programmer to define variable. The operand of the EQU statement can be any constants or valid arithmetic expressions.

   Ex: BASE EQU 15

5. **DC(Define Constant):** DC is declarative Pseudo-op used to create a memory area to hold a constant value. That is it constructs memory words containing constants.
   Syntax: <Label> DC 'Constant'
   Ex: FOUR DC F'4'
6. **DS(Define Storage)**: DS is declarative Pseudo-op that reserves some storage for the data and gives them a name.
   Syntax: <Label> DS SIZE
   Ex: BLOCK DS 100F
7. **DROP**: It is a Pseudo-op which indicates an unavailable base register.
   Syntax: DROP <BR number>
   Ex: DROP 15
8. **LTORG:** It is a Pseudo-op which tells the assembler to place the encountered literals at an earlier location.

5. Explain open subroutine and closed subroutine with an example.

Open subroutine:

An **open subroutine or macro description** is one whose code is inserted (flow continues) into the main program. Therefore, if four times were called the same open subroutine, it would appear in the calling system in four different places.

Closed subroutine:

A **closed subroutine can also be stored outside the main routine**, and monitor transfers to the subroutine. Associated with the closed subroutine are the two tasks the main program should perform a transfer of control & transfer of data.

Open Subroutine	Closed Subroutine
Open subroutine of macro definition is one whose code will be inserted at the point of function call within the main definition	A closed subroutine will be stored outside the main routine and there is a transfer in control to the subroutine for processing it
If the macro definition is very large and if you call these functions frequently shortage of memory may occur.	Large size macros can be executed, any number of times

Saves time because there is no overhead of program control transfer and return	There is an overhead of transferring the control to the function and returning back which takes time
Performs only one task i.e insert a macro	Performs two tasks that is transfer of control and transfer of data
Wastage of memory	Saves memory
These are loaded into memory at different memory locations. i.e based on location of calling macro	These are loaded into memory at a specific address.

6. Explain Long-way-no-lopping with example.

The straight forward approach is without using any looping statements. The implementation is easy. Instructions are repeated for all the data items. It is impossible to access the first data item and the last data item using register 1 as the base. Wastage of memory. Need of relocation. Instruction would overlap data in the core. Suppose you wanted to process 300data items, then the storage needed for the instructions.

=instruction * number of data

=instruction + size of instruction

= 3*300*4

=3600 bytes

7. Explain address modification using instruction as data.

In straight forward approach we used 3 instructions, i.e. load, add and store which was repeated. The only change was I the offset of load and store instructions, i.e. first offset was 904, then 908, then 912 etc.,

Another approach to the problem is to write a program consisting only of those 3 instructions followed by a sequence of commands that would change the offset of the load and store instruction by adding 4 to them.

Advantages:

1. Saves memory.
2. Address is modified easily using the instruction.

Disadvantages:

1. Treating instructions as data is not a good programming practise because over a period of time, it is difficult to understand what the original programme was.
2. Separate instructions should be used for increasing the displacement of load and store instructions.

8. Explain the use of literals in assembly language programs using example.

Literals: In many instructions, it is possible to use a shorthand notation in place of a label for the source memory operand. A literal look like a DC operand but is preceded by a '=' sign. While using literals, the assembler creates data areas containing the constants.

Ex:

```
        L     3,    TEN
TEN     DC    F '10'
```

Can be replaced by

```
        L     3,    =F'10'
```

Where =F'10' is a literal and a data area whose size is 1 full word, containing the value 10 is created. In the load instruction, the address part points to a full word containing the value 10.

Literal Table: All the constants that have been specified through literals will be placed in a table called literal table. Usually it is placed at the end of the program. LTORG is a pseudo-op which tells the assembler to place the encountered literals at an earlier location. It is used when the program is very long. To avoid addressability problems in multimodular programs, LTORG instruction is coded as the last instruction of each control section in the program. So, each control section in the program will have its own literal pool at the end of the module.

9. Differentiate between Pseudo-op and Machine-op with example.

Pseudo-op Please Refer Question no 4.

Machine-op: Machine-op represents to the assembler a machine instruction. Machine instructions should be included in the program, to load the proper value into the base register at execution time.

The different machine-op's are:

1. **BALR**: It is a branch and link instruction. It is an instruction to the computer to load a register with the next address and branch to the address specified in the second field. If the second operand is register o, then execution proceeds with the next instruction. BALR loads the base register and is not an executable statement. It is an RR type instruction; whose length is 2 bytes. Ex: BALR BASE, 0
2. **BR**: BR is a machine-op indicating branch to the location whose address is in general register. The calling programs leaves their return address in the corresponding general register. Ex: BR 14
3. **BCT**: It indicates branch on count. It is a RX type instruction; whose size is 4 bytes. Ex: BCT 3, LOOP

UNIT – 2

1. Draw an overview flowchart for Pass1 of an assembler.

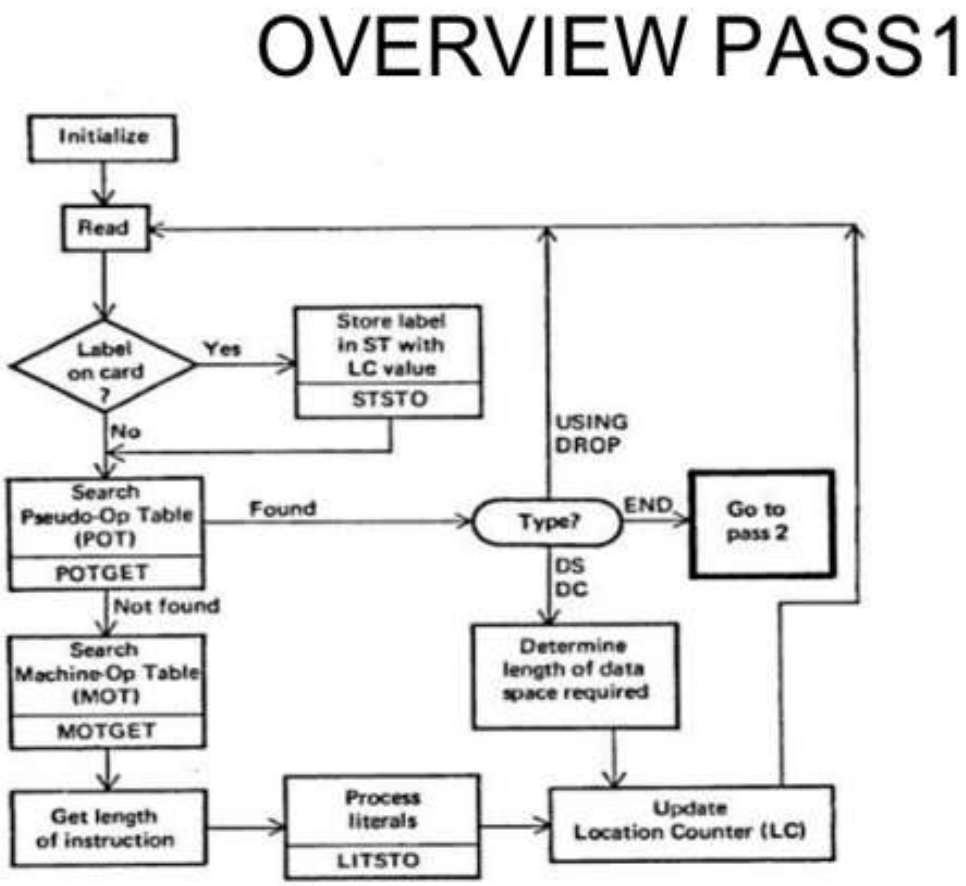
In general, an assembler must do the following tasks:

1. **Generate Instructions**:
   a. Evaluate the mnemonic of the operations field to produce its machine code.
   b. Evaluate the sub fields- find the value of each symbol, process literals and assign addresses.
2. **Process pseudo ops:**

These tasks can be grouped into two passes, associated with each task is one or more assembler modules. Pass is a sequential scan over the input.

Pass1: Purpose of pass1 is to define symbols and literals.

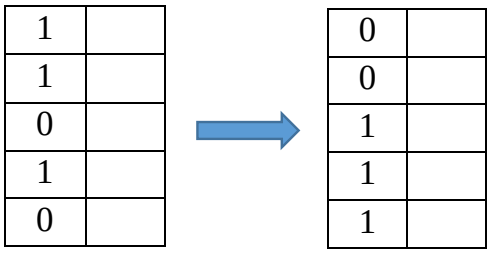
1. Determine length of machine instructions (MOTGET1)
2. Keep track of location counter (LC)
3. Remember values of symbols until pass2 (STSTO)
4. Process some pseudo-ops, eg, EQU, DS (POTGET1)
5. Remember literals (LITSTO)



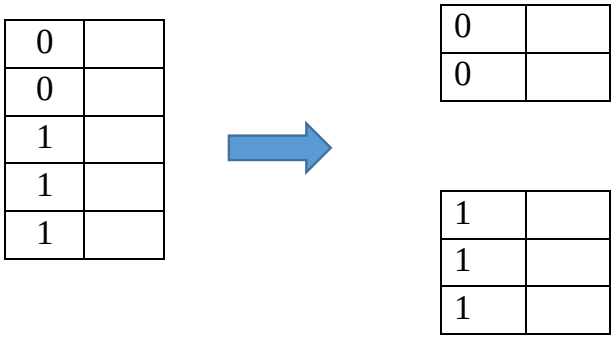
2. Explain radix sort with suitable example.

Radix exchange sort is a distributive sorting method used when the keys are represented in binaries. Sorting is accomplished by considering groups with the same (M) first bits and ordering that group with respect to the (M+1)s bit. The ordering of a group on a given bit is accomplished by scanning down from the top of the group for a one bit and up from the bottom for a zero bit, these two are exchanged and the sort continues.

Ex: 1. sort array with respect to left most bit



2. Partition array



3. Recursion

Recursively sort top sub array ignoring left most bit.

Recursively sort bottom sub array ignoring left most bit

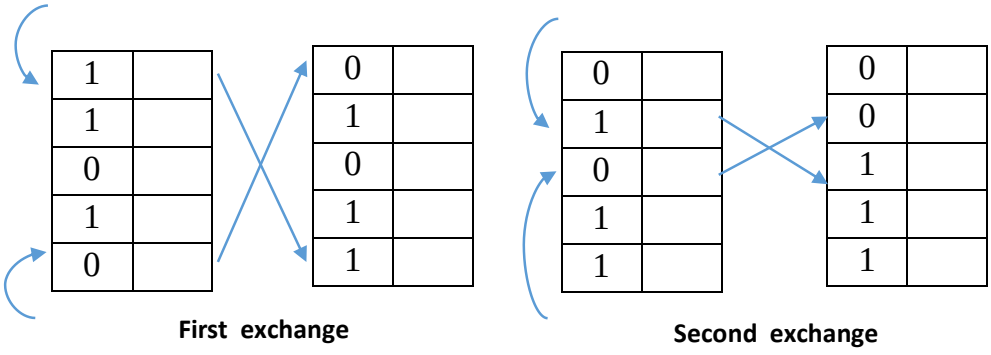
Sorting is done like partitioning in the quick sort.

The Method is Repeat

Scan top-down to find key starting with 1.

Scan bottom-up to find key starting with 0.

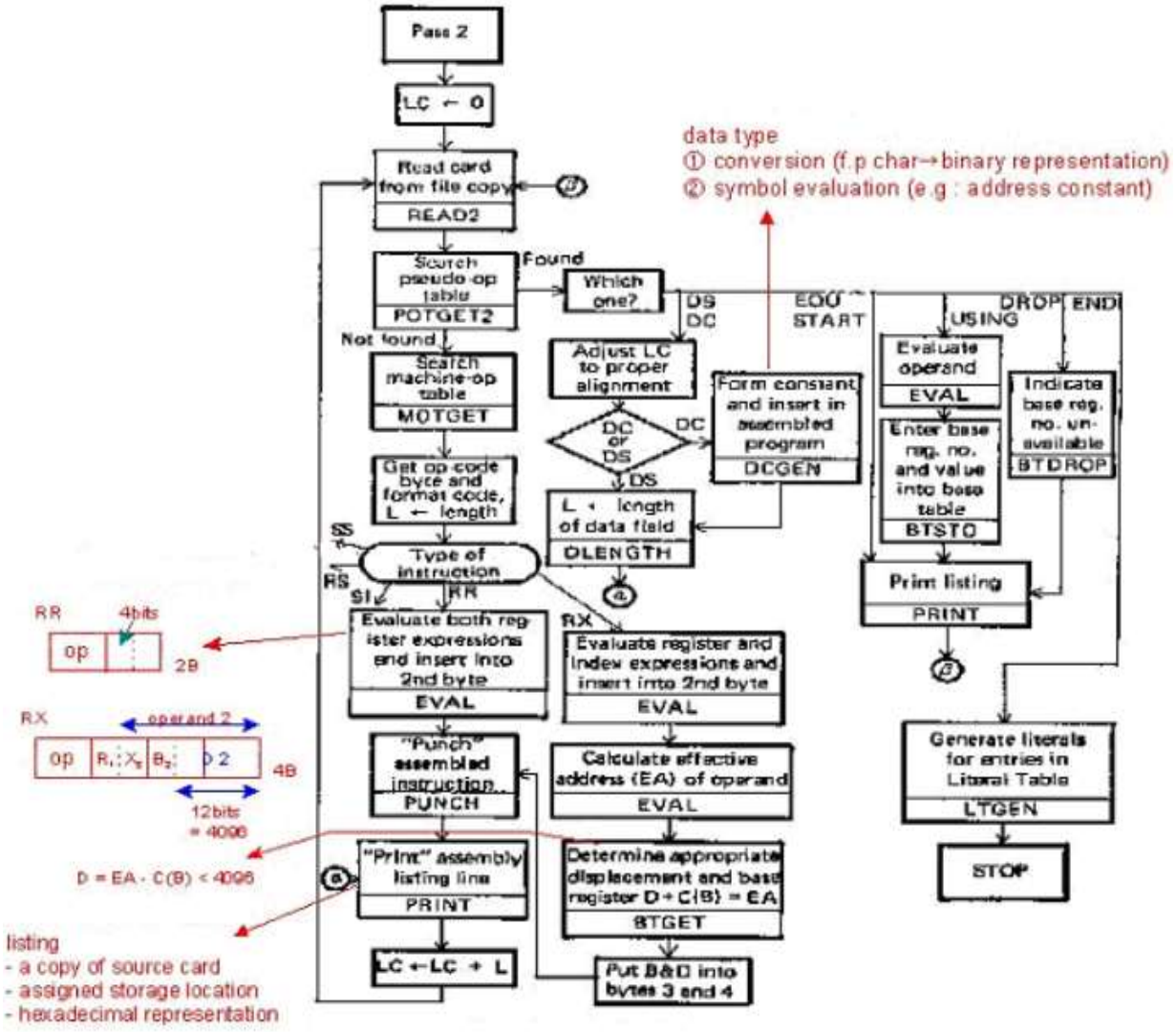
Exchange keys. Until scan indicated cross.



3. Draw the detailed Pass 2 flowchart of an assembler.

Pass2: Generate code: The purpose of Pass2 is to

1. Process each card and to find the values for its operation code and its operand field.
2. Structuring the generated code into the appropriate format for later processing by the loader.
3. To print an assembly listing containing the original source and the hexadecimal equivalent of the bytes generated.



4. Explain databases used by Pass1 and Pass2 of an assembler.

Pass1 Database:

1. Input source program.
2. A Location Counter (LC), used to keep track of each instruction's location.
3. A table, the Machine-operation Table (MOT), that indicates the symbolic mnemonic for each instruction and its length.
4. A table, the Pseudo-Operation Table (POT), that indicates the symbol mnemonic and action to be taken for each pseudo-op in pass1.
5. A table, the Symbol Table(ST), that is used to store label and its corresponding value.
6. A table, the Literal Table(LT), that is used to store each literal encountered and its corresponding assigned location.
7. A copy of the input to b used by pass2. This may be stored in a secondary storage device.

Pass2 Database:

1. Copy of the source program input to pass1.
2. Location Counter(LC).
3. A table, the MOT that indicates for each instruction, symbolic mnemonic, length, binary machine op code and format.
4. A table, the POT, that indicates for each pseudo-op the symbolic mnemonic and the action to be taken in pass2.
5. The ST, prepared by pass1, containing each label and its corresponding value.
6. A table the Base Table(BT), that indicates which registers are currently specified as base register by USING pseudo-ops and what are specified contents of these registers.
7. A work-space, INST, that is used to hold each instruction as its various parts are being assembled together.
8. A work- space, PRINT LINE, used to produce a printed listing.
9. A work-space, PUNCH CARD, used prior to actual outputting for converting the assembled instructions into the format needed by the loader.
10. An output deck of assembled instructions in the format needed by the loader.

5. Illustrate Binary Search with an example.

Binary search algorithm searches for an item only in an ordered list. It starts with the middle entry in the table, and compares it with the search element. This give rise to 3 conditions. They are

1. The search element may be equal to the middle element.
2. The search element may be greater than the middle element.
3. The search element may be less than the middle element.

The next actions taken for each of these out comes is as follows.

1. If the search element is equal to the middle element, the search element is found.
2. If the search element is greater than the middle element, use the bottom half of the given table as a new table to search.
3. If the search element is smaller than he middle element, use the top half of the table as a new table to search.

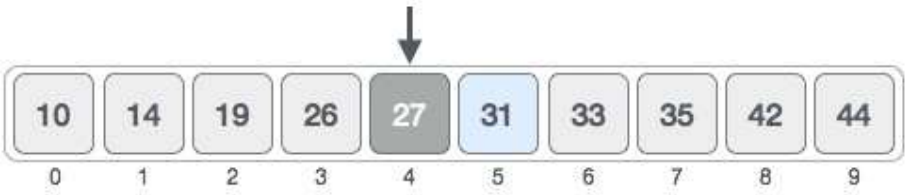
This method, effectively divides the table in half on each pass, systematically bracketing the item searched for. The search is terminated with a 'not found' condition when the length of the last sub table searched is down to 1 and the item has not been found.

For a binary search to work, it is mandatory for the target array to be sorted. We shall learn the process of binary search with a pictorial example. The following is our sorted array and let us assume that we need to search the location of value 31 using binary search.

First, we shall determine half of the array by using this formula −

mid = low + (high - low) / 2

Here it is, 0 + (9 - 0) / 2 = 4 (integer value of 4.5). So, 4 is the mid of the array.



Now we compare the value stored at location 4, with the value being searched, i.e. 31. We find that the value at location 4 is 27, which is not a

match. As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.

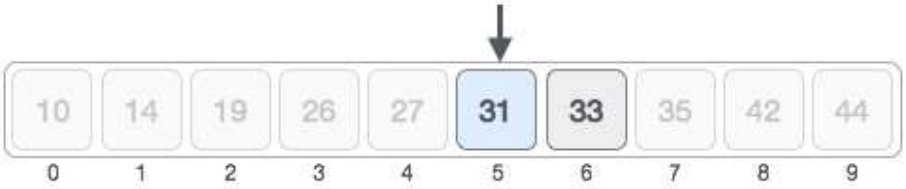
We change our low to mid + 1 and find the new mid value again.

low = mid + 1
mid = low + (high - low) / 2

Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.

The value stored at location 7 is not a match, rather it is more than what we are looking for. So, the value must be in the lower part from this location.

Hence, we calculate the mid again. This time it is 5.



We compare the value stored at location 5 with our target value. We find that it is a match.

We conclude that the target value 31 is stored at location 5.

6. List different types of sorting. Explain Interchange sort.

Different types of sorting:

1. Interchange sort.
2. Shell sort.
3. Bucket sort.
4. Radix exchange sort
5. Address calculations sort.

Interchange Sort: It is also known as **bubble sort, sinking sort or shifting sort**. It sorts by comparing each adjacent pair of items in a list and swapping the items if necessary, and repeating the pass through the list until no swap are done. After each pass at least one item is added to the bottom of the list in a perfect order. Thus, the name sinking sort, because the elements sinks down to the proper position.

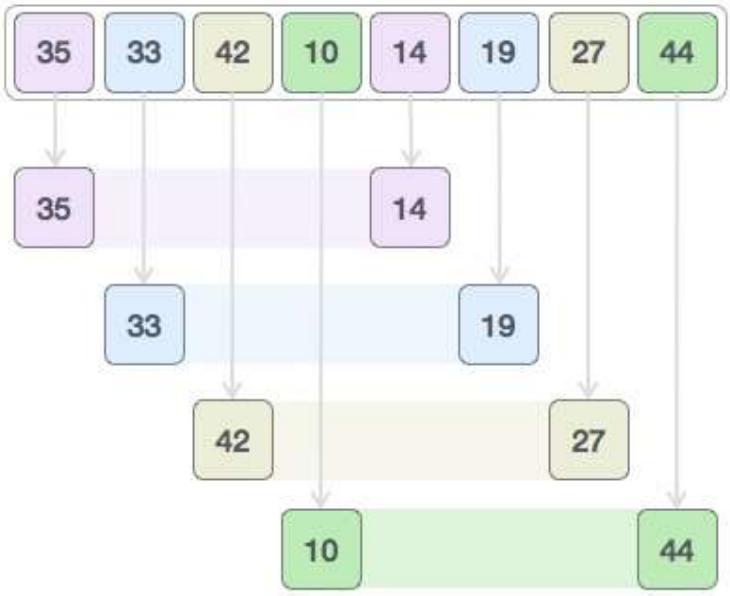
To arrange a list of items in ascending order using interchange sort, the first element is compared with the second element. If the first element is greater then the second, the elements are interchanged Next, the second element is compared with the third and the process continues until all the elements of the list are compared. After the end of this (pass1), the largest element bubbles down to the last position.

In the second pass, the same procedure is followed leaving out the last element since it is already placed in the correct position. This pass brings the second largest number to the last but one position. This procedure is repeated until there are no more elements for comparison.

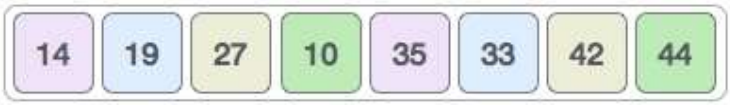
7. Explain shell sort with an example.

Shell sort is a highly efficient sorting algorithm and is based on insertion sort algorithm. This algorithm avoids large shifts as in case of insertion sort, if the smaller value is to the far right and has to be moved to the far left.

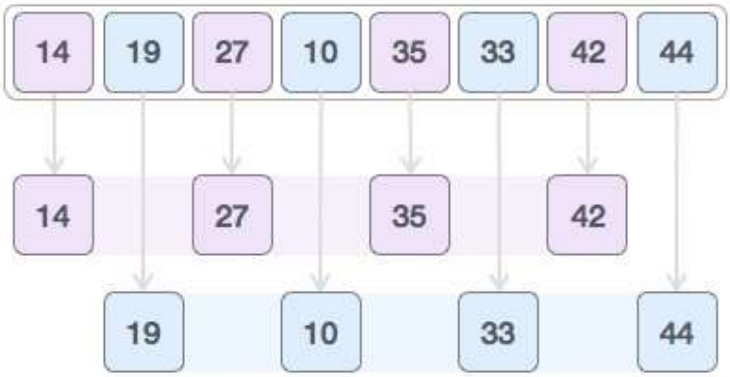
Let us consider the following example to have an idea of how shell sort works. We take the same array we have used in our previous examples. For our example and ease of understanding, we take the interval of 4. Make a virtual sub-list of all values located at the interval of 4 positions. Here these values are {35, 14}, {33, 19}, {42, 27} and {10, 44}



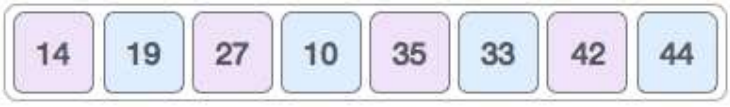
We compare values in each sub-list and swap them (if necessary) in the original array. After this step, the new array should look like this −



Then, we take interval of 1 and this gap generates two sub-lists - {14, 27, 35, 42}, {19, 10, 33, 44}



We compare and swap the values, if required, in the original array. After this step, the array should look like this −



Finally, we sort the rest of the array using interval of value 1. Shell sort uses insertion sort to sort the array.

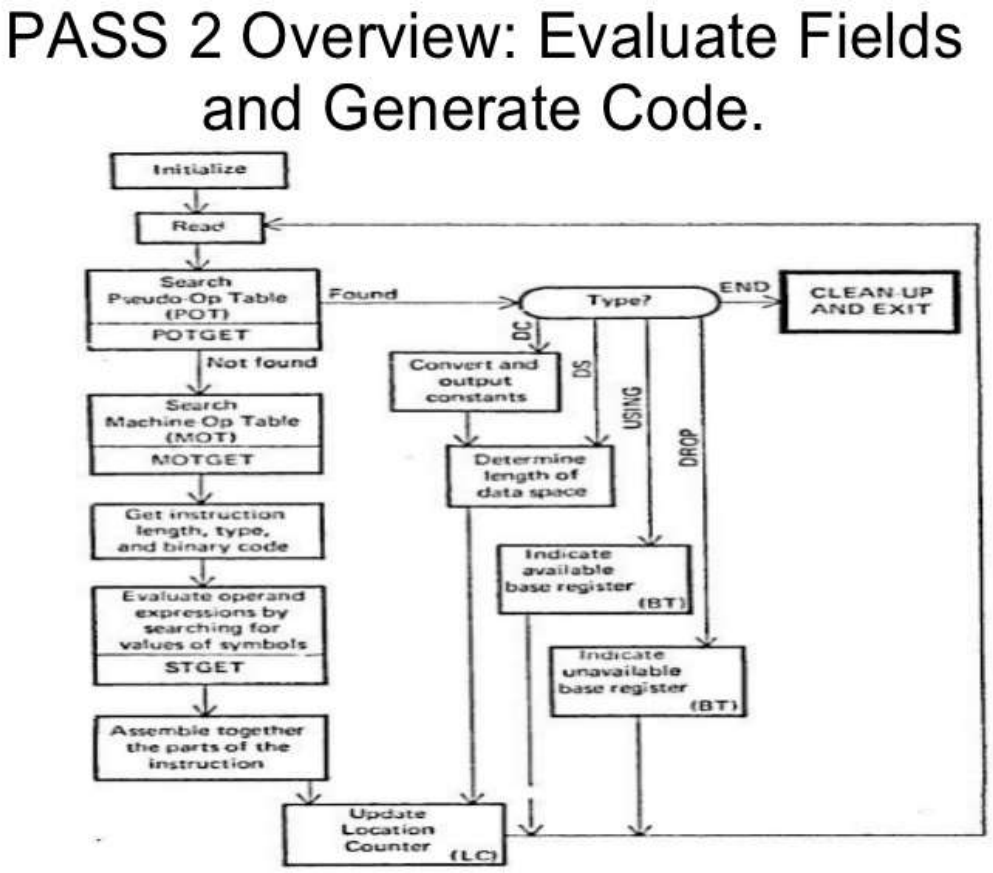
Following is the step-by-step depiction −

We see that it required only four swaps to sort the rest of the array.

8. Explain Pass 2 overview of an assembler with flow-chart.

Pass2: Purpose of pass2 is to generate object program.

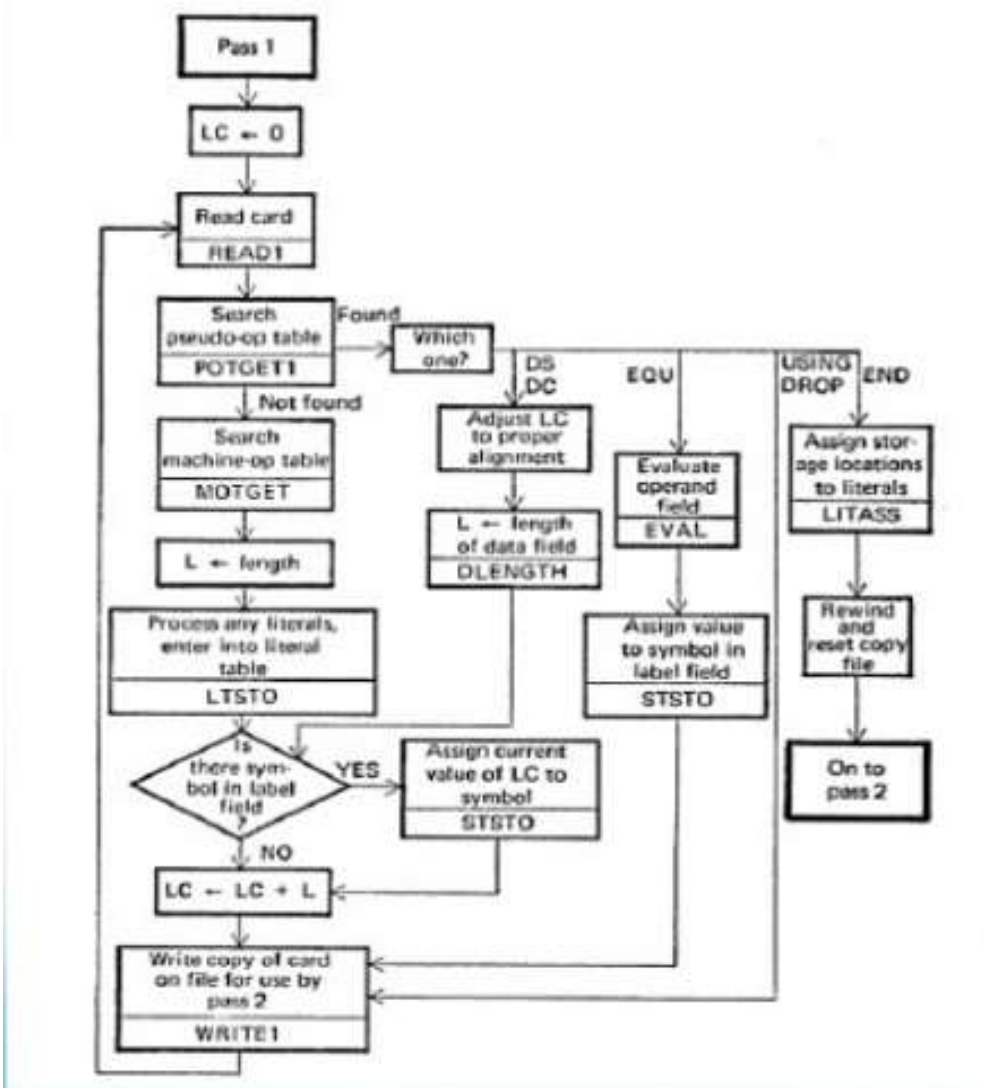
1. Look up value of symbols(STGET).
2. Generate instructions(MOTGET).
3. Generate data (for DS, DC and literals).
4. Process pseudo ops(POTGET2).



9. Draw the detailed Pass1 flowchart of an assembler.

Pass1: Define symbols: The important purpose of pass1 is to

1. Assign a location to each instruction and data and to define pseudo instructions.
2. To define values for symbols appearing in the label field of the source program.



Detailed Pass1 Flowchart

10. Give format of all five tables used in assembler.

Assembler consists of five tables. MOT, POT, ST, LT and BT.

MOT: Machine Operation Table:

MOT is a fixed table that contains the mnemonic of each instruction. Fixed table is a table, whose contents are not filled in or altered during the assembly process.

Format of MOT

----------6 bytes per entry ----------

Mnemonic op-code (4 bytes) (Characters)	Binary op-code (1 byte) (Hexadecimal)	Instructions Length (2bits) (Binary)	Instruction format (3bits) (Binary)	Not used in this design (3 bits)
"Abbb"	5A	10	001	
"AHbb"	4A	10	001	
"ALbb"	5E	10	001	
"ALRb"	1E	01	000	
"ARbb"	1A	01	000	
-----	---	---	---	
"MVCb"	D2	11	100	
----	---	----	---	

b~ represents the character "blank"

codes: **Instruction Length**

01= 1 half-words = 2 bytes
10= 2 half-words = 4 bytes
11= 3 half-words = 6 bytes

Instruction Format

000=RR
001=RX
010=RS
011=SI
100=SS

MOT for both Pass1 and Pass2

POT: Pseudo-Operation Table

The POT is fixed table that contains the pseudo-ops and corresponding actions. POT for pass1 and pass2 are the same and it contains the name and address. Size of POT is 8 bytes per entry.

Format for POT

----------8 bytes per entry----------

Pseudo-op (5 bytes) (character)	Address of routine to process pseudo-op (3-bytes=24 bit address)
"DROPb"	P1DROP
"ENDbb"	P1END
"EQUbb"	P1EQU
"START"	P1START
"USING"	P1USING

POT is similar for Pass1 and Pass2

ST: Symbol Table

ST is a variable table that contains labels and its values. ST for pass1 and pass2 are the same and contains symbol, value, length and relocation fields. Size of the table is14 bytes per entry.

ST Format:

----------14 bytes per entry----------

Symbol (8 bytes) Character	Value (4 bytes) Hexadecimal	Length (1 byte) Hexadecimal	Relocation (1 byte) Character
"JOHNbbbb"	0000	01	"R"
"FOURbbbb"	000C-	04	"R"
"FIVEbbbb"	0010-	04	"R"
"TEMPbbbb"	0014-	04	"R"

ST is similar for Pass1 and Pass2

LT: Literal Table

LT is a variable table that contains literals and its value. It is same like symbol table but instead of symbols, here we have literals.

LT Format:

----------14 bytes per entry----------

Symbol (8 bytes) Character	Value (4 bytes) Hexadecimal	Length (1 byte) Hexadecimal	Relocation (1 byte) Character
F'4'	4	04	"R"

BT: Base Table

BT is a variable table that specifies the base register. It is used only in pass2. The size of it is 4 bytes per entry.

BT Format:

----------4 bytes per entry----------

Availability indicator (1byte) Character		**Designated relative-address contents of base register (3 bytes=24 bit address) Hexadecimal**
1	"N"	---
2	"N"	---
:	:	
14	"N"	--
15	"Y"	00 00 00

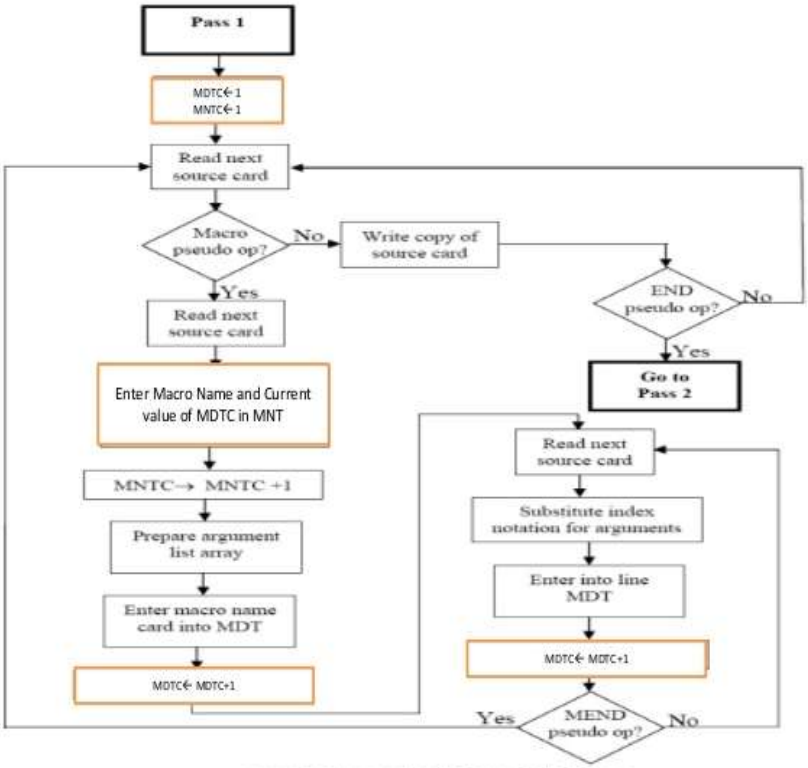
Availability indicator, Y means register specified in USING pseudo-op and N means register never specified in USING pseud-op.

UNIT – 3

1. Write an algorithm for Pass1 processing macro definitions.

Pass 1 Algorithm:

- Pass1 of macro processor makes a line-by-line scan over its input.
- Set MDTC=1 as well as MNTC=1.
- Read next line from input program.
- If it is a MACRO pseudo-op, the entire macro definition except this (MACRO) line is stored in MDT.
- The name is entered into Macro Name Table along with a Pointer to the first location of MDT entry of the definition.
- When the END pseudo-op is encountered all the macro definitions have been processed, so control is transferred to pass2.



pass 1 Processing Macro definitions

2. Explain implementation of macro processor within assembler.

The macro processor can be implemented within an assembler using two different ways,

1.It can be added as a pre-processor to an assembler, making a complete pass over the input text before pass1 of the assembler.

2.It can be implemented within pass1 of the assembler.

Advantages of incorporating the macro processor into pass1 of assembler:

1.Many functions do not have to be implemented twice.

2.There is less over head during processing.

3.Functions are combined and it is not necessary to create intermediate files as output from the macro processor and input to the assembler.

4.More flexibility is available to the programmer in that he/she may use all the features of the assembler in conjunction with macros.

Disadvantages of incorporating the macro processor into pass1 of assembler:

1.The program becomes too large to fit into the core of some machines.

2.The program becomes complex with macros.

3. Explain macro facility with macro instruction argument with an example.

Following are the important features of macro:

1. Macro instruction arguments 2. Conditional Macro Expansion
3. Macro calls within macros 4. Macro Instructions Defining Macros

Macro Instruction Arguments : The macro which we have seen before lacks flexibility, because all of the calls to any given macro will be replaced by identical blocks. There is no way for a specific macro call to modify the coding that replace it.

In order to overcome this problem, we use macro instruction in macro calls. The corresponding macro dummy arguments call appear is macro definitions.

Two methods of arguments:

1.Positional Arguments: These are matched with the dummy arguments

Ex: INCR A,B,C

2.Keyword Arguments: Allows reference to dummy arguments by name as well as by position.

Ex: INCR &ARG1=A, &ARG2=B, &ARG3=C.

Macro without argument

```
A       1, DATA 1
A       2, DATA 1
A       3, DATA 1
        :
        :
A       1, DATA 2
A       2, DATA 2
A       3, DATA 2
        :
        :
DATA DC     F '5'
DATA DC     F  '10'
```

Example with macro with a argument

```
MACRO
INCR    &ARG    ( Argument is defined with macro name)
   A    1, &ARG
   A    2, &ARG
   A    3, &ARG
MEND
        :
        :
INCR    DATA 1          Use DATA 1 as operand
        :
INCR    DATA 2          Use DATA 2 as operand
DATA DC     F '5'
DATA DC     F  '10'
```

4. Explain macro definitions with an example.

Macro instruction definition defines a name for sequence of operations.

Syntax:

```
MACRO           Start definition
[     ]         Macro name
_____  }
_____  }        Sequence to be abbreviated
_____  }
MEND            End of definition
```

1. Macro definition starts with MACRO pseudo-op, it indicates the beginning of a macro definition.
2. Following the MACRO pseudo-op is the name of the macro.
3. Next to macro name sequence of instructions to be abbreviated. These instructions are compressed macro instructions.
4. The macro definition is terminated with MEND pseudo-op, which indicates the end of macro definition.

Example to define Macro

```
MACRO          Start definition
INCR           Macro name
A      1, DATA
A      2, DATA
A      3, DATA
MEND           End of definition
```

5. Explain simple one pass macro processor.

A Single pass Algorithms for Macro Definitions within Macro.

In case of macro definition within the macro, the inner macro is defined only after the outer macro has been called.

Therefore for any use of inner macro, we have to repeat both macro definition and the macro call passes.

In order to avoid this repetition we can reduce all macro processing to single pass.

All macro definitions must be processed before calls because macros must be defined to the processor before it can expand calls.

Additional Data Structure Included

MDI-Macro Definition Input indicator: It keep track of macrocalls. MDI="ON" while expanding a macro call otherwise MDI="OFF".

MDLC-Macro Definition Level Counter: It is used to keep track of macro definitions. It is a counter for MACRO and MEND pseudo-ops, when MACRO pseudo-op is encountered, MDLC is incremented by one and is decremented by one when MEND pseudo-op is encountered.

MDT format:

Index	Card
1	DEFINE &SUB
2	MACRO
3	#1 &Y
4	CNOP 0,4
5	BAL 1,*+8
6	DC A(&Y)
7	L 15=V(#1)
8	BALR 14.15
9	MEND
10	MEND

MNT Format:

Index	Name	MDT Index
1	DEFINE	1

6. Explain conditional macro expansion.

Conditional Macro Expansion

- The sequence of macro expansion scan be reordered based on some conditions using conditional macro expansion.
- It allows conditional selection of the machine instructions that appear in expansions of a macro call.
- For macro expansion two pseudo-op are used **AIF** and **AGO.**

AIF is a conditional branch pseudo-op.

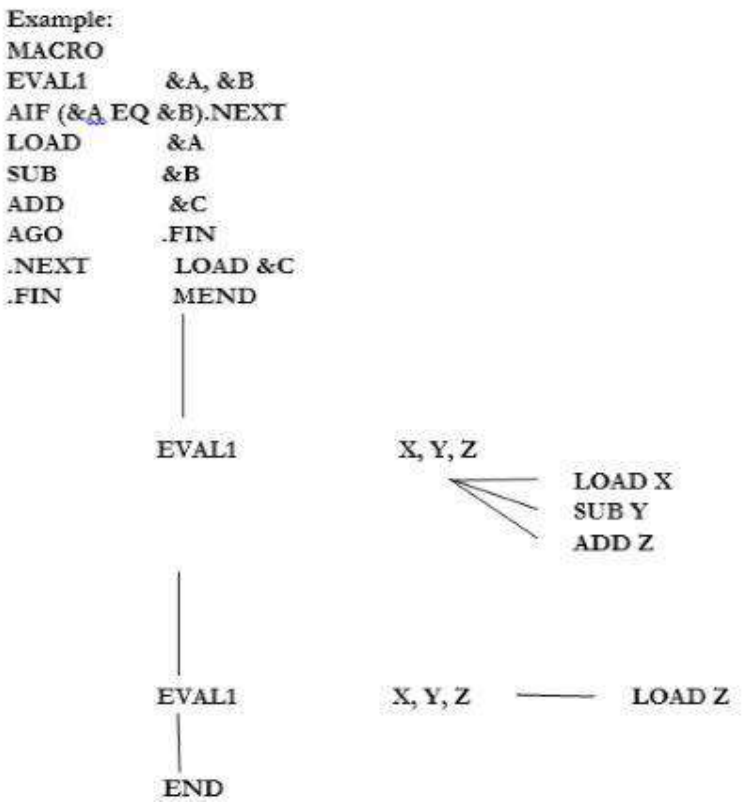
Format: AIF<Expression> <Sequencing Symbol>

Ex: AIF(&COUNTEQ1)FIN

Where expression is a relational expression.

AGO is an unconditional branch pseudo-op.

Format: AGO <Sequencing Symbol> **Ex**: AGOFIN



7. Explain macro instructions defining macros with an example.

Macro definition, which itself is a sequence of instruction, can be abbreviated by using macro.

It is important that inner macro is not defined until the outer macro is called.

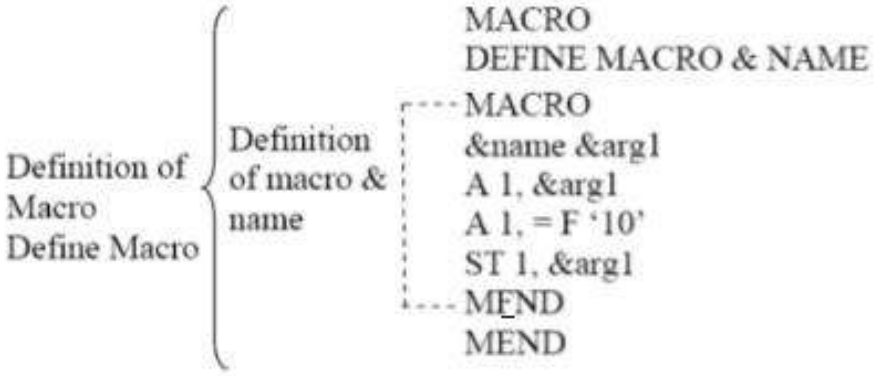
The user can call the above macro with the statement **DEFINE MACRO ADD1**

When this call is executed then the macro ADD1 will be defined. So after the above call user can call the ADD1 macro as following:

ADD1DATA1

The microprocessor will generate the sequence as

L1,DATA1
A1,=F'10'
ST1,DATA1



8. Give the database specifications for Pass1 and Pass2 of macro processor.

Pass1–Databases:

1.The input macro source deck.

2.The output macro source deck copy for pass2.

3.MDT (Macro Definition Table) to store the body of the macro definitions.

4.MNT(Macro Name Table) to store the names of defined macros.

5.MDTC(Macro Definition Table Counter) to indicate the next available entry in MDT.

6.MNTC(Macro Name Table Counter) to indicate the next available entry in MNT.

7.ALA(Argument List Array) to substitute index markers for dummy arguments before storing a macro definition.

Pass 2 –Databases:

1.Copy of the input macro source deck.

2.Output expanded source deck to be used as input to the assembler.

3.MDT created by Pass1.

4.MNT created by Pass1.

5.MDTP(Macro Definition Table Pointer) used to indicated the next line of text to be used during macro expansion.

6.ALA to substitute macro call arguments for the index markers in the stored macro definition.

9. Explain the four basic tasks of microprocessor.

Basic Tasks/ Functions of a Macro processor: There are four basic tasks or functions that any macro instruction processor must perform.

- **Recognize macro definition**: A macro instruction processor must recognize macro definition identified by MACRO and MEND pseudo-op.

- **Save the definitions**: The processor must store the macro-definitions which will be needed at the time of expansion of calls.

- **Recognize the Macro call**: The processor must recognize macro calls that appear as operation mnemonics.

- **Expand calls and substitute arguments**: The macro call is replaced by the macro definition. The dummy arguments are replaced by the actual data.

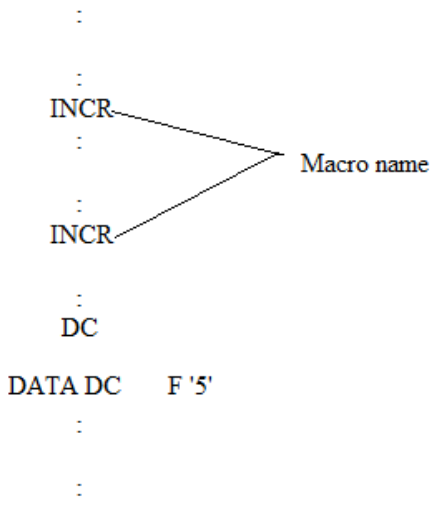
10. Explain the terms macro definition, macro call, macro expansion with syntax and example.

Macro definition Refer Question no 4

Macro Call

Once the macro has been defined, the use of macro name as an operation mnemonic in an assembly program is equivalent to the corresponding instruction sequence. i.e., in the program, the repeated instruction sequence are written with the macro name. This is called as **macro call or macro invocation**.

Ex:



Macro Expansion

Whenever there is a macro call the macro processor substitutes the macro definition in the place of the macro call. This is called as **Macro Expansion**.

•The macro definition itself appear in the expanded source code.

•The definition is saved by the microprocessor.

EX:Macro will expand the code as following in the source code**.**

```
A       1, DATA
A       2, DATA
A       3, DATA
```

UNIT – 4

1. Explain the design of Absolute Loader with a neat diagram.

Absolute Loader:

In this scheme the object program is placed in secondary devices. The loader only accepts the machine language text & places into core at the location prescribed by the assembler.

Disadvantage is that the programmer must specify the load address in the program & also if there are multiple subroutines, the programmer must remember the address of each & use that in other subroutines to perform subroutine linkages.

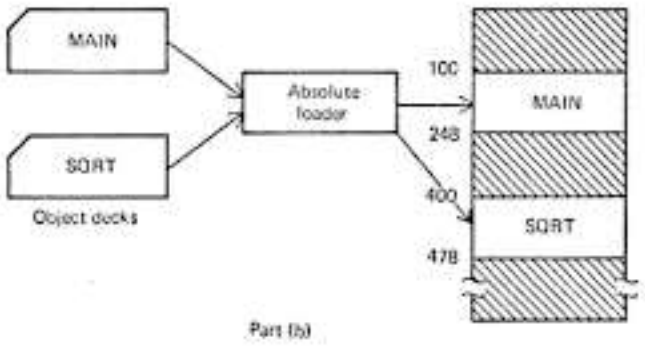


FIGURE 5.4 Absolute loader example

Design of Absolute Loader:

Absolute loader performs the tasks of allocation, relocation and linking. Therefore, it is only necessary for the loader to read cards of the object deck and move the text on the cards into the absolute locations specified by the assembler.

There are two types of information that the object deck communicate from the assembler to the loader.

1. It must convey the machine instructions that the assembler has created along with the assigned core locations.
2. It must convey the entry point of the program, which is where the loader is to transfer control when all instructions are loaded.

Text Cards for instructions and data	
Card Column	**Contents**
1	Card type=0 for text card identifier
2	Count of number of bytes of information on card .(1 byte per column)
3-5	Address at which data on card is to be put.
6-7	Empty (used for validity checking)
8-72	Instructions and data to be loaded.
73-80	Card sequence number

Transfer cards to hold entry point to program	
Card Column	**Contents**
1	Card type=1 for transfer card identifier
2	Count=0
3-5	Address of entry point
6-72	Empty
73-80	Card sequence number

2. Write a note on direct linking loader.

- It is a general relocatable loader, and is perhaps the most popular loading scheme presently used.
- It has advantage of allowing the programmer multiple procedure segments and multiple data segments.
- Other segments provide flexible intersegment referencing and accessing ability, while translation of the programs.
- The assemble must give the loader the following information with each procedure or data segment:

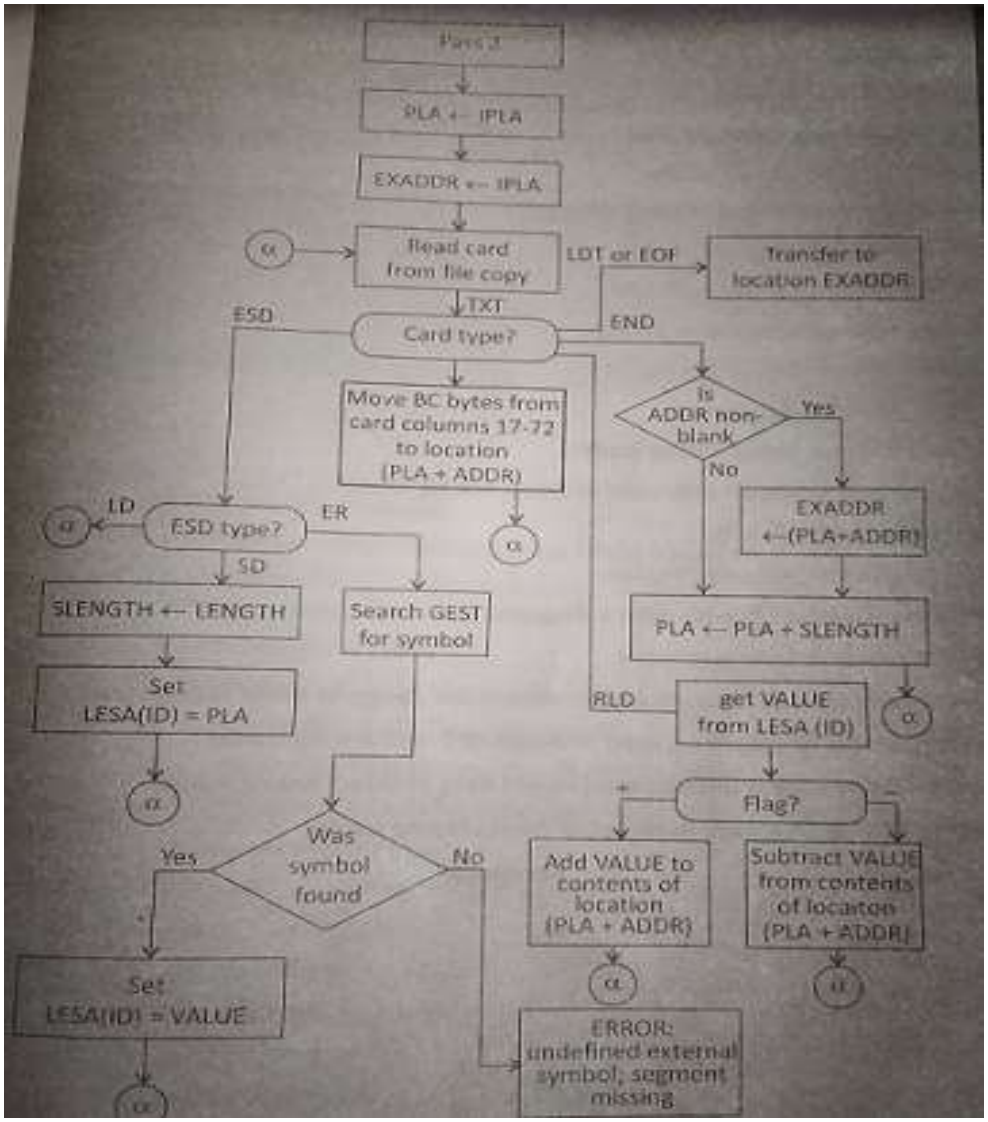
1. The length of object code segment.
2. A list of all the external symbol (could be used by other segment).
3. A list of all the external symbol (used by the segment)
4. Information about address constants.
5. The machine code translation of the source program.

The assembler produces four types of cards in the object deck

- **External Symbol Dictionary(ESD):** It contains information about all symbols that are defined in the program but referred somewhere.
- **The Text Cards(TXT):** It contains the actual object code translated version of the source program.
- **The Relocation and Linkage Directory(RLD):** Contain information about locations in the program whose contents depend on the address at which the program is placed.
- **The END card**: Indicates the end of the object deck and specifies the starting address for execution.

3. Draw a flow chart for Pass2 of Loader.

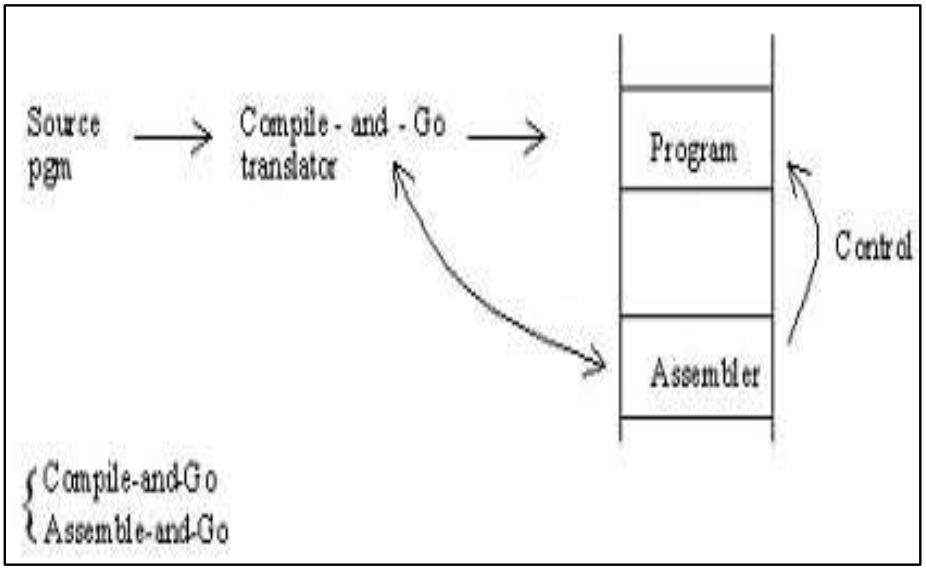
Detailed Pass2 flowchart of Loader:



4. Explain compile-and-go loader with a diagram.

Method of performing the loader functions is to have the assembler run in one part of memory and place the assembled machine instructions and data, as they are assembled, directly into their assigned memory locations. When the assembly is completed, the assembler causes a transfer to the starting instruction of the program. This is simple solution, involving no extra procedures. It is used by the complier and several other language processors.

It is relatively easy to implement. The assembler simply places the code into core, and the "loader" consists of one instruction that transfers to the starting instruction of the newly assembled program is called **"compile-and-go or assemble-and-go".**



Advantages:

1. It is relatively easy to implement.
2. It is simple and easy to follow.

Disadvantages:

1. A portion of the memory is wasted.
2. It is necessary to retranslate the program every time.
3. It is very difficult to handle multiple segments, specifically when they are in different languages, so it is very difficult to produce modular programs.

5. Explain the databases used by Pass1 and Pass2 of a direct linking loader.

Pass 1 Data Bases:

1. Input object decks.
2. A parameter, the Initial Program Load Address(IPLA) supplied by the programmer or OS, that specifies the address to load the first segment.
3. A Program Load Address (PLA) counter, used to keep track of each segment's assigned location.
4. A table, the Global External Symbol Table(GEST), that is used to store each external symbol and its corresponding assigned core address.
5. A copy of the input to be used later by pass 2. this may be stored on an auxiliary storage device, such as magnetic tape, disk, drum or the original object decks may be reread by the loader a second time for pass 2.
6. A printed listing, the load map, that specifies each external symbol and its assigned value.

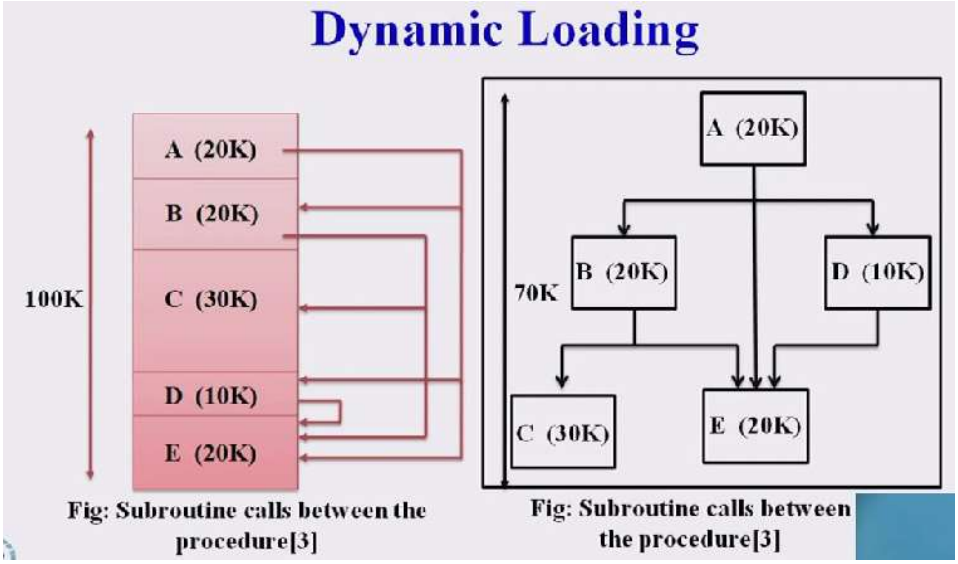
Pass 2 Data Bases:

1. Copy of object programs inputted to pass 1.
2. The Initial Program Load Address parameter (IPLA).
3. The Program Load Address Counter(PLA).
4. The Global External Symbol Table(GEST), prepared by pass 1, containing each external symbol and its corresponding absolute address value.
5. An array, the Local External Symbol Array(LESA), which is used to establish a correspondence between the ESD ID numbers, used on ESD and RLD cards, and the corresponding external symbol's absolute address value.

6. Explain dynamic loading.

In each of the loader schemes we have assumed that all of the subroutines needed are loaded into the memory at the same time. If the total amount of memory required by all these subroutines exceeds the amount available, then there is trouble. Then we use dynamic loading schemes to solve this problem.

Dynamic loading is also called load-on-call. This is done to save memory. If all the subroutines are loaded simultaneously, a lot of space is taken up. But only one is used at a time. So, here only the required subroutines are loaded. To identify the call sequence, we use a data structure called OVERLAY STRUCTURE. It defines mutually exclusive subroutines. So, only the ones needed are loaded and a lot of memory is saved. In order for the overlay to work, it is necessary for the module loader to load the various subroutines as they are needed.



7. Describe four types of cards used in direct linking loader.

Four types of cards used in direct linking loader:

1. **External Symbol Dictionary(ESD):** It contains information necessary to build the external symbol dictionary or symbol table. External symbols are symbols that can be referred beyond the subroutine level. The normal labels in the source program are used only by the assembler and information about them if not included in the object deck.

   There are 3 types of external symbol:

   1. Segment Definition (SD) name on START or CSECT card.
   2. Local Definition(LD) specified on ENTRY card.
   3. External Reference(ER) specified on EXTRN card.

2. **The Text Cards(TXT):** It contains the block of data and the relative address at which the data is to be placed. Once the loader has decided where to load the program, it merely adds the Program Load Address(PLA) to the relative address and moves the data into the resulting location. The data on the TXT card may be instructions, nonrelocated data, or initial values of address constants.
3. **The Relocation and Linkage Directory(RLD):** Contain information about
   1. The location and length of each address constants that needs to be changed for relocation or linking.
   2. The external symbol by which the address constant should be modified.
   3. The operation to be performed.
4. **The END card**: Indicates the end of the object deck and specifies the start of execution point for entire program. This address is recorded on the END

card. There is a final card required to specify the end of a collection of object decks. The 360 loaders usually use either a Loader Terminate(LDT) or End of File (EOF).

8. Write a short note on Relocating loaders.

Relocating Loader:

- Loaders that allow for program relocation are called ***relocating loaders*** or ***relative loaders***.
- Two methods for specifying relocation as part of the object program

Transfer Vector (Modification records)

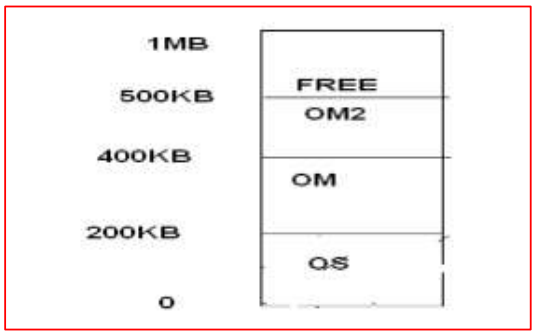
- Suitable for a ***small*** number of relocations required
- When relative or immediate addressing modes are extensively used

Relocation bits

- Suitable for a ***large*** number of relocations required
- When only direct addressing mode can be used in a machine with fixed instruction format (e.g., the standard SIC machine)

Example:

In the main memory, already OS occupies the memory from 0 to 200KB.Now the object module OM has to be placed over the OS from 200KB onwards. But OM has been placed from 0 to 200KB.Now there is an overlap, this leads to problems and confusions. The module OM has to be addressed from 200KB to 400KB.So the starting address of OM is 200.This is called relocation.



Advantages:

- Relocation bits are used to solve the problem of relocation , transfer vector is used to solve the problem of linking & program length information to solve allocation.

Disadvantages:

1. Not suited for loading external data. Transfer vector increases the size.
2. Does not facilitate access to data segments that can be shared
3. It is necessary to allocate, relocate, link and load all the subroutines each time, in order to execute a program.

9. Define binder. What are the classes of binders? Explain.

Binders: It is a program similar to direct-linking loader in **binding** subroutine together, but rather than placing the relocated and linked text directory into memory, it outputs the text as a file or card deck. This output file is in a format ready to be loaded and typically called a load module.

The binder essentially performs the functions of allocation, relocation and linking the module loader merely the function of loading.

There are two major classes of binders:

1. **Core Image Builder**: the simplest type produces a load module that looks very much like a single absolute loader deck. This means that the specific core allocation of the program is performed at the time that the subroutines are bound together. Since this kind of module looks like an actual "snapshot of image" of a section of core, it is called core image module and the corresponding binder is called a core image builder.
2. **Linkage Editor**: A more sophisticated binder, called a linkage editor, can keep track of the relocation information so that the resulting load module, as an ensemble, can be further relocated and thereby loaded anywhere in core. In this case the module loader must perform additional allocation and relocation s=as well as loading, but it does not have to worry about the complex problems of linking.

10. Explain the use of EXTERN and ENTRY statements.

ENTRY and EXTRN statements Consider an application program AP consisting of a set of programs unit's SP = {P (i)}. A program unit P (i) interacts with another program unit P (j) using addresses of P (j)'s instructions and data in its own instructions. To realize such interactions, P (j) and P (i) must contain public definitions and external references as defined in the following.

Public definition: a symbol pub_symb defined in a program unit which may be referenced in other program units. This is denoted with the keyword ENTRY and in the e.g. it is TOTAL.

External reference: a reference to a symbol ext_symb which is not defined in the program unit containing the reference. This is denoted with the keyword EXTRN and in the e.g. it is MAX and ALPHA.

These are collectively called subroutine linkages. They link a subroutine which is defined in another program.

EXTRN: EXTRN followed by a set of symbols indicates that they are definedin other programs, but referred in the present program.

ENTRY: It indicates that symbols that are defined in the present program are referred in other programs.

UNIT – 5

1. What are the functions of analysis and synthesis phases of compiler.

Analysis phase reads the source program and splits it into multiple tokens and constructs the **intermediate representation** of the source program.

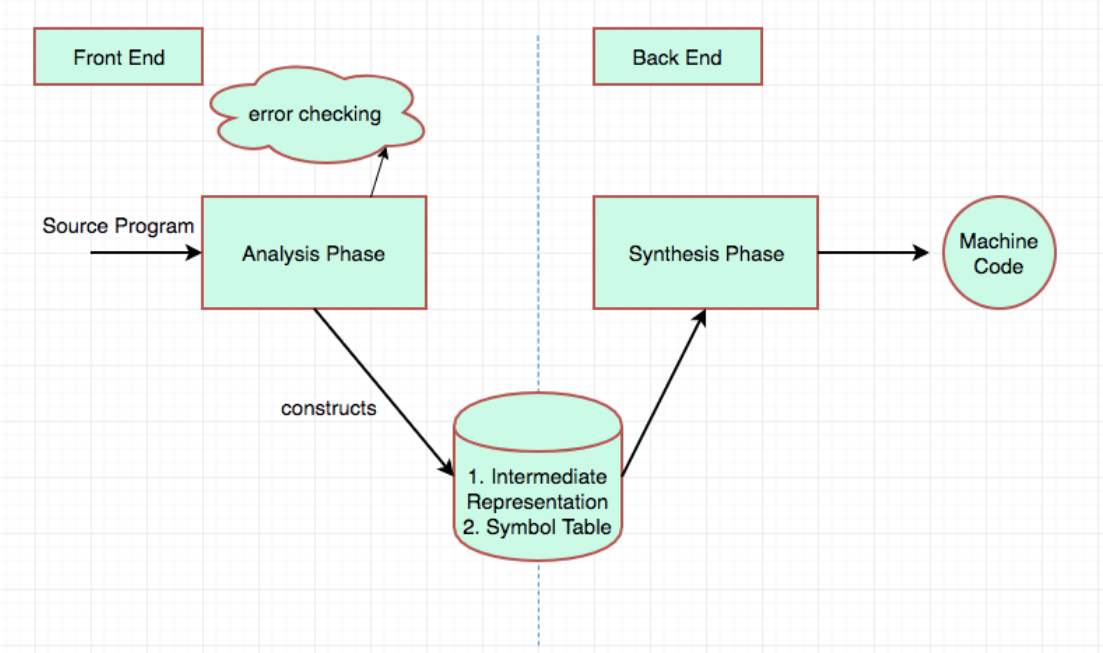
And also checks and indicates the syntax and semantic **errors** of a source program.

It collects information about the source program and prepares the **symbol table**. Symbol table will be used all over the compilation process.

This is also called as the **front end** of a compiler.

Synthesis phase: It will get the analysis phase input (intermediate representation and symbol table) and produces the targeted **machine level code.**

This is also called as the **back end** of a compiler.



A compiler can have many phases and passes.

- **Pass** : A pass refers to the traversal of a compiler through the entire program.
- **Phase** : A phase of a compiler is a distinguishable stage, which takes input from the previous stage, processes and yields output that can be used as input for the next stage. A pass can have more than one phase.

2. Write a note on assembly phase of compiler.

The process of generating the actual code is known as **Assembly phase**.

1. The code generation phase is producing assembly language, the compiler is generating references to actual code in place of literals and variables assigned by a storage allocation routine.
2. The assembly phase must resolve the label references in the object program, format the appropriate information for the loader. If the code generator simply generates symbolic machine names instructions and labels, the assembly phase must

- Resolve label references.
- Generate binary machine instruction.
- Convert literals.
- Calculate addresses.
- Generate storage.

3. Explain intermediate form of compiler.

The process of generating the object code for each construction after determining the syntactic construction is known as **Intermediate form.**

Advantages of Intermediate form:

- This facilitates optimization of object code.
- It allows logical separation between the machine-independent phase and machine-dependent phase.

The Intermediate form depends on syntactic construct such as:

1. Arithmetic statements.
2. Non-Arithmetic statements.
3. Non-Executable statements.

Arithmetic statements: One intermediate form of the arithmetic statement is a **parse tree.**

The rules for converting arithmetic statement into a parse tree are:

1. Any variable is a terminal node of a tree.

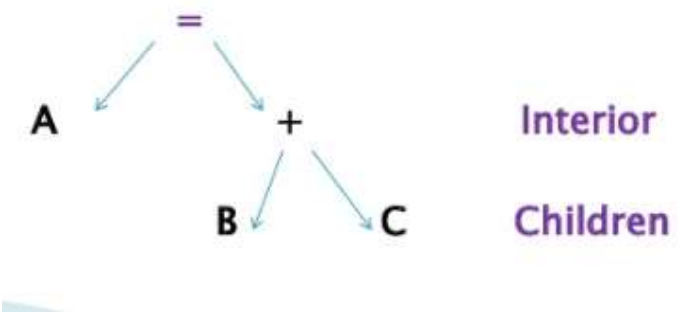
2. For every operand, construct a binary tree whose left branch is the tree for operand1 and whose right branch is for opernad2.

Use algebra rules to construct the tree:

1. Highest priority is given to the expression in the ().
2. * and / are having second priority.
3. + and – are having third priority.
4. If the sequence of the operators is same, then start solving from left to right.

Consider the ex. A= B+C

Then parse tree is



Non-Arithmetic statements: The statements DO, IF, GOTO etc. can all replaced by a sequentially ordering of individual matrix entries.

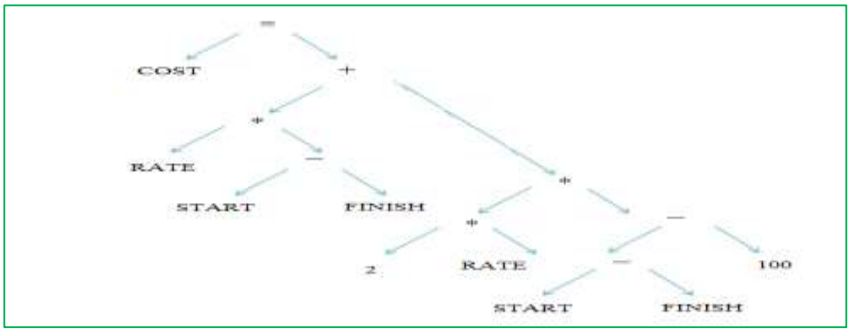
	Matrix		
	Operator	**Operand 1**	**Operand 2**
RETURN (COST);	**Return**	**COST**	
END;	**END**		

Non-Executable statements:

1. It declares the compiler information that clarifies the referencing or allocation of variables and associated storage.
2. There is no intermediate form of the statement, but the information contained in a non executable statement is entered into tables, to be used by other parts of the compiler.

Ex: DECLARE (COST, RATE, START, FINISH) FIXED BINARY (31) STATIC;

Fixed binary datatype, 31 bits length and static storage.



4. Discuss briefly about lexical phase of compiler.

Lexical Analysis: Recognition of basic elements or tokens and uniform symbols.

The three tasks of the lexical phase are:

1. To parse the source program into the basic elements of the language.
2. To build a literal table and an identifier table.
3. To build a uniform symbol table.

Databases: It manipulates 5 databases.

1. **Source Program**: Original form of the program; appears to the compiler as a string of character.
2. **Terminal Table:** It is a permanent table. Each entry consists of **Terminal symbol** (arithmetic operators, keywords etc.,) **an Indication** of its classification (operator, break character) and **Precedence** (used in later phases).

SYMBOL	INDICATOR	PRECEDENCE

3. **Literal Table:** It is created by lexical analysis to describe all literals used in the source program.

   There is one entry for each literal consisting of a value, a number of attributes, an address denoting the location of the literal at execution time, and other information.

LITERAL	BASE	SCALE	PRECISION	OTHER INFORMATION	ADDRESS

4. **Identifier Table:** It is created by lexical analysis to describe all identifiers used in the source program.

   There is one entry for each identifier and places the name of the identifier into that entry. Since in many languages identifiers may be from 1 to 31 symbols long. The pointer points to the name in a table for efficiency storage.

NAME	DATA ATTRIBUTES	ADDRESS

5. **Uniform Symbol Table:** It is created by lexical analysis to represent the program as a string of tokens rather than individual characters.

   There is one uniform symbol for every token in the program.

   Each uniform symbol contains the identification of the table of which the token is a member and its index within that table.

TABLE	INDEX

5. Explain the structure of a compiler with a neat diagram.

Structure of Compiler:

In analysing the compilation process, there are 7 distinct logical problems as follows:

1. **Lexical Analysis**: Recognition of basic elements or tokens and uniform symbols.
2. **Syntax Analysis**: Recognition of basic Syntactic construct through reductions.
3. **Interpretation**: Definition of exact meaning, creation of matrix and tables.
4. **Machine Independent Optimization**: Creation of more optimal matrix.
5. **Storage Assignment**: Modification of identifier and literal tables.
6. **Code Generation**: A macro processor is used to produce more optimal assembly code.
7. **Assembly and Output**: Resolving symbolic addresses and generating the machine language.

   Phase 1 to 4 is machine-independent and language dependent.

   Phase 5 to 7 is machine-dependent and language independent.

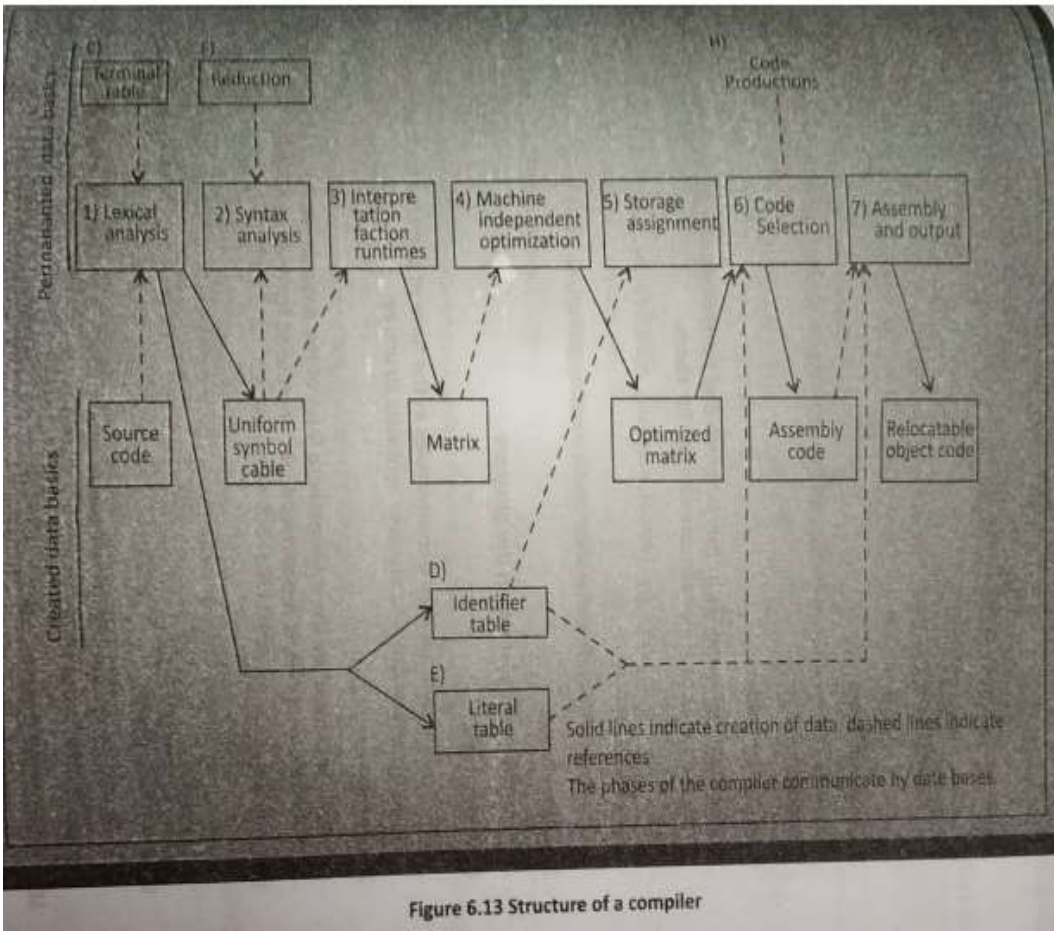


Figure 6.13 Structure of a compiler

The following are the database used by the compiler.

- ✓ **Source code**: The user program is the source code.
- ✓ **Uniform symbol table**: It consists of full or partial list of the tokens.
- ✓ **Terminal table**: It lists all keywords and special symbols.
- ✓ **Identifier table**: It contains all variable in the program.
- ✓ **Literal table**: It consists of constants in the program.
- ✓ **Reductions**: It is a permanent table of decision rules in the form of pattern for matching with the uniform symbol table.
- ✓ **Matrix**: It is created by the intermediate form of the program which is in turn created by the action routines.
- ✓ **Code productions**: It is a permanent table of definitions. There is one entry defining code for each matrix operator.
- ✓ **Assembly code**: Assembly language version of the program which is created by the code generation phase and is input to the assembly phase.
- ✓ **Relocatable object code**: The final output of the assembly phase, ready to be used as input to loader.

6. Explain the passes of compiler with neat diagram.

Pass1: Corresponds to the lexical analysis of a compiler. It scans the source program and creates the identifiers, literals and uniform symbol tables.

Pass2: Corresponds to the syntactic and interpretation phases. Pass2 scans the uniform symbol table produces the matrix and place the information about identifiers into the identifier table.

Pass3 through N-3: They correspond to the optimization phase. Each separate type of optimization may require passes over the matrix.

Pass N-2: It corresponds to the storage assignment phase. This is a pass over the identifier and literal tables rather than the program itself.

Pass N-1: It corresponds to code generation phase. It scans the matrix.

Pass N: It corresponds to assembly phase. It resolves symbolic address and creates information for the loader.

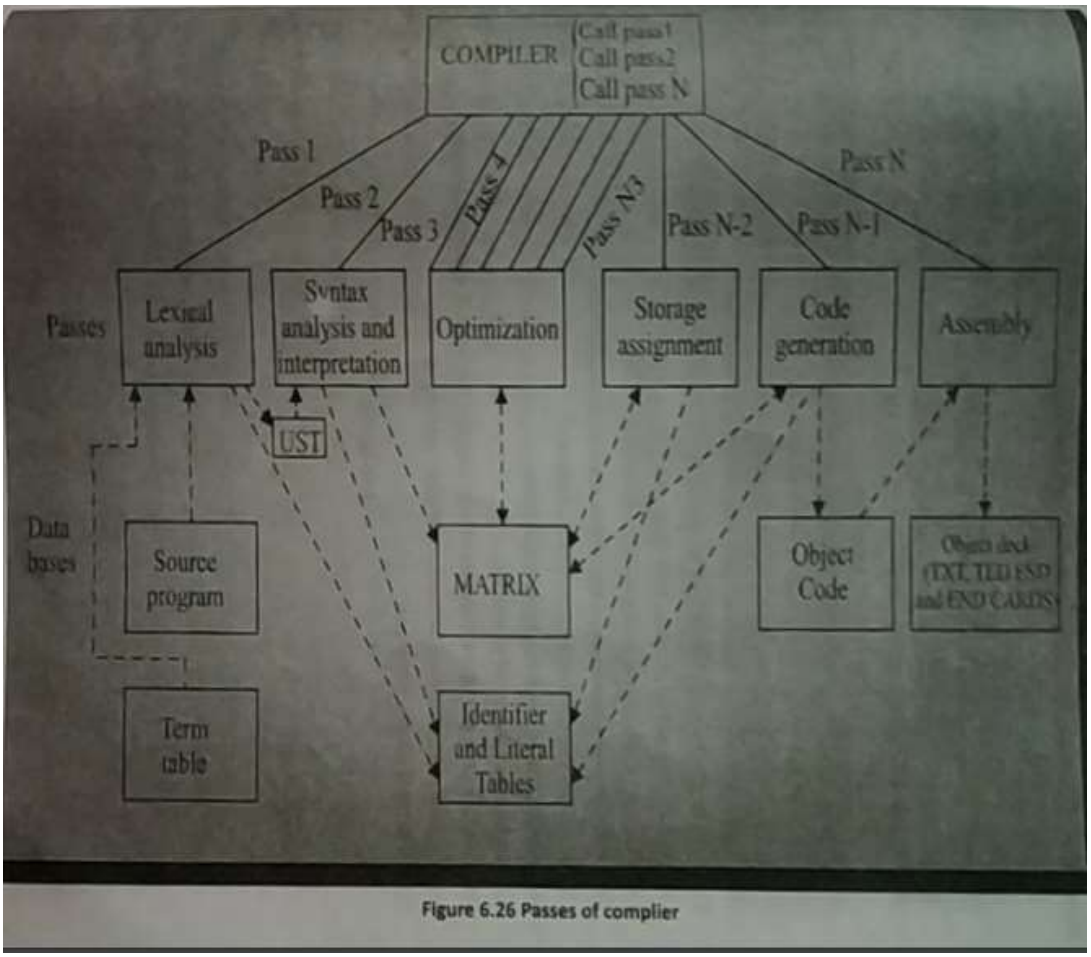


Figure 6.26 Passes of complier

7. Explain Machine independent optimisation.

Removing or deleting the duplicate entries in the matrix and modifying all references to the deleted entries is one type of machine independent optimization.

Other machine independent optimization are:

1. Compile time computation of operations, both of whose operands are constants.
2. Movement of computations involving non varying operands out of loops.
3. Use of the properties of Boolean expression to minimize their computation.

Ex: When a subexpression occurs in the same statement more than once, the duplicate entries can be deleted in the matrix and can be modified as shown in the table.

Matrix with common subexpression	Matrix after elimination of common subexpression
1 - START FINISH	1 - START FINISH
2 * RATE M1	2 * RATE M1
3 * 2 RATE	3 * 2 RATE
4 - START FINISH	4
5 - M4 100	5 - M4 100
6 - M3 M5	6 - M3 M5
7 - M2 M6	7 - M2 M6
8 = COST M7	8 = COST M7

Elimination of common subexpression

8. Explain syntax phase of a compiler.

Syntax Phase: The process of recognizing and separating the syntactic construct and interpreting their meaning is known as Syntax Analysis.

- ❑ Each syntactic construct is associated with a defined meaning which is in the form of actual code or intermediate form of the construction.
- ❑ Syntactic constructs are recognized using two general methods:

1. **Reductions**: They are rules (reductions) that specify the syntax form of the source language.
2. **Action Routines**: After defining the syntactic construction by the Reductions, appropriate compile routines or action routines are executed when a construction is recognized.
   Ex:

COST=RATE*(START-FINISH)+2*RATE*(SART-FINISH-100);

The reduction will specify that any identifier is followed by an equal sign, then call the action routine "**ARITHMETIC_STM**".

1. Reductions are tested consecutively for match between Old Top of Stack field and the actual Top of Stack, until match is found.
2. When match is found, the action routines specified in the action field are executed in order from left to right.
3. When control returns to the syntax analyser, it modifies the Top of Stack field.
4. Step 1 is repeated starting with the reduction specified in the next reduction field.

*****

US – 646

VI Semester B.C.A. Examination, May 2017
(CBCS) (2016-17 and Onwards)
COMPUTER SCIENCE
BCA-603 : Cryptography and Network Security

Time : 3 Hours Max. Marks : 100

Instruction : *Answer* ***all*** *the Sections.*

SECTION – A

Answer **any ten** questions. **Each** question carries **two** marks : **(10×2=20)**

1. What is information security ?
2. What is data integrity ?
3. Who is cryptanalyst ?
4. Define symmetric key cryptography.
5. What is FIPS ?
6. What is permutation process in cryptography ?
7. What is co-prime ? Give examples.
8. What is integer factorization ?
9. Define stream cipher.
10. What is payload ?
11. What is a session ?
12. What is IPSec ?

SECTION – B

Answer **any five** questions. **Each** question carries **five** marks : **(5×5=25)**

13. Explain symmetric key encryption model with a neat diagram.
14. Explain various security mechanisms.
15. Explain Euclid's algorithm with example.
16. Explain transpositional Cipher with an example.

P.T.O.

17. Explain CBC mode of operation.

18. Explain digital signature process with a neat diagram.

19. Explain PGP services.

20. Compare SSL and TLS protocols.

SECTION – C

Answer **any three** questions. **Each** carries **fifteen** marks : **(3×15=45)**

21. a) Explain key elements of public key encryption. **8**

    b) Differentiate equality and congruence with examples. **7**

22. a) Draw the block diagram of DES algorithm. Explain briefly. **8**

    b) Write a short note on multiple DES. **7**

23. a) Explain Fermat's theorem of primality test. **7**

    b) Explain RSA algorithm with one example. **8**

24. a) Write a short note on Whirlpool hash function. **7**

    b) Explain Diffie-Helman key agreement. **8**

25. a) Write a short note on IKE. **7**

    b) Explain the modes of IPSec. **8**

SECTION – D

Answer **any one** question. **Each** question carries **ten** marks : **(1×10=10)**

26. Explain one round of processing in AES.

27. Explain SHA-512 algorithm with a neat diagram.

SM – 625

VI Semester B.C.A. Examination, May/June 2018
(CBCS) (F + R)
(2016-17 and Onwards)
COMPUTER SCIENCE
BCA – 603 : Cryptography and Network Security

Time : 3 Hours Max. Marks : 100

***Instruction** : Answer **all** the Sections.*

SECTION – A

Answer **any ten** questions. **Each** question carries **two** marks : **(10×2=20)**

1. What is cryptosystem ?
2. Define Hashing.
3. What are the basic properties of divisibility ?
4. Define cipher text with an example.
5. What is Brute Force attack ?
6. Write any two applications of RSA algorithm.
7. Define Encryption and Decryption.
8. What is Trapdoor one-way function ?
9. Explain Avalanche Effect.
10. What is message padding ?
11. Define digital signature.
12. What are the protocols used to provide IP security ?

SECTION – B

Answer **any five** questions. **Each** question carries **five** marks. **(5×5=25)**

13. Discuss the classification of security goals.
14. Find GCD (2740, 1760) using Euclidean Algorithm.

P.T.O.

15. Differentiate between block cipher and a stream cipher.

16. Explain caesar cipher with an example.

17. Explain Fermat's little theorem.

18. What is primality test ? Explain in brief.

19. Explain cipher Feedback Mode.

20. Explain the practical applications of watermarking.

SECTION – C

Answer **any three** questions. **Each** carries **fifteen** marks. **(3×15=45)**

21. a) Explain in detail the taxonomy of attacks with relation to security goals. **10**
    b) Discuss Extended Euclidean Algorithm. **5**

22. a) Explain steps in DES Algorithm. **10**
    b) Discuss any two modes of operations in DES. **5**

23. a) State and explain Chinese Remainder Theorem with an example. **10**
    b) Discuss different attacks on RSA. **5**

24. a) Explain digital signature process with its security mechanism. **10**
    b) Write a note on Kerberos. **5**

25. a) Explain Public Key Infrastructure (PKI) in detail. **10**
    b) Differenciate between MIME and S/MIME. **5**

SECTION – D

Answer **any one** question. **Each** question carries **ten** marks. **(1×10=10)**

26. Explain Diffie-Helman key exchange technique with an example. **10**

27. a) Explain SSL Handshake protocol action. **5**
    b) Write a note on PGP services. **5**

100384

No. of Printed Pages : **2**

GS-644

VI Semester B.C.A. Examination, May/June - 2019

COMPUTER SCIENCE

BCA 603 : CRYPTOGRAPHY AND NETWORK SECURITY

(CBCS) (F+R)(2016-17 & Onwards)

Time : 3 Hours Max. Marks : 100

Instructions : *Answer* ***all*** *the sections.*

SECTION - A

Answer **any ten** questions. Each question carries **two** marks. **10x2=20**

1. Define Cryptography.
2. Distinguish between active and passive attacks.
3. Define Integrity and Non-repudiation.
4. Find the GCD of 16 and 48.
5. Define Padding in block cipher.
6. Define Resedue class.
7. Estimate the block size of MD5.
8. Define S/MIME.
9. What is Kerberos ?
10. Define the Diffie - Hellman protocol.
11. List any 2 applications of X.509 certificate.
12. Define Hijacking.

SECTION - B

Answer **any five** questions. Each question carries **five** marks. **5x5=25**

13. Compare steganography and watermarking. **5**
14. State and explain the principles of public key cryptography. **5**
15. With a neat diagram explain the general structure of DES. **5**

P.T.O.

16. Explain Transposition cipher with an example. 5

17. State the important properties of public key encryption scheme. 5

18. Why SHA more secure than MD5 ? 5

19. Briefly explain the architecture of SSL. 5

20. Explain Tunnel mode of IPSec. 5

SECTION - C

Answer **any three** questions. Each question carries **fifteen** marks. **3x15=45**

21. (a) Briefly explain the model of conventional cryptosystem. 8

(b) Find det.A if $A=\begin{bmatrix} 9 & 0 & -2 \\ -3 & -5 & 2 \\ 2 & 0 & 6 \end{bmatrix}$ 7

22. (a) Explain the four stages of AES algorithm. 8

(b) Explain the rules of play fair cipher with an example. 7

23. (a) Explain the procedure for RSA cryptosystem. 10

(b) Differentiate between Symmetric and Asymmetric key Cryptography. 5

24. (a) Explain the working of Digital Signature with a neat diagram. 8

(b) How does PGP provide confidentiality and authentication service for e-mail ? Explain. 7

25. (a) List and explain the four protocols of SSL. 8

(b) Explain X.509 certificate. 7

SECTION - D

Answer **any one** question. Each question carries **ten** marks. **1x10=10**

26. Discuss in detail block cipher modes of operations. 10

27. List and explain the properties of Hash functions. 10

- o 0 o -

SOUNDARYA EDUCATIONAL TRUST (R)
SOUNDARYA INSTITUTE OF MANAGEMENT & SCIENCE
Soundarya Nagar, Sidedahalli, Hessaraghatta Main Road, Bangalore
Department of Computer Science

BCA- 603: Cryptography and Network Security

Answer the following each carries two marks

UNIT – 1

1. **What is information security?**

   Information security is the state of being protected against the unauthorized access or use of information. It is also referred as 'Infosec'.

   Example: Disclosure, disruption, modification or destruction of information.

2. **What is data integrity?**

   Data integrity refers to the accuracy and consistency of data stored in a database. It is a fundamental component of information security and must be imposed when sending data through a network. It can be achieved by error checking and correction protocols.

3. **Who is cryptanalyst?**

   Cryptanalyst is a person expert in analyzing and breaking codes and ciphers, also able to decrypt new pieces of cipher text without additional information. The idea for a cryptanalyst is to extract the secret key.

4. **What are the basic properties of divisibility?**

   If a=1, then a = ±1.

   - If a|b and b|a, then a = ±b.
   - If a | b and b | c, then a | c
   - If a|b and a|c, then a|(m*b + n*c) where m and n are arbitrary integers.

5. **What is Brute Force Attack?**

The attacker tries every possible key on a piece of cipher text until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success

6. **Define Cryptography.**

Cryptography is a basic building block in computer security. It originates from Greek where crypto means secret and graphy means writing, the art of transforming messages to make them secure and immune to attacks involving three distinct mechanisms: symmetric-key encipherment, asymmetric-key encipherment and hashing.

7. Distinguish between active and passive attacks

Active Attack: It is a network exploit in which a hacker attempts to make changes to data on the target or data en-route to the target.

Passive Attack: They are in the nature of eavesdropping on or monitoring of transmissions.

The goal of the opponent is to obtain information that is being transmitted.

8. Define integrity and non repudiation. (2018)

Integrity: It is designed to protect data from modification, insertion, deletion and replaying by adversary.

Non-repudiation: It prevents either sender or receiver from denying a transmitted message.

9. Define symmetric key cryptography.

In symmetric key the encryption and decryption keys are known both to the sender and receiver. It is a cryptographic algorithm that uses the same key for encryption and decryption.

10. What is FIPS?

Federal Information Processing Standards is the standard publication series of National Institute of Standard and Technology (NIST). This standard specifies the security requirements that will be satisfied by a cryptographic model.

11. What is permutation process in cryptography?

A permutation in cryptography is a method of bit-shuffling used to permute or transpose bits across S-boxes retaining diffusion while transposition.

12. What is co-prime? Give examples

In number theory, two integers a and b are said to be co-prime if the only positive integer that divide both of them is 1 i.e., the only common factor of two numbers is 1.The numerator and denominator of a reduced fraction are co-prime.

13. What is integer factorization?

It is a commonly used mathematical problem often used to secure public key encryption systems. Common practice is to use very large semi primes as the number securing encryption.

14. Define cipher text with an example.

Cipher text is also known as cryptogram means the encoded message resulting from an encryption.

Example: Plain text->pay more money

Cipher text->sdbpruhprqhb

15. Find the GCD of 18 and 48.

GCD(48,18)-> divide 48 by 18 to get q=2 and r=12

Then divide 18 by 12 to get q=1 and r=6

Then divide 12 by 6 to get remainder 0.

UNIT - 2

16. Define Padding in block cipher.

Padding is a way to take data that may or may not be a multiple of the block size for a cipher and extend it out so that it is. This is required for many block cipher modes as they require the data to be encrypted to be an exact multiple of the block size.

17. Define stream cipher

The way in which the plain text is processed such that it produces the input elements continuously and producing output as one element at a time is called as a stream cipher.

18. What is cryptosystem?

A system which converts plain text to cipher text or cipher text to plain text by the application of encryption/decryption algorithm is known as cryptosystem.

UNIT - 3

19. Write any two applications of RSA algorithm.

This is used regularly in >Web browsers>Chat applications >e-mail >VPN's

Communication that require securely sending data to servers.

20. Define encryption and decryption.

Encryption is the process of translating plain text data into an unintelligible cipher text.

Decryption is the process of converting cipher text back to understandable plain text.

21. Define Resedue class.

It is a complete set of integers that are congruent modulo for some positive integers. In modulo, there are exactly different resedue classes and corresponding to the possible resedues.

UNIT - 4

22. What is Kerberos?

Kerberos is an authentication protocol and at the same time at a KDC that has become very popular and they provide a centralized authentication server whose function is to authenticate users to servers and servers to users.

23. Define hashing.

Hashing is a method of cryptography that converts any form of data into a unique string of text. Any piece of data can be hashed and easy to calculate any hash for any given data.

24. What is a session?

The associations between two end points i.e., peer to peer communication in bit-torrent file sharing p2p session ends when the connection between two systems is terminated.

25. What is Trapdoor one way function?

It is a function that is easy to compute in one direction, yet difficult to compute in the opposite direction without special information called the 'trapdoor'.

26. Explain Avalanche Effect.

The Avalanche effect is the desirable property of cryptographic algorithms, typically block ciphers and cryptographic hash functions, wherein if an input is changed slightly, the output changes significantly. In case of high quality block ciphers, such a small change in either the key or the plain text should cause a drastic change in the cipher text.

27. What is message padding?

Padding is a way to take data that may or may not be a multiple of the block size for a cipher and extend it out so that it is. This is required for many block cipher modes as they require the data to be encrypted to be an exact multiple of the block size.

28. Estimate the block size of MD5.

First the MD5 algorithm divides the input in blocks of 512bits each. 64 bits are inserted at the end of the last block. These 64 bits are used to record the length of the original input. If last block is less than 512bits some extra bits are padded

29. Define the Diffie-Hellman protocol.

It is a method for two computer users to generate a shared private key with which there can then exchange information across an insecure channel.

30. List any 2 applications of X.509 certificate.

- Issuer name: this field identifies the CA that issued the certificate.
- Subject name: this field defines the entity that owns the public key stored in this certificate.
- Validity period: this field defines the earliest time and the latest time during which the certificate is valid.
- Signature: this field contains the digest of all other fields in the certificate encrypted by CA's private key.

31. What is payload?

The essential data that is being carried within a packet or other transmission unit. It is the bits that get delivered to the end user at the destination.

32. What is IPSec?

Internet protocol security uses cryptographic security services to protect communication over internet protocol networks. It supports network layer, peer authentication, data integrity, confidentiality and replaces protection.

33. Define digital signature.

A digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party. They were created in response to the rising need to verify information via electronic systems.

UNIT - 5

34. What are the protocols used to provide IP security?

- Authentication Header (AH)
- Encapsulating Security Payload(ESP)

35. Define S/MIME.

S/MIME-Secure/Multipurpose Internet Mail Extension is a security enhancement to the MIME internet email format standard, based on technology from RSA Data security. It is used to send digitally signed and encrypted messages.

36. Define Hijacking

Hijacking is a type of network security attack in which the attacker takes control of a communication.The most common type of **hijacking** is when malware infects your **computer** and redirects your web browser, homepage, or search engine to a malicious site or somewhere you don't want to be.

Answer the following each carries five marks

UNIT – 1

1. Explain various security mechanisms.

The various security mechanisms to provide security are as follows-

1. Encipherment:

This is hiding or covering of data which provides confidentiality. It is also used to complement other mechanisms to provide other services. Cryptography and Steganography are used for enciphering

2. Digital Integrity:

The data integrity mechanism appends to the data a short check value that has been created by a specific process from the data itself. Data integrity is preserved by comparing check value received to the check value generated.

3. Digital Signature:

A digital signature is a means by which the sender can electronically sign the data and the receiver can electronically verify the signature. Public and private keys can be used.

4. Authentication Exchange:

In this two entities exchange some messages to prove their identity to each .

5. Traffic Padding:

Traffic padding means inserting some bogus data into the data traffic to thwart the adversary's attempt to use the traffic analysis.

2. Explain Euclid's algorithm with example.

The *Euclidean algorithm* is an efficient method to compute the *greatest common divisor* of two integers. It was first published in Book VII of Euclid's *Elements* sometime around 300 BC.

We write gcd `(a, b) = d` to mean that d is the largest number that will divide both a and b. If gcd `(a, b) = 1` then we say that a and b are *co prime* or *relatively prime*. The gcd is sometimes called the *highest common factor* (hcf).

Example:

421 = 111 x 3 + 88	(larger number on left)
111 = 88 x 1 + 23	(shift left)
88 = 23 x 3 + 19	(note how 19 moves down the "diagonal")
23 = 19 x 1 + 4	
19 = 4 x 4 + 3	
4 = 3 x 1 + 1	(last non-zero remainder is 1)
3 = 1 x 3 + 0	

The last non-zero remainder is 1 and therefore gcd (421, 111) = 1

3. Discuss the classification of security goals.

- Confidentiality- Most common aspect of information security, an organization needs to guard against those malicious actions that endanger the confidentiality of its information
- Data confidentiality: Assures the private or confidentiality information is not made available or disclosed to unauthorized individuals.
- Privacy – Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.
- Integrity-Guarding against improper information modification or destruction
- Data integrity – Assures that information and programs are changed only in a specified and authorized manner.
- System integrity- Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of system.
- Availability- Ensuring timely and reliable access to and use of information.
- Authenticity _ the property of being genuine and being able to be verified and trusted confidence in the validity of a transmission, a message, or message originator.
- Accountability- The security goal that generates the requirement for actions of an entity to be traced uniquely to the entity.

4. Find GCD (2740, 1760) using Euclidean algorithm.

GCD of 2740 and 1760 is 20

Step1: Divide the larger number by the smaller one: 2740÷1760= 1+980;

Step 2: Divide the smaller number by the above operation' remainder: 1760÷980= 1+780;

Step 3: Divide the remainder from the step 1 by the remainder from the step2: 980÷780= 1+200;

Step 4: Divide the remainder from the step 2 by the remainder from the step3: 780÷200= 3+180;

Step 5: Divide the remainder from the step 3 by the remainder from the step4: 200÷180= 1+20;

Step 6: Divide the remainder from the step 4 by the remainder from the step5: 180÷20= 9+0;

At this step, the remainder is zero, so we stop: 20 is the number we were looking for, the last remainder that is not zero. This is the greatest common factor (divisor).

Greatest (highest) common factor (divisor): gcf, gcd (2740;1760) = 20 = 2^2 x 5;

q	R1	R2	r
1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180
1	200	180	20
9	180	20	0

GCD of 2740 and 1760 is 20

5. Compare steganography and watermarking.

The word steganography, the art of secret writing is derived from the Greek word steganos, meaning "covered" and graphy, meaning "to write. The modern version of steganography involves hiding information within files that contain digital pictures.

Basic components of steganosystems are:

- Cover text is an original unaltered message.
- Embedding is a process in which the sender Alice tries to hide a message by embedding it into a cover text.
- Stegotext – (stego-data-stego-object)
- Recovering process in which the receiver Bob tries to get using the key only not the cover text the hidden message in the stegotext.
- Security requirement is that a third person watching such a communication should not be able to find out whether the sender has been active and when in the sense that he really embedded a message in the cover text.

Watermarking techniques is to provide a proof of ownership of digital data by embedding copyright statements into video or image digital products. Or in other words watermarking is the process of hiding digital information in a carrier signal. Watermarking leaves the original file/image intact and recognizable. Its characteristics are readily detectable, unambiguous, robust and visible. Its classifications are:

- Based on visibility of watermarks
- Based on the content to be watermarked
- Digital wa

6. Explain the practical applications of watermarking.

In a broadcast monitoring system identifying data is added to the video/audio signal prior to transmission.

I. Two kinds of monitoring system exist

i. Passive Monitoring

- Recognize the content being broadcast
- Compares received signals against a database of known content
- Very expensive as large frames need to be compared

ii. Active Monitoring

- Rely on information that is broadcast along with the content
- Relatively easier to implement
- Identification information is easily to interpret

II. Automatic monitoring and tracking of copy-write material on WEB

- Automatic audit of radio transmissions
- Data augmentation- to add information for the benefit of the public
- Fingerprinting applications

UNIT-2

7. Explain transposition cipher with an example.

All the techniques examined so far involves the substitution of a cipher text symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

Example: To encipher the message “meet me after the party” with a rail fence of depth 2, we write the following:

m e m a t r h p r y

e t e f e t e a t

The encrypted message is mematrhpryetefeteat.

8. Explain Caesar cipher with an example.

The Caesar or shift cipher is earliest known use of a substitution cipher and the simplest

was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing 3 places further down the alphabet.

Example:

Plaintext: pay more money

Cipher text: sdb pruh prqhb

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Let us assign a numerical equivalent to each letter

Note that the alphabet is wrapped around, so that letter following ‘z’ is ‘a’.

For each plaintext letter p, substitute the cipher text letter c such that

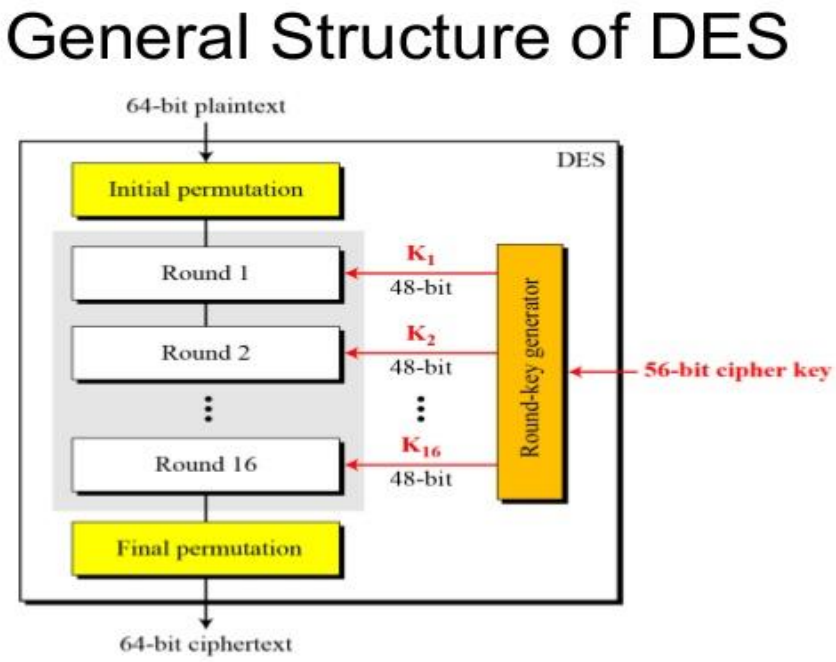
$C=E(p)=(p+3) \bmod 26$

A shift may be any amount, so that general Caesar algorithm is

$C=E(p)=(p+k) \bmod 26$

Where k takes on a value in the range 1 to 25 . The decryption algorithm is simply $P=D(C)=(C-k) \bmod 26$.

9. With a neat diagram explain the general structure of DES.



The encryption process is made of two permutations (p-boxes), which we call initial and final permutation and 16 fiestal rounds. Each round uses a different 48 bit round key generated from the

cipher key according to a predefined algorithm. The elements of DES cipher at the encryption site is shown in the below diagram. The Data Encryption System (DES) is a symmetric key block cipher published by the National Institute of Standards and Technology (NIST). At the encryption site, DES takes a 64bit plaintext and creates a 64 bit cipher text, at the decryptionsite DES takes a 64 bit cipher text and creates a 64 bit block of plaintext. The same 56 bit cipher key is used for both encryption and decryption.

10. Differentiate between block cipher and a stream cipher.

Block cipher	Stream cipher
• Block Cipher converts the plain text into cipher text by taking plain text's block at a time.	• Stream Cipher Converts the plain text into cipher text by taking 1 byte of plain text at a time.
• Block cipher uses either 64 bits or more than 64 bits.	• While stream cipher uses 8 bits
• The complexity of block cipher is simple.	• While stream cipher is more complex
• Block cipher Uses confusion as well as diffusion.	• While stream cipher uses only confusion
• In block cipher, reverse encrypted text is hard.	• While in stream cipher, reverse encrypted text is easy
• The algorithm modes which are used in block cipher are: ECB (Electronic Code Book) and CBC (Cipher Block Chaining).	• The algorithm modes which are used in stream cipher are: CFB (Cipher Feedback)

UNIT-3

11. Explain CBC mode of operation.

CBC mode of operation provides message dependence for generating ciphertext and makes the system non-deterministic.

The operation of CBC mode is depicted in the following illustration. The steps are as follows:

- Load the n-bit Initialization Vector (IV) in the top register.
- XOR the n-bit plaintext block with data value in top register.
- Encrypt the result of XOR operation with underlying block cipher with key K.
- Feed ciphertext block into top register and continue the operation till all plaintext blocks are processed.
- For decryption, IV data is XORed with first ciphertext block decrypted. The first ciphertext block is also fed into to register replacing IV for decrypting next ciphertext block.

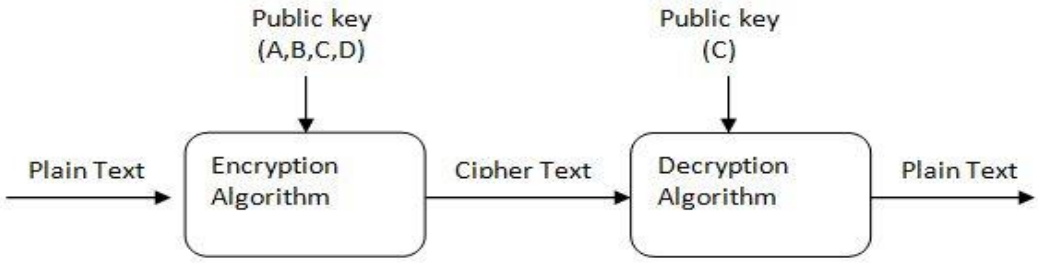
12. State the important properties of public key encryption scheme.

Characteristics of Public Encryption key:

- Public key Encryption is important because it is infeasible to determine the decryption key given only the knowledge of the cryptographic algorithm and encryption key.
- Either of the two key (Public and Private key) can be used for encryption with other key used for decryption.
- Due to Public key cryptosystem, public keys can be freely shared, allowing users an easy and convenient method for encrypting content and verifying digital signatures, and private keys can be kept secret, and ensuring only the owners of the private keys can decrypt content and create digital signatures.
- The most widely used public-key cryptosystem is RSA (Rivest–Shamir–Adleman). The difficulty of finding the prime factors of a composite number is the backbone of RSA.

Example:

Public keys of every user are present in the Public key Register. If B wants to send a confidential message to C, then B encrypts the message using C Public key. When C receives the message from B then C can decrypt it using its own Private Key. No other recipient other than C can decrypt the message because only C know C's private key.



13. What is primality test? Explain in brief.

A deterministic primality testing algorithm accepts an integer and always outputs a prime or composite.

Divisibility algorithm: The most elementary deterministic test for primality is the divisibility test. We use as divisors all numbers smaller that $\sqrt{n}$. If any of the numbers divides n, then n is composite. Algorithm shown below shows it is very inefficient form.

```
Divisibility (n)
{
        R ← 2
    while (r < √n)
    {
        If(r|n) return "a composite"
        R ← r+1
    }
  return "a prime"
}
```

14. Explain Fermat's little theorem.

Fermat's little theorem plays a very important role in number theory and cryptography. Two versions of the theorem are

First theorem: The first version says that if p is a prime and a is an integer such that p does not divide a, then

$$a^{p-1} \equiv 1 \bmod p$$

Second theorem: The second version removes the condition on a. It says that if p is a prime and a is an integer, then

$a^n \equiv a \bmod p$

Example:

I. To find result of $6^{10} \bmod 11$

We have $6^{10} \bmod 11=1$. This is first version where p=11

II. To find result of $3^{12} \bmod 11$

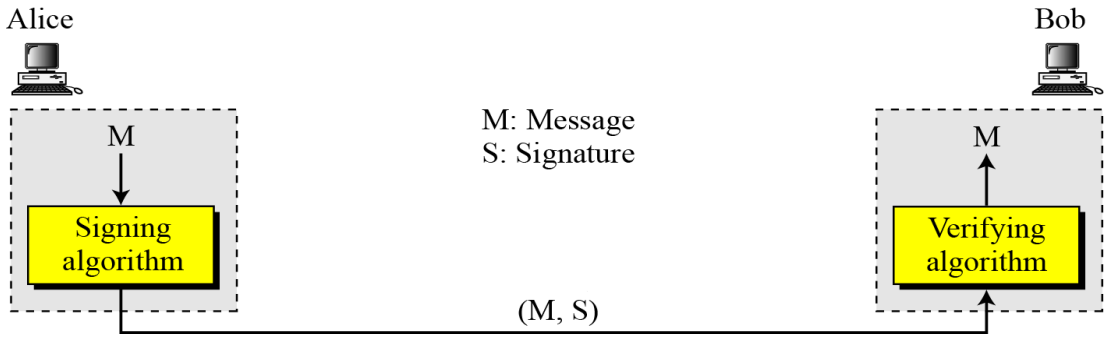
Here the exponent (12) and the modulus(11) are not the same. With substitution this can be solved using Fermat's little theorem.

UNIT- 4

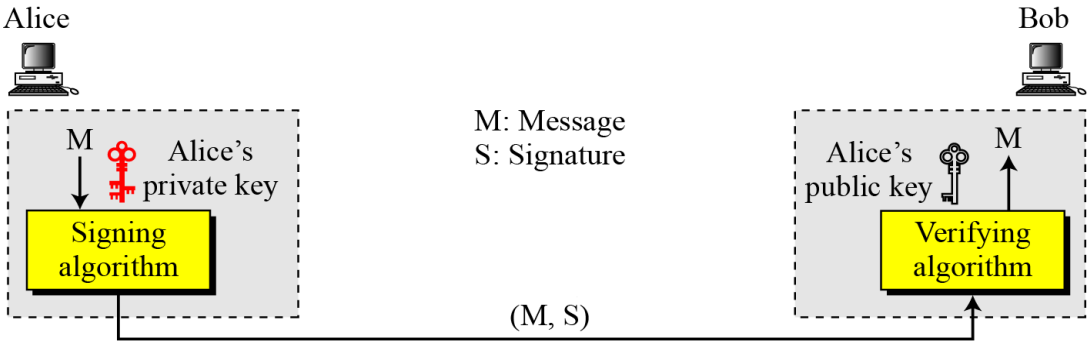
15. Explain digital signature process with a neat diagram.

The steps to process a digital signature are:-

- The sender uses a signing algorithm to sign the message
- The message and the signature are sent to the receiver
- The receiver receives the message and the signature and applies the verifying algorithm to the combination if the result is true, the message is accepted, and otherwise it is rejected.



- Need for keys: In a digital signature, the signer uses he/she uses her private key, applied to a signing algorithm, to sign the document, the verifier, on the other hand, uses the public key of the signer, applied to verifying algorithm, to verify the document. In digital signature s public and private keys as used in a crypto system for confidentiality. The sender uses the public key of the receiver to encrypt. The receivers use his/her own private key to decrypt.



16. Why SHA more secure than MD5?

- MD5 can create 128 bits long message digest while SHA1 generates 160 bits long message digest.
- To discern the original message the attacker would need 2^{128} operations while using the MD5 algorithm. On the other hand, in SHA1 it will be 2^{160} which makes it quite difficult to find.
- If the attacker wants to find the two messages having the same message digest, he would require 2^{64} operations for MD5 whereas 2^{80} for SHA1.
- When it comes to security by the above-given fact SHA1 hold more points relative to MD5.

MD5 is faster than SHA1, but SHA1 is more complex as compared to MD5.

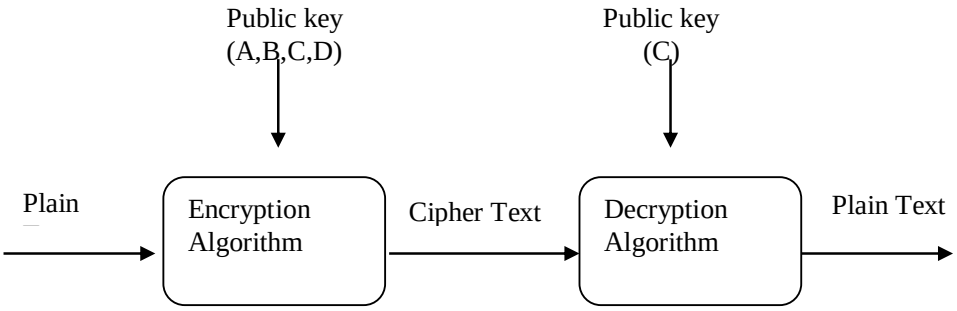
17. State and explain the principles of public key cryptography.

- Public key Encryption is important because it is infeasible to determine the decryption key given only the knowledge of the cryptographic algorithm and encryption key.
- Either of the two key (Public and Private key) can be used for encryption with other key used for decryption.
- Due to Public key cryptosystem, public keys can be freely shared, allowing users an easy and convenient method for encrypting content and verifying digital signatures, and private keys can be kept secret, and ensuring only the owners of the private keys can decrypt content and create digital signatures.
- The most widely used public-key cryptosystem is RSA (Rivest–Shamir–Adleman). The difficulty of finding the prime factors of a composite number is the backbone of RSA.

Example :

Public keys of every user are present in the Public key Register. If B wants to send a confidential message to C, then B encrypt the message using C Public key. When C receives

the message from B then C can decrypt it using its own Private key. No other recipient other than C can decrypt the message because only C know C's private key.



UNIT-5

18. Explain PGP services.

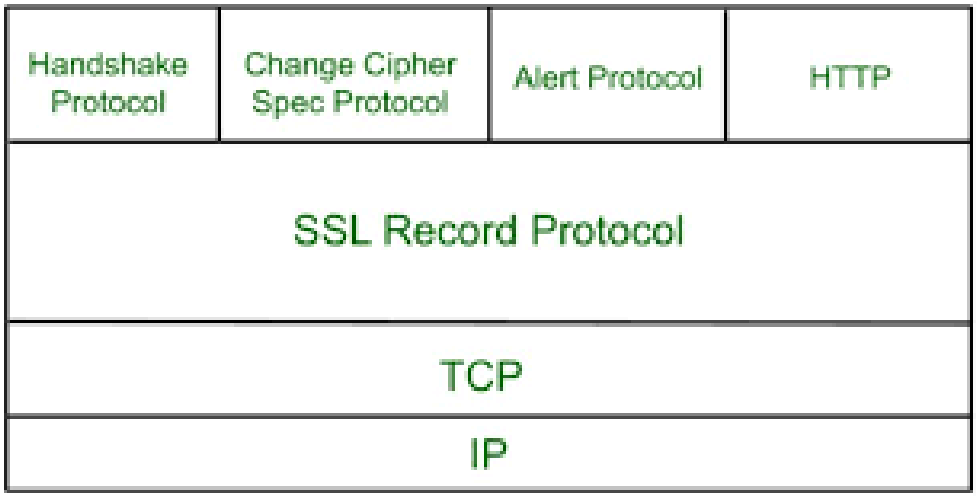
- PGP stands for Pretty Good Privacy (PGP) which is invented by Phil Zimmermann.
- Pretty Good Privacy (PGP) can be used to create a secure e-mail message or to store a file securely for future retrieval.
- PGP was designed to provide all four aspects of security, i.e., privacy, integrity, authentication, and non-repudiation in the sending of email.
- PGP uses a digital signature (a combination of hashing and public key encryption) to provide integrity, authentication, and non-repudiation.
- PGP is an open source and freely available software package for email security.
- PGP provides authentication through the use of Digital Signature.

19. Compare SSL and TLS protocols.

SSL Protocol	TLS Protocol
Secure Socket Layer protocol	Transport Layer Protocol
SSL network protocol allows data to be transferred privately security between a web server browser	TLS network protocol allows data to be transferred privately security between a web server browser
Prevents intruders from tampering	TLS will add latency to your sites traffic
Preventing intruders from passively listening	Hand shake is resource intensive
Secure your site using HTTPS (It is HTTP	TLS will add complexity to your server

protocol embedded within TLS protocol HTTP)	management uses HTTP, TLS connection will be significantly faster
SSL is older	TLS is modern encryption standard
SSL Version two or SSL V_2	TLS $V_1.0$
SSL V_3- Currently used version	TLS $V_2.1$
	TLS $V_2.2$
	$V_1.3$ currently using

20. Briefly explain the architecture of SSL.



SSL Architecture:

SSL is designed to provide security and compression services to data generated from the application layer:

Services provided by SSL are:

- Fragmentation: Each upper layer message is fragmented into blocks of 214 bytes (16384bytes) or less.
- Compression: Each fragment of data is compressed using one of the optionally applied. Compression must be lossless negotiated between the client and server.
- Message Integrity: To preserve the integrity of data, SSL uses keyed-hash function to create a MAC
- Confidentiality: To provide confidentiality, the original data and the MAC encrypted using symmetric encryption.

- Framing: A header is added to the encrypted payload. The pay load is then passed to a reliable transport layer protocol.

21. Explain tunnel mode of IPSec.

Tunnel mode provides protection to the entire IP packet. To achieve this, the header or trailer fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new "outer" IP packet with a new outer IP header.

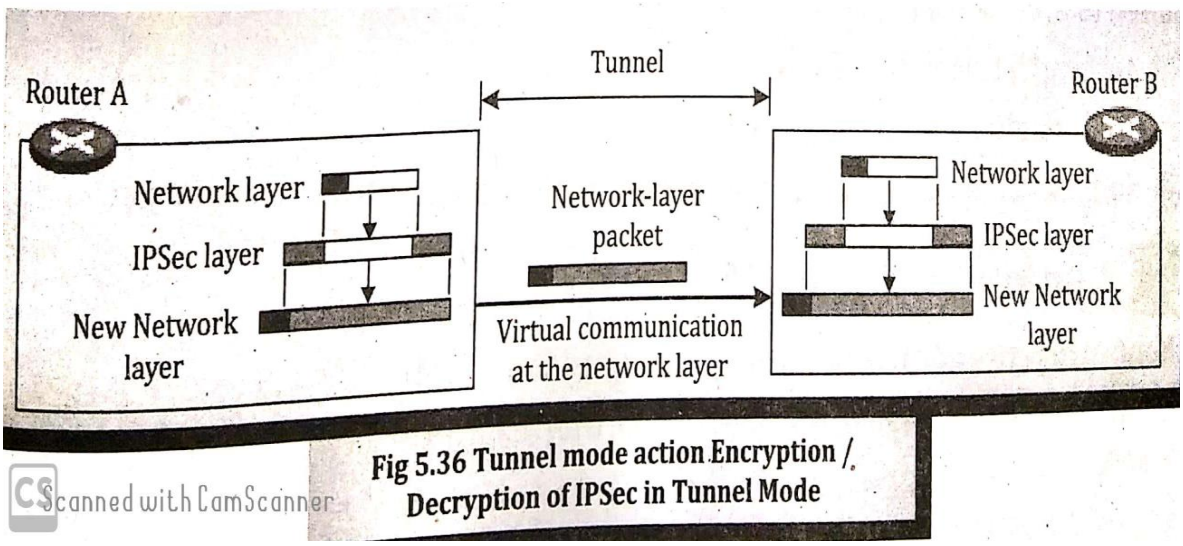


Fig 5.36 Tunnel mode action Encryption / Decryption of IPSec in Tunnel Mode

The entire original, or inner, packet travels through a "tunnel" from one point of an IP network to another; no routers along the way are able to examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding to the security. With the tunnel mode, a number of hosts on the networks behind firewalls may engage in secure communications without implementing IPSec.

Section C

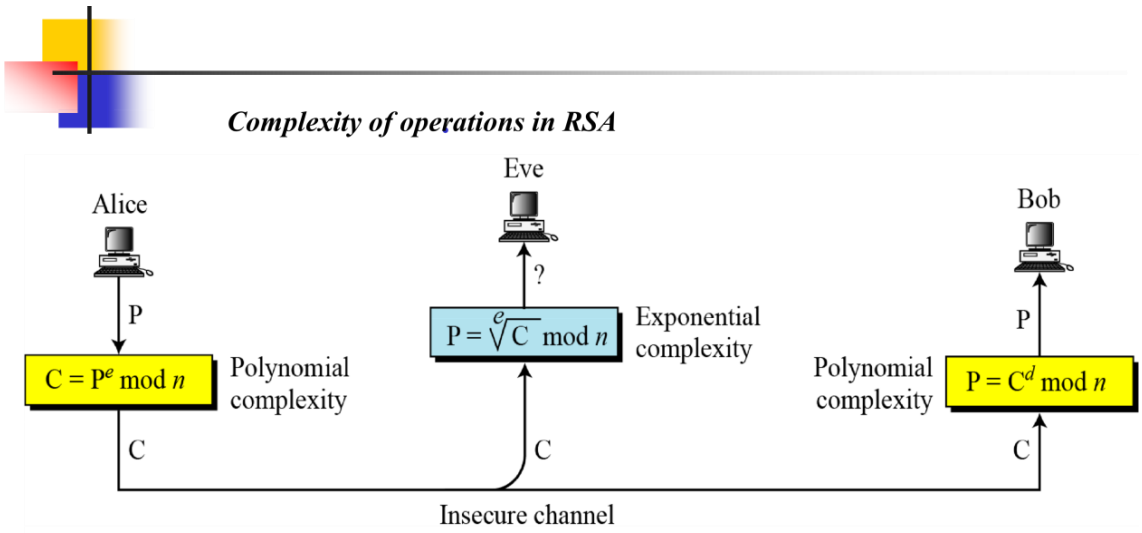
Answer the following each carries fifteen marks

1) a) Explain key elements of public key encryption. 2017 QP (unit-3)

The public key can be known by everyone and is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.

Public-key encryption is a system that uses two keys a *public key* known to everyone and a *private* or *secret key* known only to the recipient of the message. An important

element to the public key system is that the public and private keys are related in such a way that only the public key can be used to encrypt messages and only the corresponding private key can be used to decrypt them. In RSA encryption/decryption ring is public because the modulus n is public. Anyone can send message to Bob using his public key, encryption in RSA can be using the algorithm with polynomial time.



b) Differentiate equality and congruence with examples. (unit-1)

Equality: Two matrices are said to be equal if they have two same number of rows and columns and corresponding elements are equal.

In other words we can say

A=B if $a_{ij}=b_{ij}$ for all if i's and j's.

Congruence: This term is used instead of equality. Mapping of z to Z_n is not one to one. Infinite members of z can map to one member of Z_n.

For congruence operator we will use(≡).

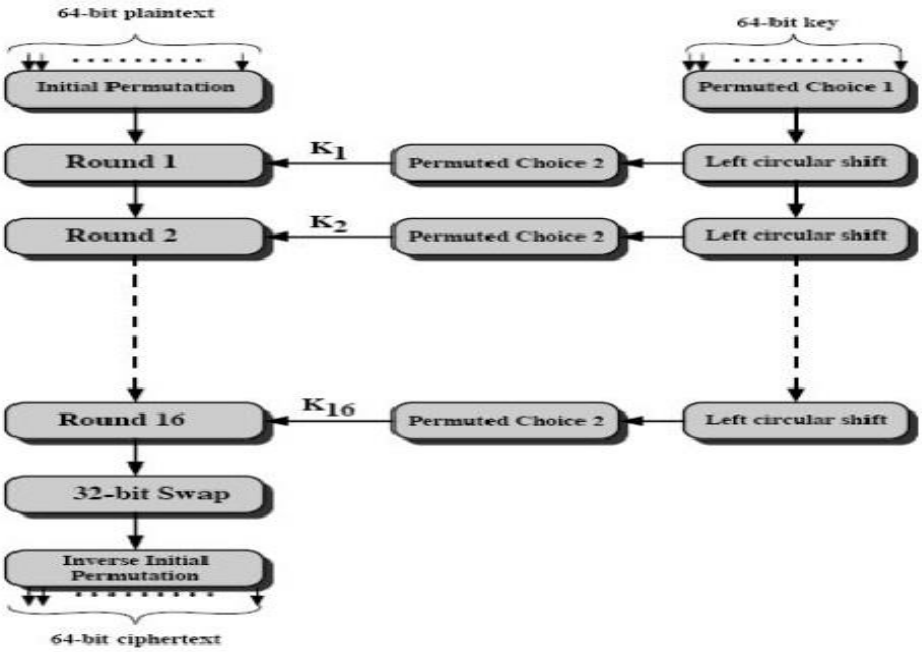
We add the phrase to the right side of modulus to define the value of modulus that makes the relationship valid.

Ex: 2≡12(mod10) 2=>remainder

2) a) Draw the block diagram of DES algorithm. Explain briefly. (unit-2)

The processing on the plaintext proceeds in three phases. First, the 64bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input. This is followed by a phase consisting of 16 rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round

consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the pre output. Finally, the pre output is passed through a permutation (IP^{-1}) that is the inverse of the initial permutation function, to produce the 64-bit cipher text. With the exception of the initial and final permutation function. Then, for each of the 16 rounds, a sub key (K_i) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different sub key is produced because of the repeated shifts of the key bits.



b) Write a short note on multiple DES. (unit-2)

Multiple DES-Conventional Encryption Algorithms

If a block cipher has a key size, which is small in context to the present day computation power, then a natural way out may be to perform multiple encryptions by the block cipher. As an example, consider the DES algorithm which has a key size of 56 bits, which is short in context to the modern computation capability. The threat is that such a key value can be evaluated by brute force key search. Hence two DES applications give what is known as 2-DES.

- DES and Meet in the Middle Attack

Consider a message m, which is to be encrypted. The corresponding block cipher for one application of the DES applications is represented by EK, where k is the corresponding DES key. The output of 2-DES is $c=EK_2(EK_1(m))$. To decrypt similarly, $m=Dk_1(DK_2(c))$. This

cipher, 2-DES should offer additional security, equivalent to both k_1 and k_2. The cipher2-DES obtained by the composition of two ciphers. Such an idea can similarly be extended to multiple ciphers. It may be noted that such a product on the DES cipher is expected to provide additional security, because DES does not form a group under the composition operation.

- Meet-in-the-middle(MIM) Attack and 3DES

There are in general two flavors of 3-DES.There are at least two flavors of implementation of 3-DES.

- The first implementation uses three keys, namely K1, K2, K3. The cipher text of m is thus obtained by $C=DES_{k1}[DES_{k2}(DES_{k3}(m))]$.
- The second way to implement 3-DES using two keys, thus $C=DES_{k1}[DES^{-1}_{k2}(DES_{k3}(m))]$ thus if the keys K1 and K2 are the same then we obtain a single DES. This backward compatibility of the two key version of 3-DES is the reason why the middle layer is a decryption. It has otherwise no security implications.

3) a) Explain Fermat's theorem of primality test. (unit-3)

Fermat's little theorem plays a very important role in number theory and cryptography. Two versions of the theorem are:

FIRST THEOREM

The first version says that if p is a prime and a is an integer such that p does not divide a, then $a^{p-1} \equiv 1 \bmod p$

SECOND THEOREM

The second version removes the condition on a. It says that if p is a prime and a is an integer, then $a^p \equiv a \bmod p$

Example:

i) Find the result of $6^{10} \bmod 11$

We have $6^{10} \bmod 11=1$. This is the first version of theorem where p=11

ii) Find the result of $3^{12} \bmod 11$

Here the exponent (12) and the modulus (11) are not the same. With substitution this can be solved using Fermat's little theorem.

$3^{12} \bmod 11= (3^{11} * 3) \bmod 11= (3^{11} \bmod 11)(3 \bmod 11) = (3*3) \bmod 11=9$

MULTIPLICATIVE INVERSE

If p is a prime and a s an integer such that p does not divide a(p|a), then $a^{-1} \bmod p=a^{p-2} \bmod$

p. This application eliminates the use of extended Euclidean algorithm for finding some multiplication inverses.

Example: $8^{-4} \bmod 17 = 8^{17-2} \bmod 17 = 8^{15} \bmod 17 = 15 \bmod 17$

b) Explain RSA algorithm with one example. (unit-3)

RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption.

The RSA algorithm holds the following features:

- RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
- The integers used by this method are sufficiently large making it difficult to solve.
- There are two sets of keys in this algorithm: private key and public key.

You will have to go through the following steps to work on RSA algorithm −

Step 1: Generate the RSA modulus

The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown −

$N=p*q$

Here, let N be the specified large number.

Step 2: Derived Number (e)

Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1

Step 3: Public key

The specified pair of numbers **n** and **e** forms the RSA public key and it is made public.

Step 4: Private Key

Private Key **d** is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows −

$ed = 1 \bmod (p-1)(q-1)$

The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

4) a) Write a short note on Whirlpool hash function. (unit-4)

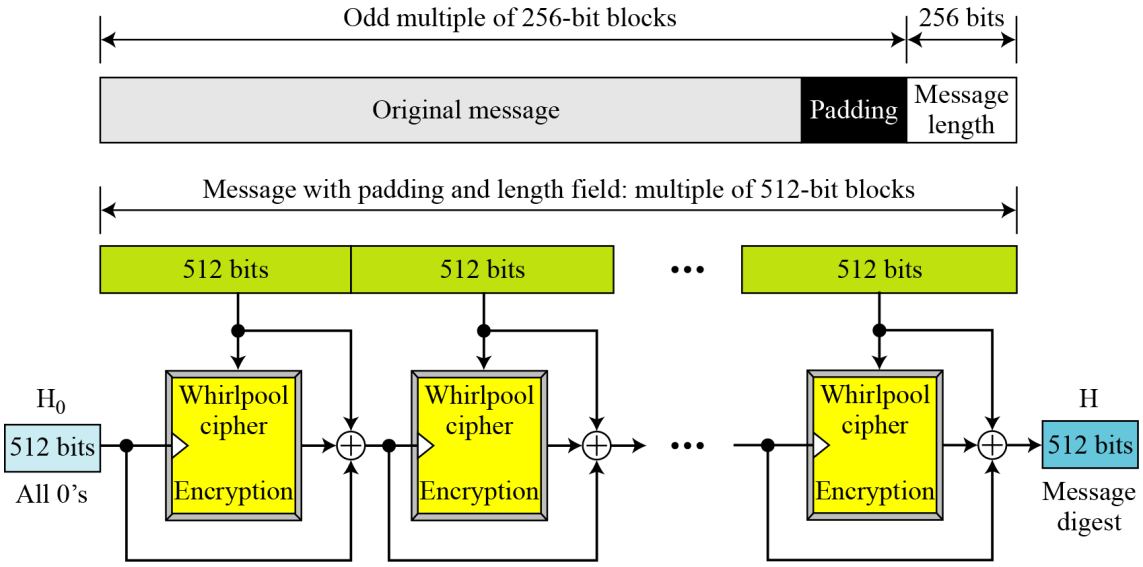
Whirlpool is a cryptographic hash function. It was designed by Vincent Rijmen and Paulo S.L.M Barreto, who first described it in 2000. The hash has been recommended by

the NESSIE project. It has also been adopted by the ISO and IEC as part of the joint whirlpool. The block cipher is a modified AES cipher that has been tailored for this purpose. This is a 512-bit hash function three versions of whirlpool have been released, they are

i) WHIRLPOOL-0

ii) WHIRLPOOL-T

iii) WHIRLPOOL



Steps to prepare the message:

- Whirlpool requires that the length of original message be less than 256bits.
- A message needs to be padded before being processed. The padding is a single 1-bit followed by the necessary numbers of 0-bits to make length of padding an odd multiple of 256 bits
- After padding a block of 256 bits is added to define length of original message.
- After padding and adding the length field, the augmented message size is an even multiple of 256 bits or multiple of 512 bits.
- Whirlpool creates a digest of 512 bits from a multiple of 512 bit block message.
- H_0 is initialized to all 0's this becomes the cipher key for first block.
- The cipher text resulting from each block becomes the cipher key for next block after being XO red with previous cipher key and plain text block.
- The message is the final 512-bit cipher text after the last XOR operation.

b) Explain the Diffie Helman key agreement. (unit-4)

In Diffie Helman protocol, two parties create a symmetric session key without the need of a KDC.

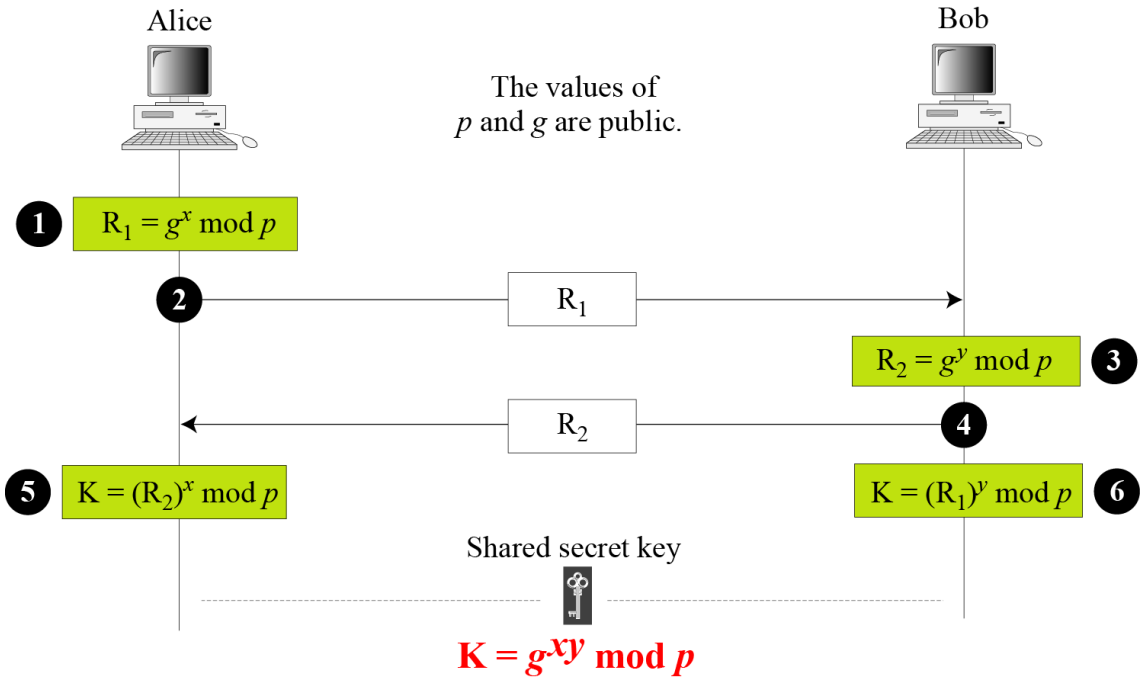
- Before establishing a symmetric key, the two parties need to choose two numbers:

p→A large prime number on the order of 300 decimal digits(1024 bits)

g→A generator of order p-1 in the group<Zp*.x>

- These p and g do not need to be confidential

They can be sent through the Internet; can be public.



The following steps are as follows:

1. Alice choose a large random number x such that $0 \leq x \leq (p-1)$ and calculates $R_1=g_x \bmod p$.
2. Bob chooses another large random number y such that $0 \leq y \leq (p-1)$ and calculates $R_2=g_y \bmod p$.
3. Alice sends R_1 to Bob (but not sends the value of x)
4. Bob sends R_2 to Alice (but not sends the value of y)
5. Alice calculate $K=(R_2)^x \bmod p$
6. Bob calculate $K=(R_1)^y \bmod p$

Where K is the symmetric key for the session

- Bob has calculate K as:

$R_1=g^x \bmod p$

$K=(R_1)^y \bmod p=(g^x \bmod p)y \bmod p=g^{xy} \bmod p$

- Alice has calculated K as:

$R_2=g^y \bmod p$

$K=(R_2)^x \bmod p=(g^y \bmod p)x \bmod p=g^{xy} \bmod p$

Both have reached the same value without Bob knowing the value of x and without Alice knowing the value of y.

The symmetric (shared) key in the Diffie-Hellman method is $K = g^{xy} \bmod p$.

5) a) Write a short note on IKE. (unit-5)

Internet key exchange: The key management portion of IPSec involves the determination and distribution of secret keys. A typical requirement is four keys for communication between two applications: transmit and receive pairs for both integrity and confidentiality.

The IPSec Architecture document mandates support for two types of key management:

- Manual: A system administrator manually configures each system with its own keys and with the keys of other communicating systems.
- Automated: An automated system enables the on-demand creation of keys for SAS and facilities the use of keys in a large distributed system with an evolving configuration. The Internet Key Exchange (IKE) is a protocol designed to create both inbound and outbound security associations.

The IKE is a complex protocol based on three other protocol :

- Oakley key determination protocol
- Secure key exchange mechanism(SKEME)
- Internet security association and key management protocol(ISAKMP)

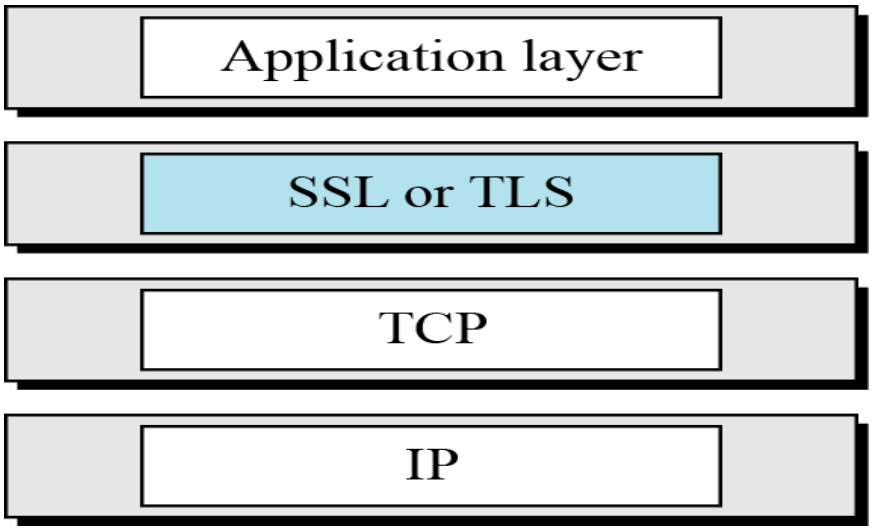
b) Explain the modes of IPSec. (unit-5)

The types of IPSec are:

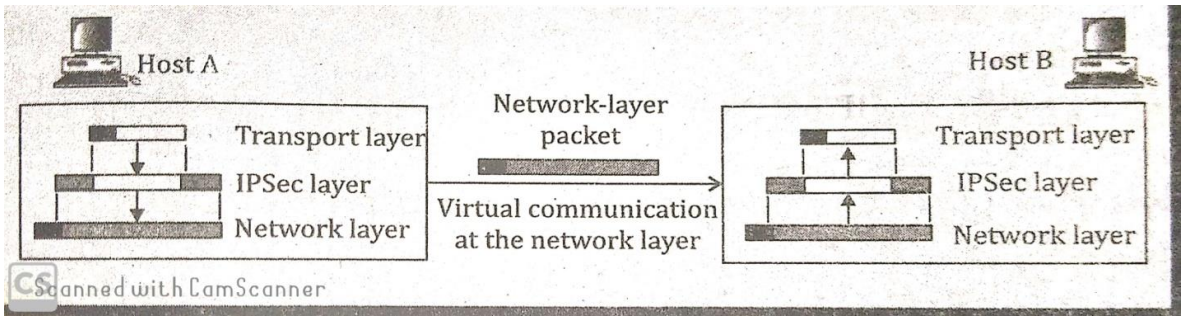
- Transport mode
- Tunnel mode

Transport mode

Transport mode provides protection primarily for upper layer protocols. That is, transport mode protects the payload of an IP packet, the pay load to be encapsulated in the network layer.

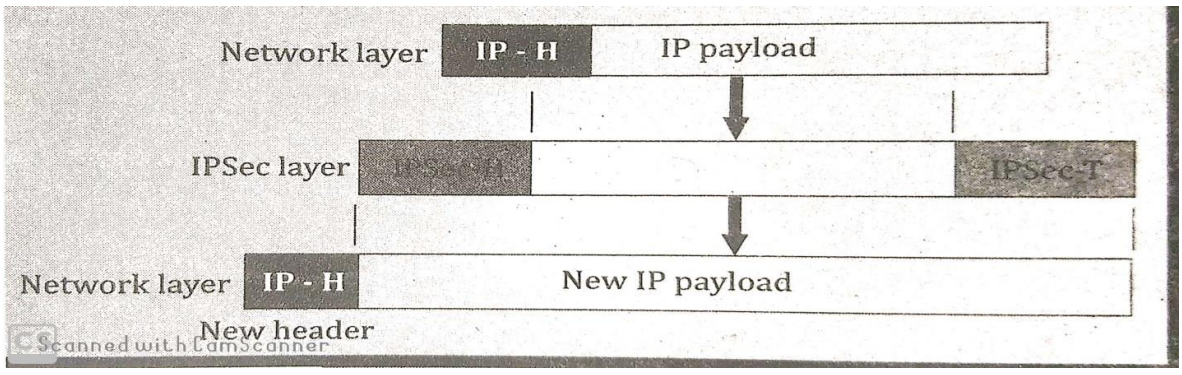


Transport mode is normally used when we need to protect end-to-end protection of data. The sending host uses IPSec to authenticate and/or encryot the payload coming from the transport layer. The receiving host uses IPSec to check the authentication and/or decrypt the IP packet and send to transport layer.



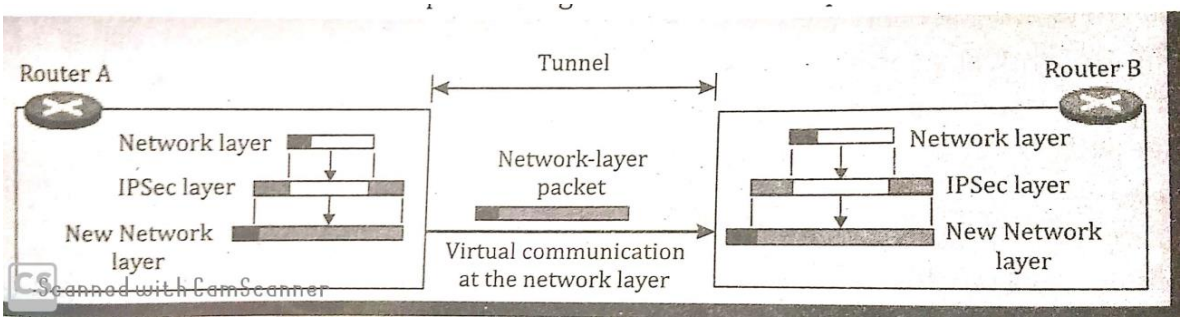
Tunnel mode:

Tunnel mode provides the protection to the entire IP packet. To achieve this, the header or trailer fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new "outer" IP packet with a new outer IP header.



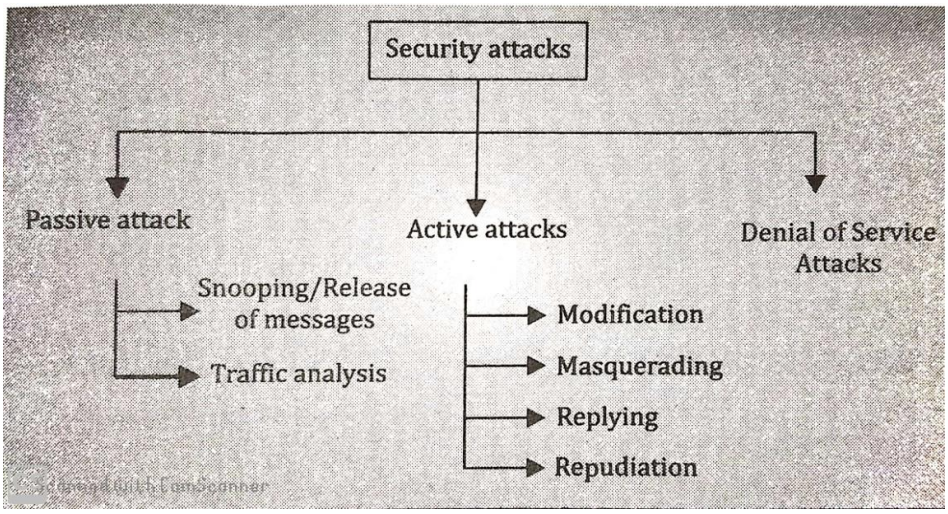
The entire original, or inner, packet travels through a "tunnel" from one point of an IP

network to another, no routers along the way are able to examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding to the security. With tunnel mode, a number of hosts on networks behind firewalls may engage in secure communications without implementing IPSec.



1) a) Explain in detail the taxonomy of attacks with relation to security goals. (unit-1) 2018

Attacks: Any action that compromises the security of information owned by an organization. Three goals of security- confidentiality, integrity and availability can be threatened by security attacks.



Passive Attacks: Attempts to learn or make use of information from the system but does not affect system resources. They are in the nature of eavesdropping on, or monitoring of transmissions. The goal of the opponent is to obtain information that is being transmitted.

i) **Release of message contents** – A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information.

ii) **Traffic analysis** – The eavesdropper analyses the traffic, determine the location, identify communicating hosts, and observes the frequency and length of being exchanged. Using all these information they predict the nature of communication is analysed through network but it cannot be altered.

Active Attacks: An active is a network exploit in which a hacker attempts to make changes to data on the target or data en-route to the target. In other words an active attack involves some modification of the data stream or the creation of a false stream and can be subdivided into four categories:

i) **Masquerade** – It may be attempted through the use of stolen login ID's and passwords, through finding security gaps in program or through bypassing the authentication mechanism.

Ex: An attacker might steal the bank card and pin and pretend to be the customer.

ii) **Replay attack** – A hacker steals an authorized user's log in information by stealing the session ID. It involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.

iii) **Modification of messages** – An intruder alters packet header addresses to direct a message to a different destination or modify the data on a target machine.

iv) **Repudiation** – Denial of service attack : Its is performed by one of the two parties in the communication The sender of the message might later deny that she has sent the message or receive of the message.. Might later deny that he has received the message

b) Discuss Extended Euclidean Algorithm (unit-1)

Given two integers a & b. We need to find the other two integers s & t such that

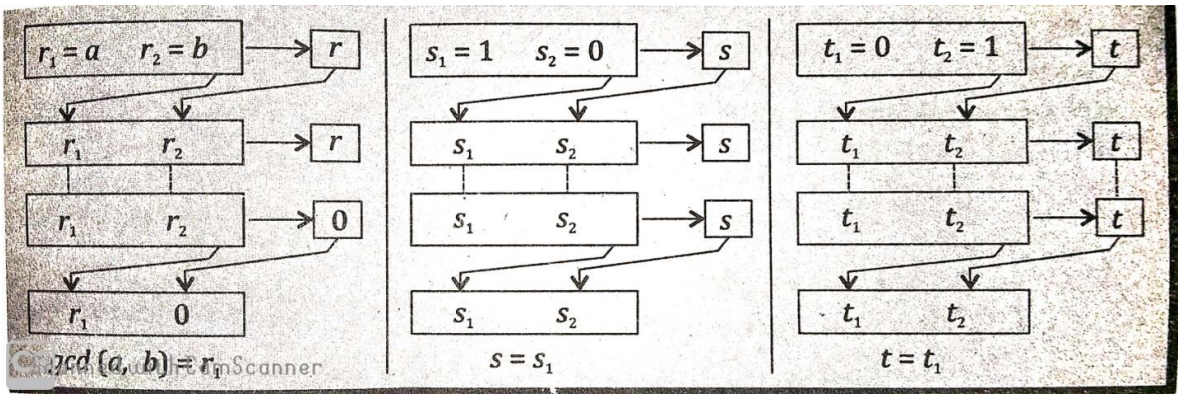
s * a + t * b = gcd(a,b)

The purpose of extended Euclidean algorithm is in Euclidean algorithm only we are concentrating about a, b & r. Here as an extension the value of sand t are find out by this method.

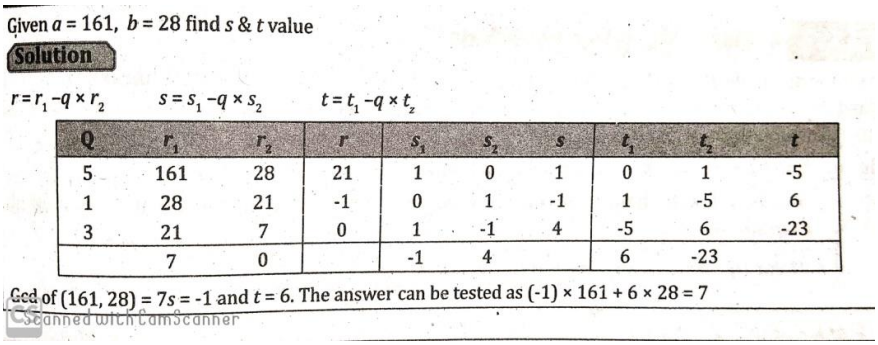
The extended Euclidean algorithm uses the same number of step as Euclidean algorithm.

In each step, the three sets of calculations & exchanges instead of one.

The algorithm uses three set of variables r's, s's & t's



In each step r_1 r_2 & r have the same values as in Euclidean algorithm. The variables r_1, r_2 are initialized to the values of a and b respectively. The variables s_1 * s_2 are initialized to 1 & 0 respectively. The variables t_1 and t_2 are initialized to 0 & 1 respectively. The calculations of r, s, & t are similar. Example:



Given $a = 161$, $b = 28$ find s & t value

Solution

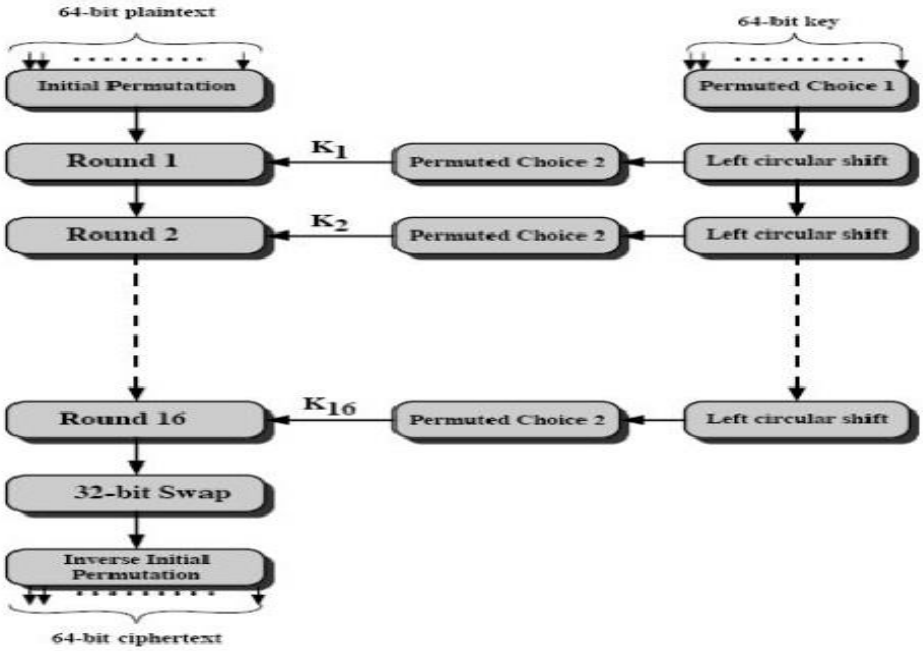
$r = r_1 - q \times r_2$ $s = s_1 - q \times s_2$ $t = t_1 - q \times t_2$

Q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	1	0	1	-5
1	28	21	-1	0	1	-1	1	-5	6
3	21	7	0	1	-1	4	-5	6	-23
	7	0		-1	4		6	-23	

Gcd of (161, 28) = $7s = -1$ and $t = 6$. The answer can be tested as $(-1) \times 161 + 6 \times 28 = 7$

2) a) Explain steps in DES Algorithm (unit-2)

The overall scheme for DES encryption is illustrated in the below figure. As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plain text must be 64-bits in length and the key is 56 bits in length. The processing of the plaintext proceeds in three phases. First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input. This is followed by a phase consisting of 16 rounds of the same function, which involves both permutation and substitute functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the preoutput.



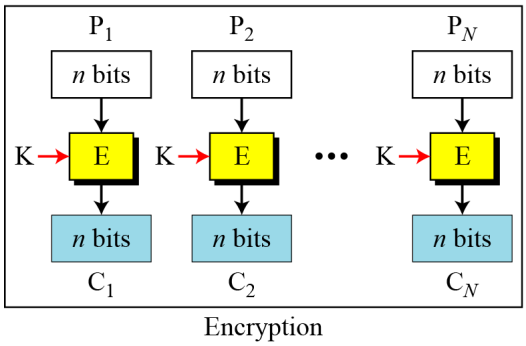
Finally, the preoutput is passed through a permutation (IP^{-1}) that is the inverse of the initial permutation function, to produce the 64-bit ciphertext. With the exception of the initial and final permutations, DES has the exact structure of a Fiestal cipher. Initially, the key is passed through a permutation function. Then, for each of the 16 rounds, a subkey (k_1) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

b) Discuss any two modes of operation in DES. (unit-3)

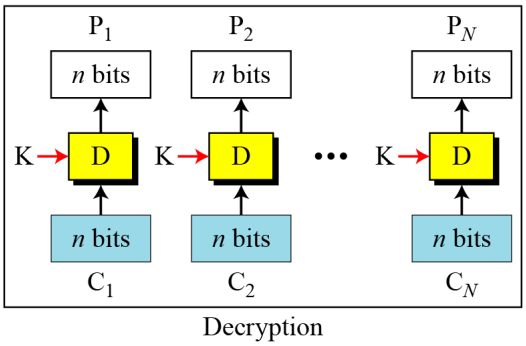
Electronic Codebook (ECB) Mode

The simplest mode of operation is called the electronic codebook (ECB) mode.

E: Encryption D: Decryption
P_i: Plaintext block *i* C_i: Ciphertext block *i*
K: Secret key



Encryption



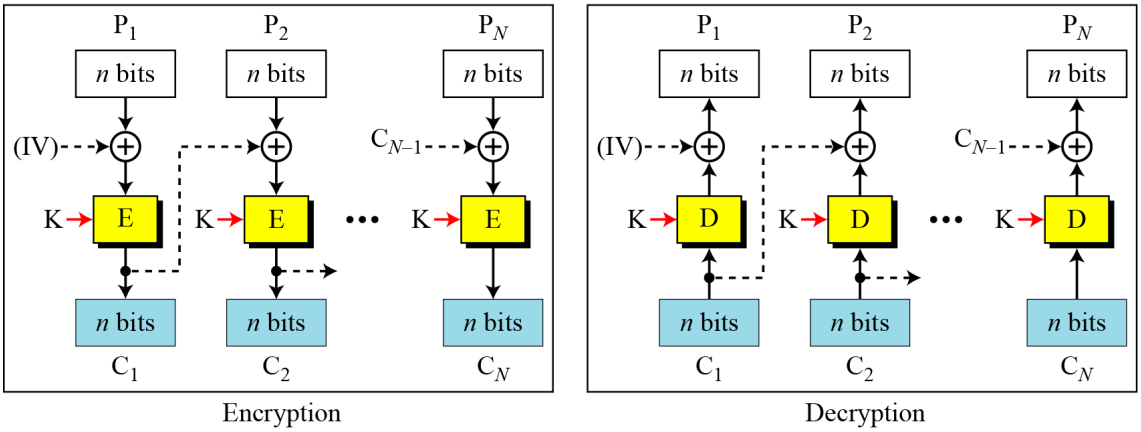
Decryption

Encryption: $C_i=E_k(P_i)$ Decryption: $P_i=D_k(C_i)$

The simplest method of running a block cipher is in the ECB mode. In this scheme, the message to be encrypted is first broken up into blocks of data equivalent to the cipher block size. If the fragment is less than this length, then padding can be used to ensure the entire block of information is filled. This method is typically insecure against modern cryptanalysis since the equal plaintext blocks always create equal ciphertext using the same key. As a result, patterns from the plain text message can be detected in the ciphertext output and ultimately be cracked.

Cipher Block Chaining (CBC) Mode

E: Encryption D : Decryption
P_i: Plaintext block *i* C_i : Ciphertext block *i*
K: Secret key IV: Initial vector (C_0)



Encryption:

C_0=IV

$C_i=E_k(P_i \oplus C_{i-1})$

Decryption:

C_0=IV

$P_i=D_k(C_i) \oplus C_{i-1}$

It can be proved that each plaintext block at Alice's site is recovered exactly at Bob's site. Because the encryption and decryption are inverses of each other.

$P_i=D_k(C_i) \oplus C_{i-1}=D_k(E_K(P_i \oplus C_{i-1})) \oplus C_{i-1}=P_i \oplus C_{i-1} \oplus C_{i-1}=P_i$

Initialization vector (IV)

The initialization vector (IV) should be known by the sender and the receiver.

In CBC mode, a single bit error in cipher text block C_j during transmission may create error in most bits in plain text block P_j during decryption.

The CBC mode, the initialization vector is sent along with the plaintext message. The value of the initialization vector has to be a pseudo-random or random value. It is added to the first plaintext block using an XOR operation prior to the initial encryption operation. The cipher text output from the first encryption block is subsequently used as the initializing vector for the next plaintext block meant to encrypt.

3) a) State and explain Chinese Remainder theorem with an example. (unit-3)

The Chinese remainder theorem (CRT) is used to solve a set of congruent equations with one variable but different moduli, which are relatively prime, as shown below

$x \equiv a_1 \pmod{m_1}$

$x \equiv a_2 \pmod{m_2}$

$x \equiv a_k \pmod{m_k}$

The CRT states that the above equations have a unique solution if the moduli are relatively prime

Example: $x \equiv 2 \pmod 3$

$x \equiv 3 \pmod 5$

$x \equiv 2 \pmod 7$

The solution to this set of equations is given in the next section; for the moment, note that the answer to this set of equations is x=23. This value satisfies all equations: $23 \equiv 2 \pmod 3$, $23 \equiv 3 \pmod 5$, and $23 \equiv 2 \pmod 7$.

Solution to CRT

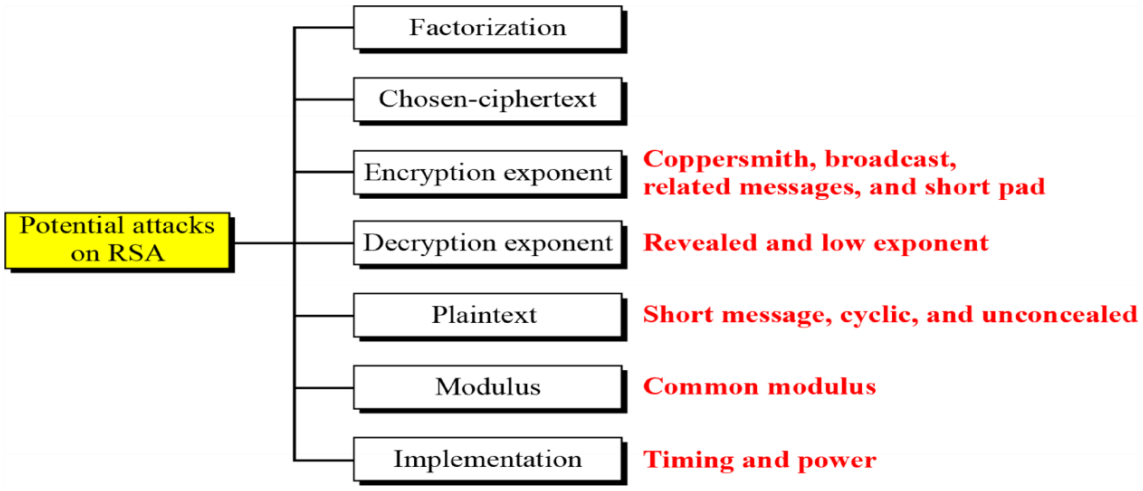
1. Find $M=m_1 * m_2 * \ldots * m_k$. This is the common modulus
2. Find $M_1=M/m_1$, $M_2=M/m_2,\ldots..$, $M_k=M/m_k$
3. Find the multiplicative inverse of $M_1, M_2, \ldots, M_k$ using the corresponding moduli $(m_1, m_2, ,\ldots, m_3)$. Call the inverses $M_1^{-1}, M_2^{-1}, \ldots, M_k^{-1}$.
4. The solution to the simultaneous equations is

$$X=(a_1*M_1*M_1^{-1}+a_2*M_2*M_2^{-1}+\ldots+a_k*M_k*M_k^{-1}) \bmod M$$

b) Discuss different attacks on RSA (unit-3)

Attacks on RSA

Taxonomy of potential attacks on RSA



Factorization Attack

The security of RSA is based on the idea that the modulus is so large that is infeasible to factor it in a reasonable time. Bob selects p and q and calculates n=p*q. Although n is public, p and q are secret. If Eve can factor n and obtain p and q, she can calculate $\phi(n)=(p-1)(q-1)$.Eve then calculated $d=e-1 \bmod \phi(n)$ because e is public. The private exponent d is the trapdoor that Eve can use to decrypt any encrypted message.

Choosen ciphertext attack

A potential attack on RSA is based on the multiplicative property of RSA. Assume Alice creates the cipher text sends C to Bob. Also assume that Bob will decrypt any arbitrary cipher text for Eve, other than C.

Attacks on the encryption exponent

To reduce the encryption time, it is tempting to use a small encryption exponent e. The common value for e is e=3.

Coppersmith theorem attack

To major low encryption exponent attack is refused to as the coppersmith theorem attack. According to the theorem in a modulo-n-polynomial f(x) of degree e, one can use an algorithm of the complexity log n to find the roots are smaller than $n^{1/e}$.

Broadcast attack

If one entity sends the same message to a group of recipients with the same low encryption exponent then it is a type of broadcast attack.

Related message attack

This attack is discovered by Franklin Reiter. Alice encrypt two plaintexts P_1 and P_2 and encrypts them with e=3 and sends C_1 and C_2 to Bob. If P_1 is related P_2 by a linear function, then Eve can recover P_1 and P_2 in a feasible computation time.

Short pad attack

This attack is discovered by Coppersmith. Alice as a message M to send to Bob. She pads the message with r_1 encrypts the result to get C_1 and sends C_1 to Bob. Eve intercepts C_1 and drops it. Bob informs Alice that he has not received message , so Alice pads the message again with r_2, encrypt it and are cipher text belonging to the same plain text.

Attacks on the decryption exponent

Two forms of attacks can be launched on the decryption exponent.

->Revealed exponent & low decryption exponent attack

Low Decryption exponent attack

Bob may think that using a small private key –d would make the decryption process faster for him. Wiener should that if $d < {}^{1}/_{3}\ n^{1/4}$, a special type of attack based on continuous fraction.

Plain text attacks

Plaintext and ciphertext in RSA are permutations of each other because they are integer in the same interval (0 to n-1).

4) a) Explain digital signature process with its security mechanism. (unit-4)

• Encipherment – It is hiding or covering of data (crypto and steganography)

• Data Integrity – This mechanism append to the data a short check value that has been created by a specific process from data itself. The receiver receives the data and checks the value .He creates new check value from received data and compares the newly created check value with the one received.

• Digital signature – The data appendned to or a cryptographic transformation of a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect forgery

• Authentication Exchange – Two entities exchange some message to prove their identify to each other – Something the user knows(unique secret) – Something the user has(security card) – Something the user is(fingerprint, retina voice)

• Traffic padding – It means inserting some dummy data into a network and present to the intruder different traffic pattern.

• Routing Control – It means selecting and continuously changing different available routes between the sender and the receiver to prevent eavesdropping

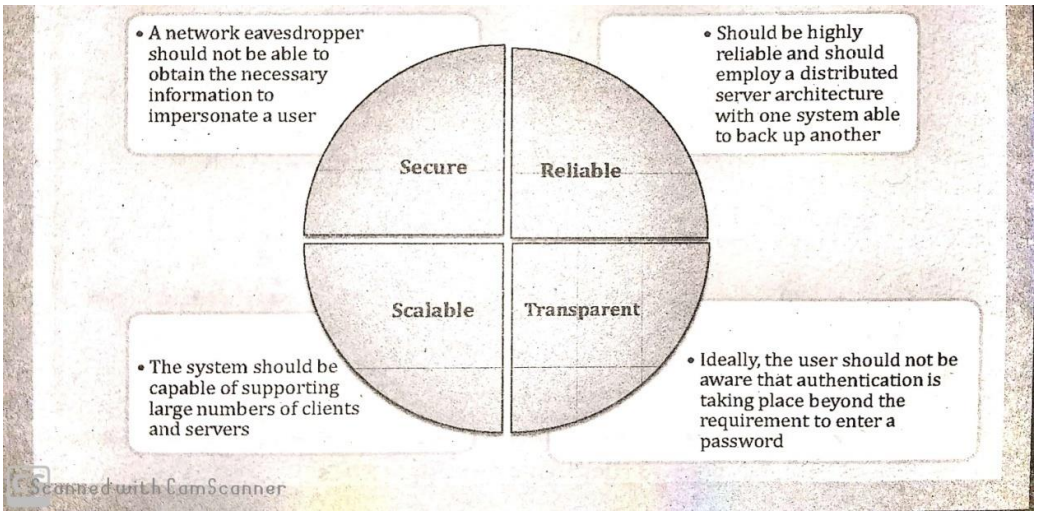
• Notarization – It means selecting third trusted party to control the communication between two parties. This can be done to prevent repudiation

• Access Control – It uses method to prove that user has to right to the data or resources owned by a system

b) Write a note on Kerberos (unit-4)

Kerberos is an authentication protocol, and at the same time a KDC that has become very popular several systems including Windows 2000, use Kerberos, originally designed at MIT, it is named after three headed dog in Greek mythology that guards the gates of Hades. Properties of Kerberos are:

- A workstation cannot be trusted to identify its users correctly to network services.
- A user may gain access to a particular workstation and pretend to be another user operating from that work station.
- A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
- A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.
- Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.
- Relies exclusively on symmetric encryption, making no use of public key encryption.



5) a) Explain Public Key Infrastructure(PKI) in detail (unit-4)

PKI is a model for creating, distributing and revoking certificates based on the X.509. IETF has created the public key infrastructure X.509(PKIX).

Some duties of PKI

- Issue, renew and revoke certificates.
- Store and update private keys for members who wish to hold their private keys at a safe place.
- Provides services to other internet security protocols that need public key info such as IPSec and TLS.
- Provide access control i.e., levels of information stored in its database.

PKI Trust Model

For scalability, there should be many certification authorities in the world; each CA handles a specified number of certificates.

- The PKI trust model defines the rules that specify how a user can verify a certificate received from a CA
- As an example, the PKI hierarchical trust model defines hierarchical rules that specify how a user can verify a certificate received from a CA.
- PKI uses the following notation to denote the certificate issued and signed by certification authority X for entity Y. X<<Y>>

b) Differentiate between MIME and S/MIME (unit-5)

PGP/MIME is an open source software package that is designed for the purpose of email security. Phil Zimmerman developed it. It provides the basic or fundamental needs of cryptography. In this multiple steps such are taken to secure the email, these are:

1) Confidentiality
2) Authentication
3) Compression
4) Resemble
5) Segmentation
6) E-mail compatibility

2. Secure/Multipurpose Internet Mail Extension (S/MIME) :

S/MIME is a security-enhanced version of Multipurpose Internet Mail Extension (MIME). In this, public key cryptography is used for digital sign, encrypt or decrypt the email. User

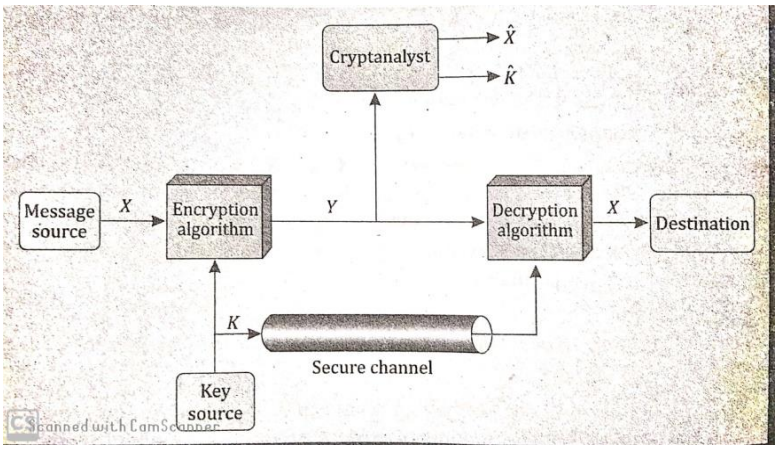
acquires a public-private key pair with a trusted authority and then makes appropriate use of those keys with email applications.

Difference between MIME/PGP and S/MIME :

1.	It is designed for processing the plain texts	While it is designed to process email as well as many multimedia files.
2.	PGP is less costly as compared to S/MIME.	While S/MIME is comparatively expensive.
3.	PGP is good for personal as well as office use.	While it is good for industrial use.
4.	PGP is less efficient than S/MIME.	While it is more efficient than PGP.
5.	It depends on user key exchange.	Whereas it relies on a hierarchically valid certificate for key exchange.
6.	PGP is comparatively less convenient.	While it is more convenient than PGP due to the secure transformation of all the applications.
7.	PGP contains 4096 public keys.	While it contains only 1024 public keys.
8.	PGP is the standard for strong encryption.	While it is also the standard for strong encryption but has some drawbacks.
9.	PGP is also be used in VPNs.	While it is not used in VPNs, it is only used in email services.
10.	PGP uses **Diffie hellman digital signature**.	While it uses **Elgamal digital signature**.

2019

1) a) Briefly explain the model of conventional cryptosystem. (unit-2)



The encryption process consists of an algorithm and a key. The key is a value independent of the plaintext. The algorithm will produce a different output depending on the specific key being used at the time. Changing the key changes the output of the algorithm, i.e., the cipher text.

Once the cipher text is produced, it may be transmitted. Upon reception, the cipher text can be transformed back to the original plaintext by using a decryption algorithm and the same key that was used for encryption.

- A source produces a message in plaintext, $X=[X_1,X_2,\ldots..,X_M]$.
- The M elements of X are letters in some finite alphabet. Traditionally, the alphabet usually consisted the 26 capital letters.
- For encryption, a key of the form $K=[K_1,K_2,\ldots..,K_J]$ is generated. If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel.
- With the message X and the encryption key K as input, the encryption algorithm forms the ciphertext $Y=[Y_1,Y_2,\ldots\ldots,Y_N]$. We can write this as Y=E(K,X)
- At the receiver end, in possession of the key, is able to convert the ciphertext:

X=D(K,Y) An opponent, observing Y but not having access to K or X, may attempt to recover X or K or both X and K

It is assumed that the opponent knows the encryption[E] and decryption[D] algorithms.

If the opponent (cryptananlyst) is interested in only this particular message, then the focus of the effort is to recover X by generating a plaintext and can also an attempt is made to recover K by generating an estimate.

b) Explain the rules of play fair cipher with an example. (unit-1)

The **Play fair cipher** was the first practical digraph substitution cipher. The scheme was invented in **1854** by **Charles Wheatstone** but was named after Lord Play fair who promoted the use of the cipher. In play fair cipher unlike traditional cipher we encrypt a pair of alphabets (digraphs) instead of a single alphabet.
It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians during World War II. This was because Play fair is reasonably fast to use and requires no special equipment.

Plaintext digraphs are encrypted with the matrix by first locating the two plaintext letters in the matrix. They are:

(1) In different rows and columns;

(2) In the same row;

(3) In the same column;

(4) Alike.

The corresponding encryption (replacement) rules are the following:

1. When the two letters are in different rows and columns, each is replaced by the letter that is in the same row but in the other column; i.e., to encrypt WE, W is replaced by U and E by G.
2. When A and R are in the same row, A is encrypted as R and R (reading the row cyclically) as M.
3. When I and S are in the same column, I is encrypted as S and S as X.
4. When a double letter occurs, a spurious symbol, say Q, is introduced so that the MM in SUMMER is encrypted as NL for MQ and CL for ME.
5. An X is appended to the end of the plaintext if necessary to give the plaintext an even number of letters.

- Repeating plaintext letters that would fall in the same pair are separated with a filler letter such as ‘x’.
- Plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row following the last.

- Plaintext letters that fall in the same column are replaced by the letter beneath, with the top element of the column following the last.
- Otherwise, each plaintext letter is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.

Example: plaintext= meet me at the school house

Splitting 2 letters as a unit= me et me at th es ch oo lh ou se

Corresponding cipher text= CL KL CL RS PD IL HY AV MP HF XL IU

2) a) Explain the procedure for RSA cryptosystem. (Unit-3)

RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption.

The RSA algorithm holds the following features:

- RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
- The integers used by this method are sufficiently large making it difficult to solve.
- There are two sets of keys in this algorithm: private key and public key.

You will have to go through the following steps to work on RSA algorithm −

Step 1: Generate the RSA modulus

The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown −

N=p*q

Here, let N be the specified large number.

Step 2: Derived Number (e)

Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1

Step 3: Public key

The specified pair of numbers **n** and **e** forms the RSA public key and it is made public.

Step 4: Private Key

Private Key **d** is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows −

ed = 1 mod (p-1) (q-1)

The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

Encryption Formula

Consider a sender who sends the plain text message to someone whose public key is **(n,e).** To encrypt the plain text message in the given scenario, use the following syntax −

C = Pe mod n

Decryption Formula

The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver **C** has the private key **d**, the result modulus will be calculated as −

Plaintext = Cd mod n

b) Differentiate between symmetric and asymmetric key cryptography. (Unit-4)

Symmetric key	**Asymmetric Key**
Symmetric encryption uses a single key that needs to be shared among the people who need to receive the message	Asymmetrical encryption uses a pair of public key and a private key to encrypt and decrypt messages when communicating.
Symmetric encryption is an old technique	Asymmetric encryption is relatively new.
Symmetrical encryption model, eliminating the need to share the key by using a pair of public-private keys.	Asymmetric encryption was introduced to complement the inherent problem of the need to share the key
Symmetric encryption takes relatively less time than the Asymmetric encryption.	Asymmetric encryption takes relatively more time than the symmetric encryption.
In symmetric key encryption, resource utilization is low as compared to asymmetric key encryption.	In asymmetric key encryption, resource utilization is high.
It is used when a large amount of data is required to transfer.	It is used to transfer small amount of data.

The size of cipher text is same or smaller than the original plain text.	The size of cipher text is same or larger than the original plain text.

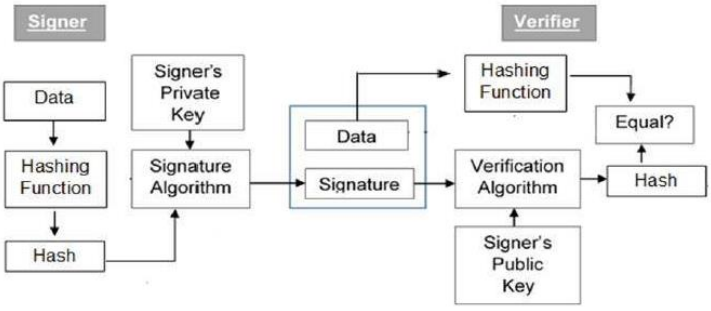
3) a) Explain the working of digital signature with neat diagram. (unit-4)

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message.

Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.



The following points explain the entire process in detail −

- Each person adopting this scheme has a public-private key pair.
- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.

- Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
- Verifier also runs same hash function on received data to generate hash value.
- For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
- Since digital signature is created by 'private' key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

b) How does PGP provide confidentiality and authentication e-mail? Explain. (unit-5)

PGP (Pretty Good Privacy), is a popular program that is used to provide confidentiality and authentication services for electronic mail and file storage. It was designed by **Phil Zimmermann** way back in 1991. He designed it in such a way, that the best cryptographic algorithms such as RSA, Diffie-Hellman key exchange, DSS are used for the public-key encryption (or) asymmetric encryption; CAST-128, 3DES, IDEA are used for symmetric encryption and SHA-1 is used for hashing purposes. PGP software is an open source one and is not dependent on either of the OS (Operating System) or the processor.

The following are the services offered by PGP:

1. **Authentication:**

Authentication basically means something that is used to validate something as true or real. To login into some sites sometimes we give our account name and password, that is an authentication verification procedure.

In the email world, checking the authenticity of an email is nothing but to check *whether it actually came from the person it says*. In emails, authentication has to be checked as there are some people who spoof the emails or some spams and sometimes it can cause a lot of inconvenience.

2. **Confidentiality:**

Sometimes we see some packages labeled as 'Confidential', which means that those packages are not meant for all the people and only selected persons can see them. The same applies to the email confidentiality as well. Here, in the email service, only the sender and the receiver should be able to read the message that means the contents have to be kept secret from every other person, except for those two.

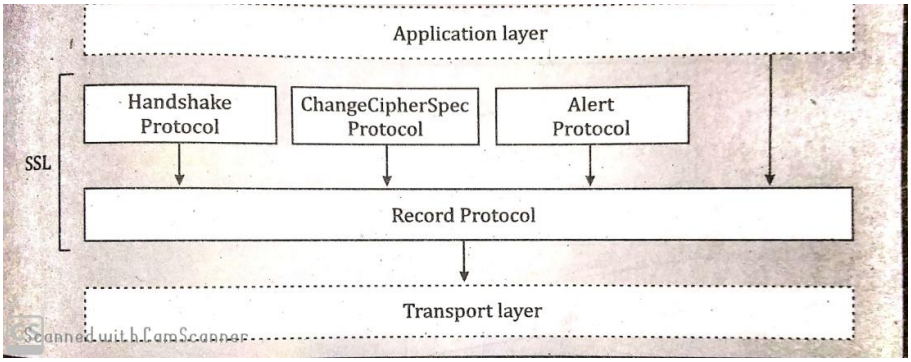
4 a) List and explain the four protocols of SSL. (unit-5)

Secure Socket Layer (SSL) provide security to the data that is transferred between web

browser and server. SSL encrypt the link between a web server and a browser which ensures that all data passed between them remain private and free from attack.

Secure Socket Layer Protocols:

- SSL record protocol
- Handshake protocol
- Change-cipher spec protocol
- Alert protocol



SSL Record Protocol:

SSL Record provide two services to SSL connection.

- Confidentiality
- Message Integrity

In SSL Record Protocol application data is divided into fragments. The fragment is compressed and then encrypted MAC (Message Authentication Code) generated by algorithms like SHA (Secure Hash Protocol) and MD5 (Message Digest) is appended. After that encryption of the data is done and in last SSL header is appended to the data.

Handshake Protocol:

Handshake Protocol is used to establish sessions. This protocol allows client and server to authenticate each other by sending a series of messages to each other. Handshake protocol uses four phases to complete its cycle.

- **Phase-1:** In Phase-1 both Client and Server send hello-packets to each other. In this IP session, cipher suite and protocol version are exchanged for security purpose.
- **Phase-2:** Server send his certificate and Server-key-exchange. Server end the phase-2 by sending Server-hello-end packet.
- **Phase-3:** In this phase Client reply to the server by sending his certificate and Client-exchange-key.

- **Phase-4:** In Phase-4 Change-cipher suite occurred and after this Handshake Protocol ends.

Change-cipher Protocol:

This protocol uses SSL record protocol. Unless Handshake Protocol is completed, the SSL record Output will be in pending state. After handshake protocol the Pending state is converted into Current state.

Change-cipher protocol consists of single message which is 1 byte in length and can have only one value. This protocol purpose is to cause the pending state to be copied into current state.

Alert Protocol :

This protocol is used to convey SSL-related alerts to the peer entity. Each message in this protocol contains 2 bytes.

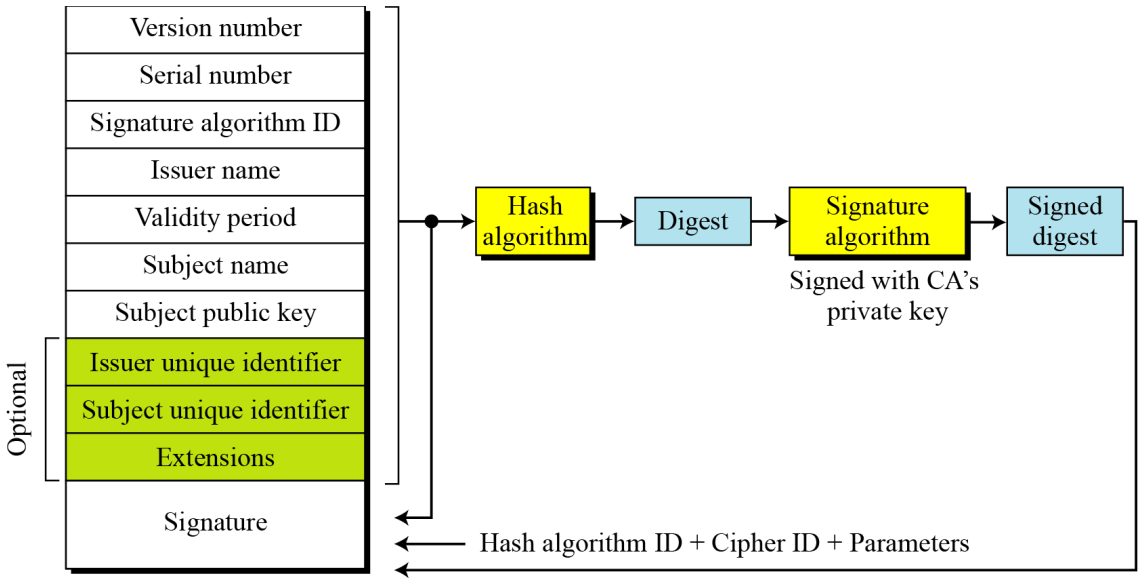
Level is further classified into two parts:

- **Warning:**

  This Alert have no impact on the connection between sender and receiver.
- **Fatal Error:**

  This Alert breaks the connection between sender and receiver.

b) Explain x.509 certificate. (Unit-4)



- **Version Number:** this field is the version of X.509
- **Serial number:** this field is the serial number assigned to each certificate and is unique for each certificate and is unique for each certificate issuer.

- **Signature algorithm ID:** This field identifies the signature algorithm used in the certificate. This field is repeated in the signature field.
- **Issuer name:** this field identifies the CA that issued the certificate.
- **Validity Period:** this field defines the earliest time and the latest time during which the certificate is valid.
- **Subject name:** this field defines the entity that owns the public key stored in this certificate.
- **Subject public key:** this field gives the value of the public key of the owner of the certificate and defines the public key algorithm.
- **Signature:** this field contains the digest of all other fields in the certificate encrypted by the CA's private key and also contains the ID of the signature algorithm.
- **Certificate Renewal:** Each certificate has a period of validity. If there is no problem with the certificate, the CA issues a new certificate before the old one expires.
- **Certificate Revocation:** In some cases a certificate must be revoked before its expiration. The revocation is done by periodically issuing a certificate revocation list that contains all revoked certificates that have not expired on the date the CRL is issued. It includes the fields such as:
    - Signature algorithm ID
    - Issuer name
    - This update date
    - Next update date
    - Revoked certificate
    - Signature
    - **Delta Revocation:** To make revocation more efficient, the delta certificate revocation list has been introduced. A delta CRL is created and posted on the directory if there are changes after this update date and next update date.

5) Discuss in detail block cipher modes of operations. (Unit-2)

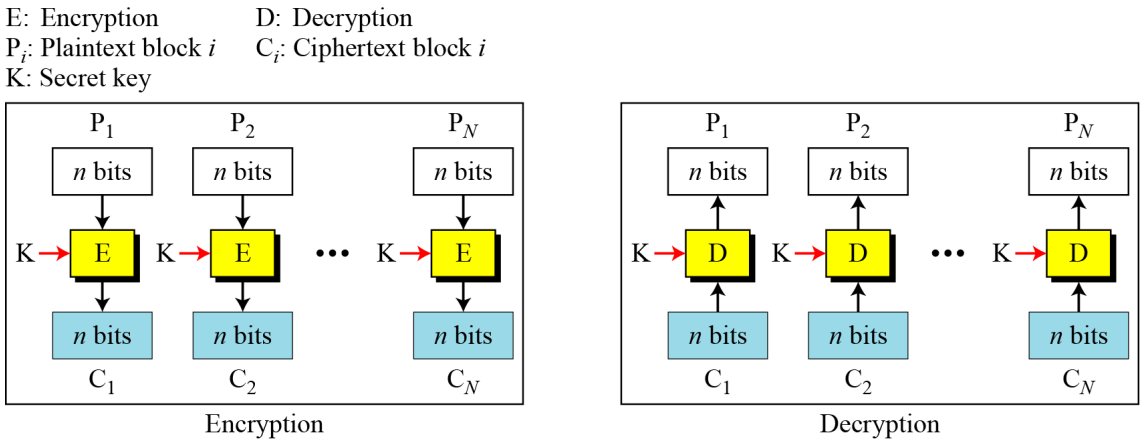
Block cipher is an encryption algorithm which takes fixed size of input say *b* bits and produces a cipher text of *b* bits again. If input is larger than *b* bits it can be divided further. For different applications and uses,

There are several modes of operations for a block cipher.

Electronic Code Book (ECB) –

Electronic code book is the easiest block cipher mode of functioning. It is easier because of

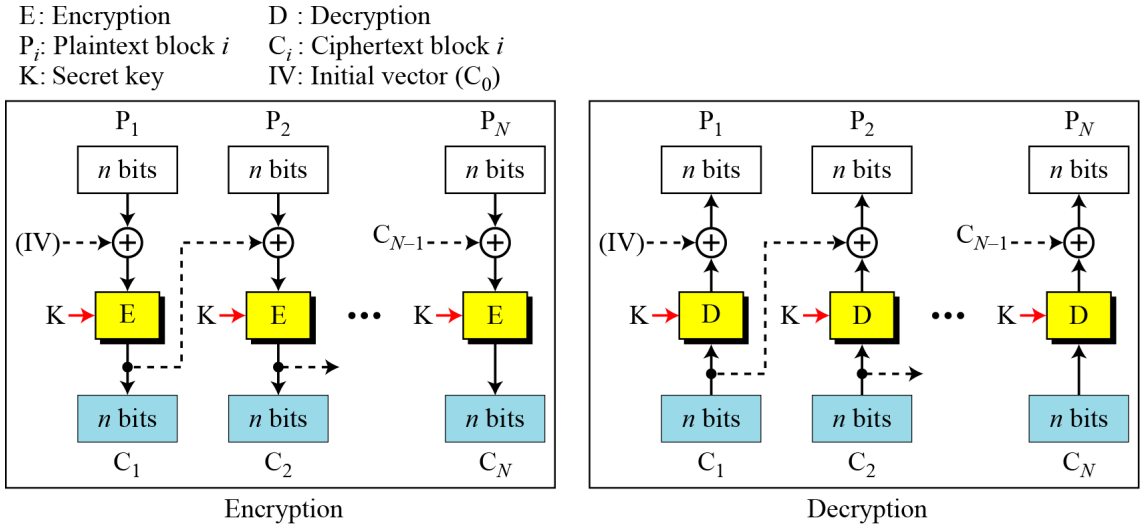
direct encryption of each block of input plaintext and output is in form of blocks of encrypted cipher text. Generally, if a message is larger than *b* bits in size, it can be broken down into bunch of blocks and the procedure is repeated.



Cipher Block Chaining –

Cipher block chaining or CBC is an advancement made on ECB since ECB compromises some security requirements. In CBC, previous cipher block is given as input to next encryption algorithm after XOR with original plaintext block. In a nutshell here, a cipher block is produced by encrypting a XOR output of previous cipher block and present plaintext block.

The process is illustrated here:

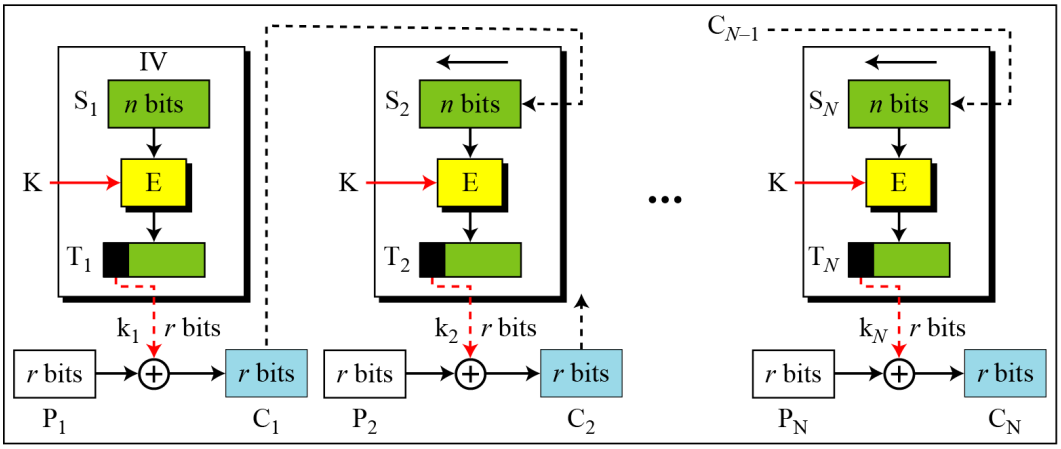


Cipher Feedback Mode (CFB) –

In this mode the cipher is given as feedback to the next block of encryption with some new specifications: first an initial vector IV is used for first encryption and output bits are divided

as set of *s* and *b-s* bits the left hand side *s* bits are selected and are applied an XOR operation with plaintext bits. The result given as input to a shift register and the process continues. The encryption and decryption process for the same is shown below, both of them use encryption algorithm.

E : Encryption
P_i: Plaintext block *i*
K: Secret key
D : Decryption
C_i: Ciphertext block *i*
IV: Initial vector (S_1)
S_i: Shift register
T_i: Temporary register

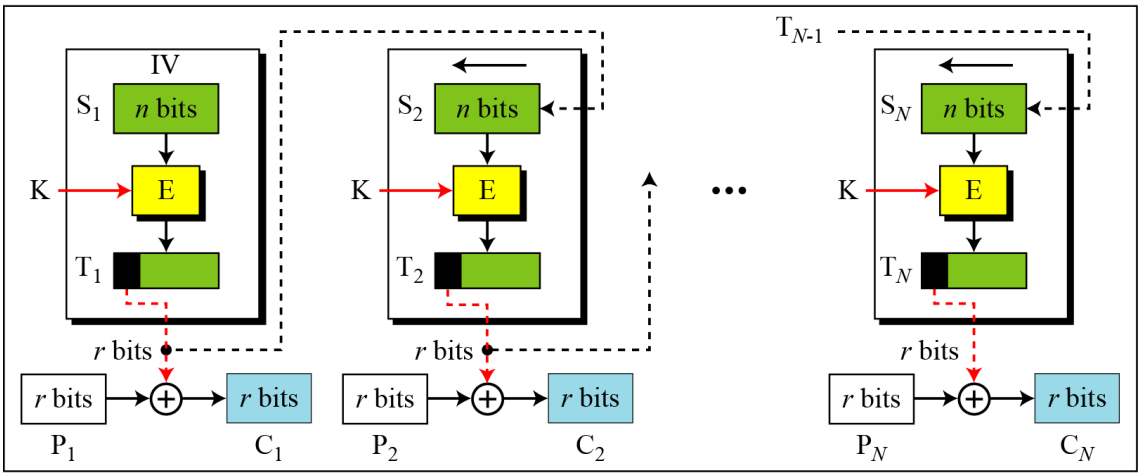


Encryption

Output Feedback Mode –

The output feedback mode follows nearly same process as the Cipher Feedback mode except that it sends the encrypted output as feedback instead of the actual cipher which is XOR output. In this output feedback mode, all bits of the block are send instead of sending selected *s* bits. The Output Feedback mode of block cipher holds great resistance towards bit transmission errors. It also decreases dependency or relationship of cipher on plaintext.

E : Encryption
P_i: Plaintext block i
K: Secret key
D : Decryption
C_i: Ciphertext block i
IV: Initial vector (S_1)
S_i: Shift register
T_i: Temporary register

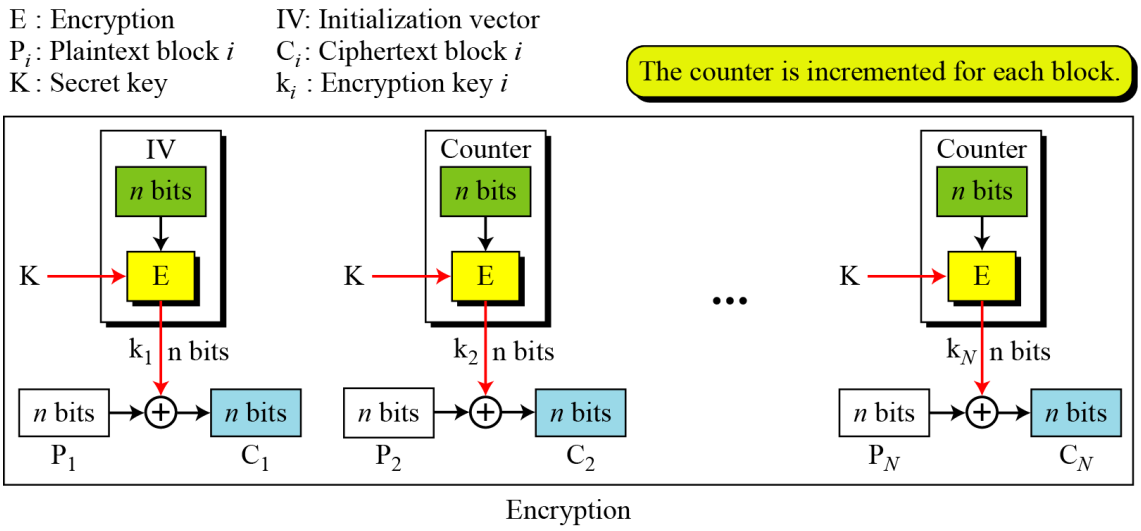


Encryption

Counter Mode –

The Counter Mode or CTR is a simple counter based block cipher implementation. Every time a counter initiated value is encrypted and given as input to XOR with plaintext which results in cipher text block. The CTR mode is independent of feedback use and thus can be implemented in parallel.

Its simple implementation is shown below:



Encryption

6) List and explain the properties of Hash functions. (Unit-4)

The hash function is desired to possess following properties −

- **Pre-Image Resistance**
    - This property means that it should be computationally hard to reverse a hash function.
    - In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.
    - This property protects against an attacker who only has a hash value and is trying to find the input.
- **Second Pre-Image Resistance**
    - This property means given an input and its hash, it should be hard to find a different input with the same hash.

- In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).
- This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.

- **Collision Resistance**
  - This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function.
  - In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).
  - Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.
  - This property makes it very difficult for an attacker to find two input values with the same hash.
  - Also, if a hash function is collision-resistant **then it is second pre-image resistant.**

*****

SA – 920

VI Semester B.C.A. Degree Examination, April/May 2015
(Y2K8 Scheme)
COMPUTER SCIENCE
BCA – 604 : Web Programming
(70 Marks 2013-14 and Onwards/60 Marks Prior to 2013-14)

Time : 3 Hours Max. Marks : 60/70

Instructions: *1) Answer **all** questions.*
*2) Section **D** is applicable to students who have admitted in **2013-14** and onwards **only**.*

SECTION – A

Answer **any ten** questions. **Each** question carries **one** mark. **(10×1=10)**

1. What is DNS ?
2. Explain rowspan and colspan with an example.
3. How to create hyperlinks in HTML ?
4. Explain < img> tag with an example.
5. What are and tags ?
6. What is universal selector ?
7. Write any two methods of string object.
8. What are composite data types in java script ?
9. What is DHTML ?
10. What is the use of navigator object ?
11. What is the use of focus and blur events ?
12. What is XML ?

P.T.O.

SECTION – B

Answer **any five** questions. **Each** question carries **three** marks. **(5×3=1**

13. Explain GET and POST request methods.
14. Explain the basic table tags with different attributes.
15. What are the different levels of CSS ? Explain.
16. Explain the various looping statements in java script.
17. Write a java script code to find factorial of a given number.
18. Write a short note on DOM.
19. What are the different ways of registering an event handler ?
20. Write the differences between XML schema and DTD.

SECTION – C

Answer **any five** questions. **Each** question carries **seven** marks. **(5×7=35)**

21. What is the role of web server ? Explain about Apache and IIS web servers.
22. a) Explain <frame> tag with its attributes. **3**
    b) Explain the attributes of <input> tag with an example. **4**
23. Explain any seven types of selectors in CSS.
24. What is pattern matching ? Explain different methods used in pattern matching.
25. What are the different types of positioning techniques ? Explain.
26. What are the different ways of accessing form elements ? Explain.
27. What are the different schema elements in XML ?
28. Create a form for student information. Write a java script code to find total, average, result and grade.

SECTION – D

Answer **any one** question. **Each** question carries **ten** marks. **(10×1=10)**

29. What are lists in XHTML ? Give an example to implement various types of lists.
30. What are events ? Explain different categories of events in XHTML.

MS – 585

39

VI Semester B.C.A. Degree Examination, May 2016
(Y2K8 Scheme) (Fresh+Repeaters)
COMPUTER SCIENCE
BCA – 604 : Web Programming
70 Marks – 2013-14 & Onwards
60 Marks – Prior to 2013-14

Time : 3 Hours Max. Marks : 60/70

Instructions : *A) Answer **all** the questions.*
*B) Section **D** is applicable to students who have admitted in **2013 – 14** and onwards.*

SECTION – A

Answer **any 10** questions. **(10×1=10)**

1. What is a protocol ? State any two protocols.
2. What is the use of <img> tag ? Give the syntax of <img> tag.
3. What are forms ?
4. What is universal selector in CSS ?
5. What is the use of and tags ?
6. What do you mean by an object in java script ?
7. What is DHTML ? Explain.
8. What do you mean by relative positioning ?
9. What are mouse events ?
10. What is XML ?
11. What is MIME ? Explain.
12. What is ID selector in CSS ? How it will be represented ?

SECTION – B

Answer **any five** questions. **(5×3=15)**

13. What is DNS ? Explain domain and subdomain with suitable examples.
14. What is an unordered list ? Explain with example.

P.T.O.

15. What is CSS ? What are the advantages and disadvantages of using CSS in XHTML ?
16. How to create date object in javascript ? Explain any four methods of javascript date object.
17. Explain screen output and keyboard input methods in javascript.
18. Explain match (), search () and split (), the pattern matching methods of string.
19. What are functions ? Explain how to create and call functions ?
20. Write notes on XML name spaces.

SECTION – C

Answer **any 5**. **Each** question carries **7** marks. **(5×7=35)**

21. What is a frame ? Explain <frame> and <frameset> tags with its attributes and examples.
22. Explain HTTP request method.
23. Explain any 7 XHTML tags with examples.
24. What are the different levels of CSS ? Explain.
25. Explain javascript. What are the different ways that we can include javascript to a XHTML document ?
26. Explain different ways of element access in javascript with example.
27. Compare XML schema and DTD.
28. Write a javascript code to find sum of N natural numbers using user defined function.

SECTION – D

Answer **any one** question. **(1×10=10)**

29. What is a form ? What are the major attributes of the form ? Explain any six form components with example.
30. What is an array object ? How to create an array object ? Explain any six array methods.

———————

US – 647

VI Semester B.C.A. Examination, May 2017
(CBCS)
(2016-17 and Onwards)
COMPUTER SCIENCE
BCA 604 : Web Programming

Time : 3 Hours Max. Marks : 70

Instruction : *Answer all Sections.*

SECTION – A

Answer **any ten** questions. **Each** question carries **two** marks. **(10×2=20)**

1. What is a web browser ? Mention any three web browsers.
2. What is a DTD ? Mention its types.
3. What is the use of <img> tag ? Give the syntax of <img> tag.
4. What is universal selector ?
5. What is an ID selector in CSS ?
6. What do you mean by an Math and date object in Java Script ?
7. What is DHTML ? Explain.
8. What are composite datatypes in Java Script ?
9. What is XML schema ?
10. What are Mouse events ?
11. What is an event handler ?
12. What is MIME ? Explain.

P.T.O.

SECTION – B

Answer **any five** questions. **(5×10=50)**

13. a) What is DNS ? Explain domain and sub-domain with suitable examples. **5**

    b) What are the different levels of headings in HTML ? **5**

14. What are the different types of lists supported in XHTML ? Explain **10**

15. a) Explain different levels of CSS. **5**

    b) Explain any five types of selectors in CSS. **5**

16. a) What is a Java Script ? What are data types supported by Java Script . **5**

    b) Write a Java Script code to find factorial of a number. **5**

17. What is a form ? Explain form components with example. **10**

18. a) What are the different types of positioning techniques ? Explain. **5**

    b) Explain table tag with example. **5**

19. a) Create a form for student information. Write Java Script code to find total, average, result and grade. **5**

    b) What is DTD ? Explain internal DTD and external DTD. **5**

20. a) Explain XHTML events. **5**

    b) Write a short note on DOM. **5**

SM – 626

36

VI Semester B.C.A. Examination, May/June 2018
(CBCS) (F+R) (2016 – 17 & Onwards)
COMPUTER SCIENCE
BCA 604 : Web Programming

Time : 3 Hours Max. Marks : 70

Instruction : *Answer **all** Sections.*

SECTION – A

Answer **any ten** questions. **Each** question carries **two** marks. **(10×2=20)**

1. Define web browser.
2. What do you mean by IP address ?
3. Define URL. Give an example.
4. What is MIME ?
5. Give any two differences between HTML and XHTML.
6. What is navigator object ?
7. What is event and event handler ?
8. What is XML schema ?
9. List out any two mouse events.
10. What do you mean by namespaces ?
11. What is constructor in Javascript ?
12. What is XSLT style sheet ?

SECTION – B

Answer **any five** questions. **Each** question carries **10** marks. **(5×10=50)**

13. a) Explain lists in XHTML with examples. **5**

    b) Illustrate < frameset > tag with example. **3**

    c) Differentiate < span > and < div > tags with examples. **2**

P.T.O.

14. a) What is CSS ? Explain the different levels of CSS with examples. 5

    b) Write a JavaScript program to implement all string operations. 5

15. a) What is selector in CSS ? Distinguish class and ID selectors with examples. 5

    b) What is a form ? Illustrate the major attributes of form with examples. 5

16. a) What is an array in JavaScript ? How to create an array object ? Give an example. 5

    b) What is the general syntax of a JavaScript function ? Give an example. 5

17. Explain pattern matching using regular expressions in JavaScript. 10

18. a) Illustrate DOM. 5

    b) Write a JavaScript program to implement keyboard events. 5

19. a) What is DTD ? Explain the type of DTD. 5

    b) Write a JavaScript program to find factorial of a list of numbers, reading input as command line argument. 5

20. a) Explain CSS box model. 5

    b) Describe XML processors. 5

103648

No. of Printed Pages : **2**

GS-645

VI Semester B.C.A. Examination, May/June - 2019

BCA 604 : WEB PROGRAMMING

(CBCS) (F+R) (2016-17 & Onwards)

Time : 3 Hours Max Marks : 70

Instructions : Answer **all** Sections

SECTION - A

I. Answer **Any Ten** questions. Each question carries **two** marks **10x2=20**

1. Define IP address.

2. What is MIME ?

3. What is HTML event with example ?

4. How to create hyper link in HTML with example ?

5. What is an ID selector in CSS ?

6. Explain any 2 string methods in Java Script ?

7. What is the use of and tags ?

8. What do you mean by absolute positioning of an element ?

9. What is an event and event handler ?

10. What is XML schema ?

11. Define <pre> tag with an example ?

12. What are mouse events ?

P.T.O.

SECTION - B

II. Answer **any five** questions. Each question carries **ten** marks. **5x10=50**

13. (a) What is the role of webserver ? Explain Apache and IIS web servers. 5

(b) Explain any 5 network protocols. 5

14. (a) Explain Table Tags and its sub tags with example. 5

(b) Explain the lists in XHTML with example. 5

15. (a) Explain the different levels of CSS with example. 7

(b) Explain the Get and post request methods. 3

16. (a) Explain any 3 selectors in CSS with example. 6

(b) Explain the attributes of <input> tag with example. 4

17. What is a Form ? What are the major attributes of the Form ? Explain Form components with example. 10

18. (a) Explain the CSS Box Model ? 5

(b) Write a Java script code to evaluate mathematical expression and display the result. 5

19. (a) What is DTD? Explain internal DTD and external DTD. 5

(b) Explain any 5 XHTML Tags with example ? 5

20. (a) Explain in detail about DOM ? 5

(b) Explain different ways of elements access in Java Script with example ? 5

- o 0 o -

SOUNDARYA EDUCATIONAL TRUST (R)
SOUNDARYA INSTITUTE OF MANAGEMENT & SCIENCE
Soundarya Nagar, Sidedahalli, Hessaraghatta Main Road, Bangalore 73
Department of Computer Science

BCA- 604: Web Programming
2 Marks Solution Bank

UNIT 1

[May/June 2015]

1. What is DNS?

Domain Name System is a database system that translates a computers fully qualified domain name into an IP address. The DNS is used to resolve human readable hostnames into machine readable IP address. DNS also provides other information about domain names such as mail services.

2. Explain rowspan and colspan with an example?

- Rowspan: Rowspan is a feature of HTML that is useful when we want a column in a multiple row table to extend downward into the next row.

Ex: <tr>

<td>January</td>

<td>$100</td>
<td **rowspan**="2">$50</td>

</tr>

- Colspan: The colspan attribute in HTML specifies the number of columns a cell should span. It allows the single table cell to span the width of more than one cell or column. It provides the same functionality as “merge cell” in the spreadsheet program like Excel. It can be used with <td> and <th> element while creating an HTML table.

Ex: </tr>

<tr>

<td **colspan**="2">Sum: $180</td>

</tr>

3. How to create hyperlinks in HTML with example?

To make a hyperlink in an HTML page, use the <a> and tags, which are the tags used to define the links. The <a> tag indicates where the hyperlink starts and the tag indicates where it ends. Whatever text gets added inside these tags, will work as a hyperlink.

Ex: team

[May/June 2016]

1. What is protocol? State any two protocols.

A protocol is a set of rules defined when two entities are communicating so that they can have uniformity, security, efficiency etc in their communication. While HTML is a markup language, the basic element of a web page.

- ❖ Two protocols are:
  - FTP, telnet, mailto, news, https, etc.

2. What is the use of <img>tag? Give the syntax of <img> tag.

The image tag is written as <img>. It is used to insert an images in an XHTML document. It is necessary to close the <img> tag in XHTML. It is done so by putting a (/) backslash after the word img i.e.<img/>.

Syntax: If the image is present in the local drive.

3. What are forms?

A form is simply an area that can contain form fields, form fields are objects that allow the visitor to enter the information.

Ex: text-boxes, drop-down menus or radio buttons.

4. What is MIME? Explain.

MIME, also known as Multipurpose Internet Mail Extensions, a specification for formatting non ASCII messages so that they can be sent over the Internet.

Syntax : main_mime_type/mime_subtype

Example : image/gif

[May/June 2017]

1. What is web browser? Mention any three web browsers.

A web browser is a software application used to locate and display web pages. It is able to retrieve, find, view, and send information over the internet.

- ❖ Web browsers are:

1. Internet Explorer
2. Netscape Navigator
3. Opera
4. Google Chrome
5. Mozilla Firefox
6. Safari

[May/June 2018]

1.What do you mean by IP address?

Every machine on the internet has a unique identifying number called an IP address. The IP stands for Internet Protocol, which is the language that computers used to communicate over the internet.

- IP address looks like 216.27.61.136

2.Define URL? Give an example.

Uniform Resource Locator or URL is a fancy name for address. It contains information about where a file is and what a browser should do while it. Each file on the Internet has a unique URL.

Ex: ftp://ftp.site.com/pub/prog.exe

When the user clicks this URL, the browser will begin an FTP transfer of the file prog.exe.

3.Give any two differences between HTML and XHTML.

HTML	XHTML
Tags aren't extensible	Tags are extensible
Tags are not case-sensitive	Only lowercase tags are allowed
Possible to leave off and ending tag like </body>	Tags should appear in pairs
Overlapping tags	No overlapping tags

[May/June 2019]

1. What is HTML event with example?

Interaction with HTML is handled through events that occur when the user or the browser manipulates a page. When the page loads, it is called an events. When the user clicks a button, that click too is an event.

Examples: Include event like pressing any key, closing a window, resizing a window etc.

2. Define <pre> tag with example?

The <pre> tag defines preformatted text. Text in a <pre> element is displayed in a fixed-width font and it preserves both spaces and line breaks.

Ex: <body>

<p>

```
<pre>
    Book            Price
  ---------------------------------
    WP              300
</pre>
</p>
</body>
```

UNIT II

[May/June 2015]

1. What are and tags and their uses?

 tag is a paired tag means it has both open(<) and closing (>) tag and it is mandatory to close the tag. The span tag is used to grouping of inline-elements.

 tag used in HTML to make divisions of content in the web page like (text, images, etc),div tag has both open( ) and closing ( ) tag and it is mandatory to close the tag.

2. What is universal selector or universal selector in CSS?

- The universal selector is an represented by asterisk(*).
- When it is used , the universal selector tells the CSS interpreter to apply the CSS rule to all elements in the document.
- Do not use Universal Selector unless it is mandatory.

[May/June 2016]

1. What is ID selector in CSS? How it will be represented.

ID selector are similar to class selector. It used to select any HTML element that has an ID attribute, regardless of their position in the document tree. # character followed by the ID of the element.

Example:
```
#para1{
    text-align: center;
    color: red;
}
```

[May/June 2017], [May/June 2018] AND [May/June 2019]

MENTIONED ABOVE QUESTIONS ARE REPEATED.

UNIT III

[May/June 2015]

1. What are the composite data types in Java script?

A composite data type in JavaScript has multiple values, which grouped together. It works the same as the object in the class.

[May/June 2016]

1. What do you mean by an object in java script?

The web page document is an object. Any table, forms, button, or link on the page is also an object, each object has an certain properties like background color etc.

[May/June 2017]

1. What do you mean by an Math and date object in java script?
   - Math object: Javascript Math object is used to perform mathematical tasks. Math object need not be defined. Math object in Javascript has two main attributes: Properties and Methods.
   - Date object: To display date in Javascript we use Date object. A Date object contains date information. The Date object is created using new operator . Methods are: getSeconds(), getMinutes(), getHours(),etc.

[May/June 2018]

1. What is constructor in java script?

A constructor is a function in JavaScript. It's not the function declaration what separates a function from constructor but the function invocation. A constructor is called with new keyword. A constructor names must start with a capital letter.

[May/June 2019]

1. Explain any 2 string methods in java script or methods of string in object?

1. charAt() : Returns the character at the specified index.

Ex: var b=' I am a Java script learner.'

document.write(b.charAt(5))

2. indexOf() : Returns the index with the calling string object of the first occurrence of the specified

value, or -1 if not found.

Ex: var a = 'Hello world!';

document.write(a.indexOf('w'))

UNIT VI

[May/June 2015]

1. What is the use of focus and blur events?

- The important events associated with textbox and password elements are onfocus and onblur.
- onfocus is when we give a focus to an element by clicking on it, makes it the active element.
- onblur is when something loses focus by clicking on tabbing out of it, or doing something that makes another element the active element.

2. What is the use of navigator object?

The Navigator object represents the particular browser and allows script functions to access information about the browser being used to access the document. The Navigator object provides information about the browser that the user is currently using. All the properties are read-only and cannot be dynamically changed.

[May/June 2016]

1. What are mouse events. Mention mouse events?

Java script programs have the ability to track mouse movement and button clicks as they relate to the web page and controls inside.

❖ List of Mouse events are:

1.mousedown	4.mousemove	
2.mouseup	5.mouseout	
3.mouseover	6.click	7.dbclick

[May/June 2017]

1. What is an event and event handler?

- Event: An event is something that triggers a specific action in the browser.
- Event handler: An event handler is a java script code that is designed to run each time a particular event occurs.

[May/June 2018] AND [May/June 2019]

MENTIONED ABOVE QUESTIONS ARE REPEATED.

UNIT V

[May/June 2015]

1. What is DHTML and explain?

DHTML stands for Dynamic Hyper Text Markup Language. DHTML is not a language or a web standard. DHTML is a term used to describe the technologies used to make web pages dynamic and interactive.

2. What is XML?

The eXtensible Markup Language is a text document used mainly for distributing the data on internet between different applications. It's a structured document for storing and transporting the data, mainly used for the interchanging the data on internet.

[May/June 2016]

1. What do you mean by relative positioning?

Relative positioning places the element in a position relative to the element where it is defined within the document. This type of positioning is used to control the way elements appear in relation to other element in the document.

[May/June 2017]

1. What is DTD? Mention its types.

A DTD is a Document Type Definition. A DTD defines the structure and the legal elements and attributes of an XML document.

- Types of DTD :
  1.DTD <!DOCTYPE>
  2.Internal and External DTD
  3.DTD Entities
  4. DTD Elements
  5.DTD Attributes
  6.DTD Examples

2. What is XML schema?

An XML Schema describes the structure of an XML document. The XML Schema language is also referred to as XML Schema Definition (XSD). The purpose of an XML Schema is to define the legal building blocks of an XML document.

1. the elements and attributes that can appear in a document.
2. the number of (and order of) child elements.
3. data types for elements and attributes.
4. default and fixed values for elements and attributes.

[May/June 2018]

1. What do you mean by namespaces?

A namespace is used as additional information to differentiate similar functions, classes, variables etc. with the same name available in different libraries. Using namespace, we can define the context in which names are defined. In essence, a namespace defines a scope.

2. What is XSLT style sheet?

XSLT is an xml based language for transforming XML documents into other XML or HTML or XHTML documents. XSLT stands for eXtensible Styles Language Transformations.

[May/June 2019]

1. What do you mean by absolute positioning of an element?

Absolute positioning is one way that an element can be positioned with CSS. If an element is positioned absolutely, the box of an element is taken out completely from the document's flow, it is placed in relation to its containing block.

SOUNDARYA INSTITUTE OF MANAGEMENT AND SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

WEB PROGRAMMING

SOLUTION BANK 2019-2020

SECTION – B (5 Marks)

UNIT I

[May/June 2015]

1. Explain GET and POST request methods?

GET request method:

GET is the most common HTTP request method. A client can use the GET request method to request for a piece of resource from an HTTP server.

A GET request message take the following syntax:

GET request-URL HTTP-version
(Optional request headers)
(Blank line)
(Optional request body)

- The keyword GET in case sensitive and must be in uppercase.
- Request-URL: specifies the path of resource requested, which must begin from the root "/" of the document base directory.
- HTTP-version: Either HTTP/1.0 or HTTP/1.1. This client negotiates the protocol to be used for the current session.
- GET request message does not have a request body.

POST request method:

- POST request method is used to post data up to the server. Issuing an HTTP URL from the browser always trigger a GET request. To register a POST request, we can use an HTML from with attribute method="post".
- POST request is same as the GET request except that the URL-encoded query string is sent in the request body, rather than appended behind the request-URL.

The POST request takes the following syntax:

POST request-URL HTTP-version
Content-Type: mime-type
Content-Length: number-of-bytes
(Other optional request headers)

(URL-encoded query string)

- Request headers content-type and content-length is necessary in the POST request to inform the server the MIME type request body.

2. Explain the basic table tags with different attributes with example?

Basic table tags:

There are three basic components of any table.

Table:<table>

Table row:<tr>(<tr> is always enclosed within <table>).

Table(data)cell:<td>(<td> is always enclosed within <tr>).

Example:

<table border=”1”>

<tr>

<td>Row 1 Cell 1</td>

<td>Row 1 Cell 2</td>

<td>Row 1 Cell 3</td>

</tr>

<tr>

<td>Row 2 Cell 1</td>

<td>Row 2 Cell 2</td>

<td>Row 2 Cell 3</td>

</tr>

</table>

Table Headers(<th>): Table heading can be defined using <th> element. This tag will be put to replace <td> tag which is used to represent actual data.

Example:

<table border=”1”>
<tr>
<th>Name</th>
<th>Class</th>
</tr>
<tr>
<td>Monika</td>

<td>I BCA</td>
</tr>
</table>

Table with border attribute: To display a table with border we need to specify the border attribute. The tags to add a border to a table is border=”X”. X is replaced with the size of the border we want ranging from 0 to 1000.

Example:

```
<table border=”5”>        // table with border value is 5
<tr>
<th>Name</th>
<th>Class</th>
</tr>
<tr>
<td>Monika</td>
<td>I BCA</td>
</tr>
</table>
```

Table width: Table width specifies the width for the entire table. To specify the width, we will need to add the lines width=”X” to the <table> tag.

Example:

```
<table width=”300” border=”2”>
<tr> <td> Cell 1 </td> Cell 2 </td> </tr>
<tr> <td> Cell 3 </td> Cell 4 </td> </tr>
</table>
```

3. What is the role of web server? Explain about Apache and IIS web servers.

The role of web servers is:

1. Web servers: the computer or the program have a vital role on the internet. The server machine hosts the web site while the server program helps deliver the web pages and their associated files like images and flash movies.

2. The process of loading a web site/page in a web browser starts with the user either entering the URL in the address bar or clicking on a link.

3. The browser broke the URL into three parts:

- The protocol(“http”)
- The server name(“www.irctc.co.in”)
- The file name(“index.html”)

Apache Http Server:

- Apache is available for a range of operating systems, including Unix, Linux, Novell Netware etc.
- It is open source product.
- The Apache web server can be controlled by configuring the httpd.conf file.

Internet Information Services:

- IIS works only in windows operating systems.
- It is vendor specific and it is not open source.
- The IIS server can be controlled by modifying the program called IIS snap in. We can access this program through Control Panel --> Administrative Tools.

4. Explain the attributes of <input> tag with an example?

- The input tag is the most commonly used tag within HTML forms. It allows to specify various types of user input fields such as text, radio buttons, checkboxes etc.
- The <input/> tag is used to define the controls for the form.
- The controls are the data fields or selectable options that allows the user to enter information into the form.

Example: <input attributes/>

The attributes of <input> tag are:

1. **Button:** This allows buttons to be included in the form that perform tasks other than submitting or resetting the form.

Ex: <input type=”button” value=”Button”/>

2. **Submit:** This creates a button that automatically submits a form.

Ex: <input type=”submit” name=”Submit” value=”Submit” />

3. **Reset:** This creates a button that automatically resets form control to their initial values.

Ex: <input type=”reset” values=”Reset”/>

4. **Checkbox:** This denotes a checkbox, a simple selectable element in a form.

Ex: <input type="checkbox" />

5. Radio: This denotes one of a set of radio buttons, which is a collection of selectable check boxes where only one in the group can be chosen.

Ex: <input type="radio" />

6. Text: This denotes a simple text entry field.

Ex: <input type="text"/>

7. Password: This denotes a text field where the content is masked, such as password fields.

Ex: <input type="password" />

8. File: This allows the user to select of files on their computer for submission to the server.

Ex: <input type="file" name="fileupload" />

5. What are the list in XHTML and different types of lists supported in XHTML with an example? Give an example to implement various types of lists.

Lists are frequently used for products offered by the site, or members currently on the site, or form controls.

Example: <list>

<item> item text </item>

</list>

Different types of lists are:

1. **Unordered list:**

- It is used to group a set of related items not in a particular order.
- Each bullet point or line we want to write should then be contained between opening <li> tags and closing tags
- We should always close the <li> and <ul> elements.

Example:

```
<html>
<head>
<title> Unordered list </title>
</head>
<body>
  <h4> My Grocery List</h4>
     <ul>
      <li>Sugar</lis>
      <li>Eggs</lis>
        <li>Rice </lis>
      </ul>
</body>
     </html>
```

2. **Ordered list:**

- Used to group a set of related items in a specific order.
- The ordered lists use the <ol> tag to delimit the entire list and the list item <li> tag to delimit each individual items.

Example:

```
     <html>
   <head>
     <title> Ordered list </title>
     </head>
     <body>
        <h4>  IT companies List  </h4>
             <ol>
                  <li>TCS</lis>
                   <li>Infosys</lis>
                    <li>Wipro</lis>
               </ol>
  </body>
</html>
```

3. **Definition list:**

- Used to display name/value pairs such as terms and definitions.
- Definition lists are contained inside the <dl> element.

- The definition list is a special kind of list for providing terms followed by a short text definition or description for them.

Example:

```
<html>
    <head>
        <title> Definition list </title>
    </head>
    <body>
        <dl>
            <dt> html </dt>
            <dd> the root element in every HTML document </dd>
            <dt> head </dt>
  </dl>
    </body>
</html>
```

[May/June 2016]

1. What is DNS? Explain domain and subdomain with suitable examples.

Domain Name System [DNS]:

DNS is a database system that translates a computer's fully qualified domain name into an IP address. The domain name system is used to resolve human readable hostnames, such as www.amazon .com, into machine readable IP addresses, such as 207.171.166.48.

DNS also provides other information about domain names, such as mail services.

Domain:

- The COM, ORG, EDU and UK portions of these domain names are called the top-level domain or first-level domain.
- Within every top-level domain there is a huge list of second level domain.

Example: yahoo, msn, Microsoft, etc..,

- Every name in the COM top level domain must be unique, but there can be duplication across domains.

Example: yahoo.com and yahoo.org

Sub-domains:

- Sub-domains are the third level domains that are used to organize the web site content in systematic manner. They are just like folders under the root directory. But they will have a special URL to access.
- **Example:**

http://www.yoursite.com is the regular URL without a sub-domain.
http://products.yoursite.com is an URL with sub-domain "products".

Here the:

com is the first level domain.
your site is the second level domain.
products is the third level domain.

2. What is a frame? Explain <frame> and <frameset> tags with its attributes and examples?

Frame:

- Frames allow a web page to be divided into several independent sections. A page using frames is actually made up of several .html files.
- There is one file for each frame on the page and one additional file that contains the frameset information

The FRAMESET tag:

- The <frameset> tag is used to specify the layout of the frames on the page. The document containing the frameset isn't visible. It simply specifies how the other documents will be arranged on the page.
- The <frameset> tag defines how to divide the window into frames. Each frameset defines a set of rows and columns.
- If we defines frame by using rows then horizontal frames are created. If we defines a frame by using columns then vertical frames are created.

Example: <frameset rows="40%,60%" cols="200,*">
Content of frames
</frameset>

The <frameset> tag attributes are:

Attributes	**Description**	**Example**
Columns(cols)	Specifies how many frames the window will be split into vertically and specifies the width of each frame. The width can be indicated in pixels or % of the available space.	<frameset cols="10%,80%,10%">

Rows	Specifies how many frames the window will be split into vertically and specifies the width of each frame. The width can be indicated in pixels or % of the available space.	<frameset rows=”50%,50%”>

The FRAME tag:

- Once we have used the <frameset>tag to define how many frames we will have, each of those frames will need to be defined with a <frame> tag.
- The <frame> tags are inserted between the start and end <frameset> tags.

Example:
```
<frameset rows=”100%” cols=”1*,400,2*”>
<frame src=”left.html”name =”left”/>
<frame src=”right.html”scrolling=”no”name=”right”/>
</frameset>
```

The <frame> tag attributes are:

Attribute	**Description**
frameborder	Specifies whether a three dimensional border should be displayed between frames. this attributes takes value either 1 (yes) or 0 (no). Example: frameborder=”0” [specifies no frame border]
marginheight	Determines the amount of space(in pixels) above and below the frames.
marginwidth	Determines the amount of space(in pixels) on either side of the frame.
name	Name of the frame. This is used as a reference for the hyperlinks. Names must begin with an alphabetical character (A to Z).
noresize	This attribute is used when the frame cannot be re-sized with the mouse.
Scrolling	Determines whether or not a scroll bar will appear in the frame. The choices are yes, no and auto.
src	Reference to the document that will initially be displayed in the frame.

3. Explain HTTP request method?

HTTP protocol defines a set of request methods. A client can use one of these request methods to an HTTP server.

The methods include:

- **GRT**: A client can use the GET request to get a web resource from the server.
- **HEAD**: A client can use the HEAD request to get the header that a GET request would have obtained.
- **POST**: Used to post data up to the web server.
- **PUT**: ask the server to store the data.
- **DELETE**: Ask the server to delete the data.
- **TRACE**: Ask the server to return a diagnostic trace of the actions it takes.
- **OPTIONS**: Ask the server to return the list of request methods it supports.
- **Connect**: Used to tell a proxy to make a connection to another host and simple reply the content, without attempting to parse or cache it. This is often used to make SSL connection through the proxy.

4. Explain any 7 XHTML tags with example.

Tags	**Description**	**Example**
<body>	Defines the body element.	<body> The content of the document. </body>
	Defines the centered text.	<**center**>This text will be center-aligned
	Defines a section in a document.	style=”color:#0000FF”> This is a heading This is a paragraph.
<h1> to <h6>	Defines a header 1 to 6.	This is heading 1 This is heading 2 This is heading 3 This is heading 4 This is heading 5

<head>	Defines information about the document.	<html> <head> <title>Web Programming</title> </head> <html>
	Defines a list item.	 - Coffee - Milk
	Defines a paragraph.	<**p**>This is some text in a paragraph.

5. What is form? What are the major attributes of the form? Explain form components with example.

Form:

- A frame is simply an area that can contain form fields.
- Form fields are objects that allows the visitors to enter information

Example: text boxes, drop-down menus or radio buttons.
<form> ….</form>

The major attributes of form are:

1. **Action:** The action attribute tells the browser where to send the form content when we submit the form. Usually, the value of the action is a page or program on a web server that will receive the information.
   Example:
   <form action=http://www.skyword.com//login.java>
   The action attribute in this example sends the data to a file called login.java. The java file process and validates the data and send the result to the browser.

2. **Method:** The method attribute tells the browser how to send the form content when we submit the form

Example:

- If the method value is "get" then it looks like this :

<form action="http://www.skyword.com//login.java"method="get">

In this case, the form data will be sent through the URL.

- If the method value is “post” then it looks like this:

<form action=”http://www.skyword.com//login.java”method=”post”>

When the form method is set to POST, the browser sends a separate server request containing some special headers followed by the data.

The components of form are:

1. **Buttons:**

Example:

```
<form action="index.htm" method="get">

<p>

<input type="submit" name="B1" " id="B1"value="Go to Contents" />

</p>

</form>
```

2. **Images:**

Example:

```
<form method="get" action="index.htm">

<input src="home.gif" name="I1" alt="Contents of HTML Tutorial"
width="72" height="41" type="image" />

</form>
```

3. **Labels:**

Example:

```
<form name>
```

<label accesskey="d" for="myText">Label Example: </label>

<input type="text" id="myText" value="Select me with alt+d" size="20">

</form>

[May/June 2017]

1. What are the different levels of headings in HTML?

- Any document starts with a heading. We can use different sizes for the headings.
- XHTML supports six levels of headings, which use the elements <h1>,<h2>,<h3>,<h4>,<h5>, and <h6>. While displaying any headings, browsers adds one line before and after that headings.
- Each headings uses a large, usually bold characters formatting style to identify itself as a heading.

Example:

```
<body>
        <h1> This is heading 1 </h1>
        <h2> This is heading 2 </h2>
        <h3> This is heading 3</h3>
        <h4> This is heading 4</h4>
        <h5> This is heading 5</h5>
        <h6> This is heading 6 </h6>
</body>
```

[May/June 2018]

MENTIONED ABOVE QUESTIONS ARE REPEATED

[May/June 2019]

1. Explain any 5 network protocol?

Network protocols are:

1. Transmission Control Protocol (TCP): TCP is a popular communication protocol which is used for communicating over a network. It divides any message into series of packets that are sent from source to destination and there it gets reassembled at the destination.

2. Internet Protocol (IP): IP is designed explicitly as addressing protocol. It is mostly used with TCP. The IP addresses in packets help in routine them through different nodes in a network until it reaches the destination system.

3. Post Office Protocol (POP): POP3 is designed for receiving incoming E-mails.

4. Simple Mail Transport Protocol (SMTP): SMTP is designed to send and distribute outgoing E-mail.

5. File Transfer Protocol (FTP): FTP allows user to transfer files from one machine to another. Types of files may include program files, multimedia files, text, and documents.

UNIT II

[May/June 2015]

1. What are the different levels of CSS? Explain with examples.

There are three levels of CSS they are:

1. **Inline Styles:**

- It is possible to create CSS styles that only work for the single tag it is defined for. Single tag CSS is used when the style is used in a single place on the entire site.
- The inline styles are applied to each individual element using style attribute of the tag.
- Applying style to each individual element increases the code and it is difficult to maintain.

Example:

 Web Programming

In the above code, we have used style attribute for (bold) tag. This tells the browser to display "Web Programming" in bold and font size 16px.

2. **Internal or Document Styles:**

- The internal or document level styles are used to define the style for whole page. This works only for the page it is defined.
- To apply internal CSS, we need to put all CSS rules in the <head>….<//head> part of the XHTML document and enclosed it with <style type=”text/css”>….</style> tags

Example:

```
<head>
<style type=”text/css”>
h1 {color:red;}
p {color:yellow;}
body {background-color: black;}
</style>
</head>
```

3. **External Styles:**

- External style is used more than a single page. Adding styles that are defined once for the entire site.
- The external style sheets can be defined for entire web pages by simply writing the CSS definition in a plain text file(file with extension .css) that is referred to form each of the pages in the web site.
- To include one or more style sheets into the XHTML document we can use the <link> tag

Example:

<link rel=”stylesheet” type=”text/css” href=”/style.css”/>

2. Explain types of selectors in CSS?

Selectors: in CSS, a selectors is the target element or elements to which each CSS rule is applied.

The different types of selectors are:

1. **Type selectors:** The most common and easy to understand selectors are type selectors. Type selectors will select any HTML element on a page that matches the selectors, regardless of their position in the document tree.

   Example: em{color:blue;}

   this rule will select any element on the page and color it blue.

2. **Class selectors:** Class selectors can used to select any XHTML element that has a class attribute, regardless of their position in the document tree.
   - The class selectors are created by a period followed by the class name.

3. **ID selectors:** ID selectors is an individually identified selectors to which a specific style is declared.
   - Using the ID attribute the declared style can then be associated with one and only one XHTML element per document as to differentiate it from all other elements.
   - ID selectors are created by a character # followed by the selector's name

4. Descendant selectors: Descendant selectors are used to select elements that are descendants of another element in the document tree.
   - For example, we may wish to target a specific element on the page, but not all elements.

4. **Child selectors:** A child selector is used to select an element that is a direct child of another element (parent). Child selectors will not select all descendants, only direct children.

5. **Universal selectors:** Universal selectors are used to select any element. For example, the rule below will color all XHTML elements on a page blue regardless of their position in the document tree.

```
*{color:blue;}
```

6. **Adjacent sibling selectors:** Adjacent sibling selectors will select the sibling immediately following an element.

7. **Attribute selectors:** Attribute selectors are used to select the elements based on their attribute and value.

Attribute selectors are the selectors defined by :

1) the attribute se to element(s)
2) the attribute and value(s)
3) the attribute and value parts.

8. **Pseudo – classes :** Pseudo classes are used when we may want to style something where there is no CSS selector available, like the state of a hyperlink (eg.active or visited).

9. **Pseudo-elements:** Pseudo elements allows to style information that is not available in the document tree. For instance, using standard selectors

there is no way to style the first letter or first line of an element's content.

[May/June 2016]

1. What is CSS? What are the advantages and disadvantages of using CSS in XHTML?

CSS: CSS (cascading style sheet) is a language that applies style on a HTML document and its element to change the look and feel and is usually stored in separate .css style sheet which can be re used for all the web pages.

Advantages of CSS:

- **CSS saves time**: we can write CSS once and then reuse same sheet in multiple HTML pages.
- **Pages load later**: if we are using CSS, we do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply to all the occurrence of that tag. So less code means faster download time.
- **Easy maintenance**: to make global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Smaller file size**: separating content from presentation helps keep HTML file sizes down and improves page loading times.
- **Consistency**: with CSS it is easy to keep a consistent look throughout the website.
- **Accessibility**: having content separated from presentation is a great help t people with viewing disabilities who often benefits from well structured HTML to understand the page.
- **Design without tables**: with CSS we can position any elements exactly where we want without using tables.
- **Superior styles to HTML**: CSS has a much wider array of attributes than HTML so we can give far better look to the HTML page in comparison of HTML attributes.
- **Multiple device compatibility**: style sheets allow content to be optimized for more than one type of device.
- **Global web standards**: it is a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

Disadvantages of CSS:

Browser compatibility: browser have varying levels of compliance with style sheets. This means that some style sheet features are supported and some are not

supported. To confuse things more, some browsers manufacturers decide to come up with their own proprietary tags.

[May/June 2017]

MENTIONED ABOVE QUESTIONS ARE REPEATED

[May/June 2018]

1. Differentiate and tags with examples.

** tag:**

- div(short for division) divides the content into individual sections. Each section can then have its own formatting, as specified by the CSS. Div is a block-level container, meaning that there is a line feed after the tag.
- The XHTML div tag is used for defining a section of the web page. With the div tag, we can group large sections of XHTML elements together and format them with CSS.
- Div tag is used with block level content.

Example:

```
div.relative {
   position: relative;
   left: 30px;
   border: 3px solid #73AD21;
   }
```

** tag:**

- The XHTML tag is used for grouping and applying styles to inline elements.
- Span tag is used with inline elements.
- The span tag goes into a finer level, so we can span to format a single character if needed.

Example:

```
<p>My mother has <span style="color:blue">blue</span> eyes.</p>
```

2. What is selector in CSS? Distinguish Class and ID selectors with examples.

Selectors: in CSS, a selectors is the target element or elements to which each CSS rule is applied.

Class selectors:

- Class selectors can used to select any XHTML element that has a class attribute, regardless of their position in the document tree.
- The class selectors are created by a period followed by the class name.

Example:

```
    <body>
  <p class="big">this is some <em>text</em></p>
  <p> this is some text </p>
  <ul>
    <li class="big"> list item </li>
    <li> list item </li>
    <li> list <em> item </em></li>
  </ul>
</body>
```

ID selectors:

- ID selectors is an individually identified selectors to which a specific style is declared.
- Using the ID attribute the declared style can then be associated with one and only one XHTML element per document as to differentiate it from all other elements.
- ID selectors are created by a character # followed by the selector's name
  #navigation {width: 12em; color: #333}

Example: Name of desired element

```
Class
em.very {color:red;}

div #gaudi {color:red;}
ID
Name of desired element
```

3. Explain CSS box model.

CSS box model: The term "box model" is about CSS-based layouts and design. The box model is term used when referring to the rectangular boxes placed around every elements in the web page.

- The box model is the specification that defines how a box and its attributes are related to each other.

The different parts of Box Model are:

1. **Content Area:**
   - the innermost part of the box is the xhtml element itself or content, such as “ ”,”<p>”,”<h1>”,”<ul>”….etc.
   - The CSS width and height property defines the width and height of this element.

Example:

CSS code:

```
P {width:100px; height:50px}
```

XHTML code:

```
<p> this is content area </p>
```

2. **Padding:**

   - Padding property is the first layer we would encounter going outwards from the actual element covered in the previous point.
   - The padding defines the space between the content area of the box and the border.

   Example:

```
P {padding: 30px 20px 30px 20px;}
```

The above code adds 30 pixels to the top and bottom, and 20 pixels to the left and right padding

3. **Border:**
   - The middle layer in the box model is the element’s border. The space used by the border in the box model is the thickness of the border.
   - The border outlines the visible portion of the element.

   Example:

```
P {border: 5px solid #EFD5B4;}
```

The above code add a solid 5 pixel thick border with #EFD5B4 color

4. **Margin:**

Margin is the space just outside the border. The margin completely invisible. It has no background color, and will not contain any elements behind it.

Example:

```
P {margin: 30px;}
```

The above add a 30 pixels margin around the element.

[May/June 2019]

MENTIONED ABOVE QUESTIONS ARE REPEATED

UNIT III

[May/June 2015]

5. Explain the various looping statements in javascript.

The various looping statements in javascript are:

- **While loop:** In a while loop, a code block is executed if a condition is true and executes as long as that condition remains true.

Syntax:

```
While(condition)
{
    Code to be executed
}
```

Example:

```
<script type=”text/javascript”>
i=1;
while(i<11){
document.write(i+”x4=”+(i*4)+”<br/>”);
i++;
}
</script>
```

- **do…while loop:** A do while loop executes a block of code once and then checks a condition. As long as the condition is true it continues to loop.

Syntax:

```
do{
```

code to be executed

}while(condition)

Example:

<script type=”text/javascript”>

i=12;

do{

document.write(i+”x4=”+(i*4)+”
”);

i++;

}

while(i<11)

</script>

- **for loop:** The for statement executes a block of code a specified number of times.

Syntax:

for(startingValue;condition;iterationValue){

Javascript statements that we want to execute repeatedly

}

Example:

<script type=”text/javascript”>

for(i=0;i<11;i++){

document.write(i+”x4=”+(i*4)+”
”);

}

</script>

6. Create a form for student information. Write a java script code to find total, average, result and grade.

<?xml version = “1.0” encoding=”ISO-8859-1”?>

<!DOCTYPE html PUBLIC”.//W3C//DTD XHTML 1.0 Transitional//EN” “http://www.w3.org/TR/xhtml1/DTD/xhtml-transitional.dtd”>

```
<html xmlns=http://www.w3.org/1999/xhtml>
<head>
<title>Student Data Example</title>
<script type = “text/javascript”>
function showResults(){
var name=document.getElementById(“name”).value;
var cls=document.getElementById(“class”).value;
var marks1=parseInt(document.getElementById(“sub1”).value);
 var marks2=parseInt(document.getElementById(“sub2”).value);
  var marks3=parseInt(document.getElementById(“sub3”).value);
 var total=marks1+marks2+marks3;
var avg=total/3;
var grade, result;
if(avg>=60){
grade=”A”;
result=”First Class”
}
else if(avg<60 && avg>=50){
grade = “B”;
result = “Second Class”
}
else if(avg<50 && avg>=40){
grade = “C”;
result = “Third Class”
}
else {
grade = “D”;
```

```
result = “Fail”;
}
document.write("<body bgcolor=pink>"); document.write("<h2>Student Marks Report </h2>");       document.write("<hr>");
document.write(“<b>Name :”+name+”</b><br/><br/>”);
document.write(“<b>Class :”+cls+”</b><br/></br>”);
document.write(“<b>Total Marks :”+total+”</b><br/></br>”);
document.write(“<b>Average :”+avg+”</b><br/></br>”);
document.write(“<b>Grade:”+grade+”</b><br/></br>”);
document.write(“<b>Result :”+result+”</b><br/></br>”); } </script>
</head>
<body>
<form>
<table border=”5”>
<tr><th>Student Details</th></tr>
 <tr>
<td> Student Name: </td>
<td> <input type=”text” id = “name”/></td>
</tr>
<tr>
<td>Subject1 Marks</td>
<td><input type = “text” id = “sub1”/ ></td>
</tr>
<tr>
<td> Subject2 Marks</td>
<td>input type = “text”    id = “sub2”/></td>
</tr>
<tr>
<td>Subject3 Marks</td>
```

<td><input type = “text” id = “sub3”/></td>

</tr>

</table>

<input type = “button” value = “view results” onclick = “showResult()”/>

</form>

</body>

</html>

[May/June 2016]

1.Explain screen output and keyboard input methods in java script.

- **Methods to show output:**

1.document.write() or document.writeln(): These two methods is used for displaying text on a web page.

Example for write method:

<script type=”text/javascript”>

document.write(“<pre>Line 1”);

document.write(“Line 2</pre>”);

</script>

Example for writeln method:

<script type=”text/javascript”>

document.writeln(“<pre>Line 1”);

document.writeln(“Line 2</pre>”);

</script>

2.window.alert(): The alert method is part of window object. The alert() method show the message in a small message box. This method is used to send a warning to the user or alert him or her to do something.

Example:

<script type=”text/javascript”>

Window.alert(“Your Email Id is Incorrect.”)

</script>

3.windows.status: The status is not a method and it is property of window object. The window.status property show the message in status bar. The message to be displayed has to be assigned to the window.status property.

Example:

<script type=”text/javascript”>

Window.status=“It is my own webpage.”;

</script>

- ❖ **Methods for taking Input from user:**

1.window.prompt(): A promt() method asks the user for some small amount of information such as a password, completion of a input, personal information such as nick name or title. Javascript does not provide a simple method for accepting user input, the prompt dialog box and XHTML forms are used.

Example:

<script type=”text/javascript”>

var name=window.prompt(“What is your name?”,””);

alert(“Welcome to my web page!.....”+name);

</script>

2.window.confirm(): The confirm dialog box is used to confirm a user’s answers to a question. The user must agree before the action is completed.

Example:

<script type=”text/javascript”>

var canFormat=window.confirm(“Do you want to format the hard disk?”);

window.alert(“Your hard disk is formatted”);

</script>

2.What are the different ways that we can include

Java script to a XHTML document?

There are four standard ways to include script in an XHTML document:

1.Within the <script> element.

2.As a linked file via through the src attribute of the <script> element.

3.Within an XHTML event handler attribute such as onclick.

4.Via the pseudo-URL javascript.

- The <script> element, the primary method to include javascript within HTML or XHTML is the <script> element. A script-awarebrowser assumes that all text within the <script> tag is to be interpreted as some form of scripting language, by default it is a javascript.

  Example:

  <script language=”text/javascript”>

  </script>

- The Linked scripts, A very important way to include a script in an HTML document is by linking it via the src attribute of a <script> tag.
- Javascript Pseudo-URL, it is possible to invoke a script using the javascript pseudo-URL. A pseudo-URL like javascript:alert(‘hello’) would invoke a simple alert displaying “hello” when typed directly in the browser’s address bar.
- Script in the <head>, A special location for the <script> element is within the <head> tag of an XHTML document. Because of the sequential nature of Web documents, the <head> is always read in first, so scripts located here are often referenced later on by script in the <body> of the document.

3.What is an array object? How to create an array object with an example? What is an array in java script? Explain any six array methods.

Array object: The Array object store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is a collection of variables of the same type.

Creating an Array object:

The new keyword is used to dynamically create the Array object. It calls the Array object’s constructor function, Array(), to create a new Array object.

Example: var myArray=new Array();

Array in Javascript:

- An array is a special variable, which can hold more than one value at a time.
- Creating an Array. Using an array literal is the easiest way to create a Javascript Array.
- Using the Javascript keyword new, can create a array.

Array Methods are:

Method	Description
concat()	Concatenates elements from one array to another array.
join()	Joins the elements of an array by a separator to form a string.
pop()	Removes and returns the last element of an array.
push()	Adds elements to the end of an array.
reverse()	Reverse the order of the element in an array.
shift()	Removes and returns the first element of an array.

[May/June 2017]

1.What is Javascript? What are the data types supported by java script.

Javascript is a cross platform, object based scripting language invented specifically for use in web browsers to make websites more dynamic and attractive. Javascript is mainly used as a client side scripting language.

The data types supported by Javascript are:

1.Numbers: A number type refers to numerical values. Numbers can be divided into two groups:

1.Floating point: are fractional numbers such as 2.45, -3.58 and .97

2.Integers: are whole numbers such as 245, -450 and 3.

2.Strings: A string type refers to one or more characters of text. In javascript, a string is enclosed inside single or double quotes.

Example: “Scripting language”, “XHTML”, “2020”

3.Booleans: Boolean logic allows the program to make decisions. A Boolean logic statement consists of a condition that evaluates to either true or false. Conditions as a question that has one of two possible answers:

1. yes or no

2. ON or OFF

3. positive or negative

4. true or false

4.Undefined type: The undefined type refers to those variables or object properties that are either undefined or do not exist.

Example: var MyAge;

5.Null type: The null type indicates an empty value. When a variable is assigned the value null, its type becomes null. A null type variable is a placeholder that represents nothing.

Example: var MyAge=null;

[May/June 2018]

1.Write a java script program to implement all string operations?

```
<html>
<body>
<h1>String Operation Program</h1>
<p id=”demo”></p>
<script>
var txt=”String Method”;
var str=”Please locate where ‘locate’ occurs”;
var pos=str.indexOf(“locate”);
var pos=str.search(“locate”);
var str=”Apple,Banana,Kiwi”;
var res=str.slice(7,13)
document.getElementById(“demo”).inner HTML=txt.length;
document.getElementById(“demo”).inner HTML=pos;
document.getElementById(“demo”).inner HTML=pos;
```

document.getElementById(“demo”).inner HTML=res;

</script>

</body>

</html>

2.What is function give the general syntax of a java script function with example.

A function is series of commands that either calculates a value or performs an action. It consists of function name, parameters, the values passed to the function and the set of commands that run when the function is called.

The general syntax of a Javascript function is:

function functionName (parameters) {

    Javascript commands

}

where,

functionName = name of the function

parameters = values sent to the functions

Javascript commands = commands that run when the function is called or used

Example:

<html>

<head>

<title>Function Demo</title>

<script type=”text/javascript”>

function isEven(num){

if(num%2==0){

return num;

}

}

function printMessage(num){

```
if(num>1){
document.write(num+"");
}
}
var evenNum;
document.write("Even Numbers:");
for(i=1;i<=20;i++){
evenNum=isEven(i);
printMessage(evenNum);
}
</script>
</head>
</html>
```

3.Explain pattern matching using regular expression in java script?

1. The match() Method: This method is used to search for a pattern of characters in a string and returns an array where each element of the array contains each matched pattern that was found. If no match is found, returns null. With the g flag, match() search globally through the string for all matching substrings.

Example:

```
<script type="text/javascript">
var matchArray=new Array();
var string="I love the smell of clover"
var regex=/love/g;
matchArray=string.match(regex);
</script>
```

2. The search() Method: This method is used to search for a pattern of characters within a string, and returns the index position of where the pattern was found in the string. The index starts at zero. If the pattern is not found,-1 is returned.

Example:

<script type=”text/javascript”>

var myString=”I love the smell of clover”

var regex=/love/;

var index=myString.search(regex);

document.write(“Found the pattern”+regex+”at position”+index+”
”);

</script>

3.The replace() Method: This method is used to search for a string and replace the string with another string. The i modifier is used to turn off case sensitivity and the g modifier makes the replacement global. The replace() method is also used with the grouping metacharacters.

Example:

<script type=”text/javascript”>

var myString=”Tommy has a stomatch ache”

var newString=myString.replace(regex,”Mom”);

document.write(newString+”
”);

</script>

4.The split() Method: This method splits a single text string into an array of substrings. If the word in a string are separated by commas, then the comma would be the delimiter and if the words are separated by colons, then the colon is the delimiter. The delimiter can contain more complex combinations of characters if regular expression metacharacter is used.

Example:

<script type=”text/javascript”>

var splitArray=new Array();

var string=”apples:grapes:banana:plums:oranges”;

var regex=/:/;

splitArray=string.split(regex);

for(i=0;i<splitArray.length;i++){

document.write(splitArray[i]+”
”);

```
}
</script>
```

4.Write a java script program to find factorial of a number.

```
<html>
<head>- - - - -</head>
<script type=”text/javascript”>
function factorial(n)
{
    If (n==0)
      return 1;
    else
      return n*factorial(n-1);
}
Var num;
num=parseInt(window.prompt(“Enter Number”));
window.alert(“Factorial of a number “+num+”is:”+factorial(num));
</script>
</html>
```

[May/June 2019]

1.Write a java script code to evaluate mathematical expression and display the result.

```
<?xml version = “1.0” encoding=”ISO-8859-1”?>
<!DOCTYPE html PUBLIC”.//W3C//DTD XHTML 1.0 Transitional//EN”>
http://www.w3.org/TR/xhtml1/DTD/xhtml-transitional.dtd>
<html xmlns=”http://www.w3.org/1999/xhtml”>
<head>
<script type="text/javascript">
function Evaluate()
```

{

var enteredExpr=document.getElementById("expr").value;
document.getElementById("result").value=eval(enteredExpr); } </script>

</head>

<body bgcolor=cyan>

<h1 align=center> Evaluating Arithmetic Expression</h1>

<hr>

<form name = “myForm”>

Enter any Valid Expression :<input type="text" id="expr"/>

 <input type="button" value="Evaluate" onclick="Evaluate()"/>

Result of the Expression : <input type="text" id="result"/>

</form>

</body>

</html>

UNIT VI

[May/June 2015]

1.Write a short note on DOM?

The Document Object Model (DOM) is an application programming interface (API) for XHTML documents. The DOM represents a document as a hierarchical tree of nodes, allowing developers to add, remove and modify individual parts of web page.

The DOM allows a developer to access the document via a common set of objects, properties, methods, events and to alter the contents of the web page dynamically using scripts.

Levels of DOM:

1. DOM 0: All browsers have a Document Object Model, which gives you access to various parts of the document. The Level 0 DOM is the oldest of them. It was implemented in Netscape 2 and 3 and still works in all java script browsers. It gives easy access to forms and their elements, images and links.

2. DOM 1: The initial DOM standard, known as DOM Level 1, was recommended by W3C in 1998. DOM Level 1 provided a complete model for

an entire HTML or XML document, including means to change any portion of the document.

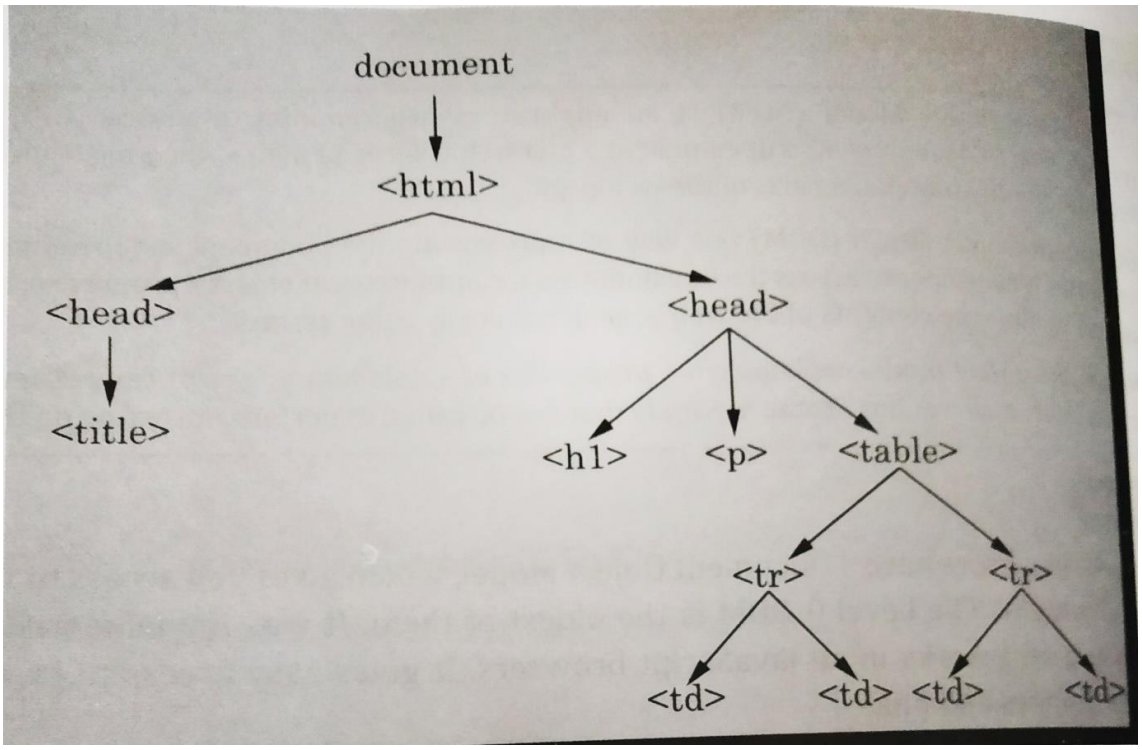
3. DOM 2: Level 2 is complete and many of the properties, methods, and events have been implemented by today's browsers. It has sections that add specifications for events and style sheets to the specification for core and HTML specific properties and events.

4. DOM 3: Level 3 archived recommendation status in 2004. It is intended to resolve a lot of the complications that still exists in the event model in Level 2 of the standard and adds support for XML features such as content models and being able to save the DOM as an XML document. Only few browsers support some features of Level 3.

Example of DOM:

```
<!DOCTYPE HTML>
<html>
<head>
 <title>About DOM</title>
</head>
<body>
 The truth about DOM.
</body>
</html>
```

Tree representation of DOM:



2.What are the different ways of registering an event handler.

- One of the way is assigning the event handler script to an event tag attribute.

Example: <a href=”books.html”onClick=”alert(‘I have clicked book page!’)”>

Now the script alert(‘I have clicked book page!’) is executed whenever the click event takes place on this link. It is registered as an event handler.

- When we want more than one statement to be executed when an event occurs. We can create user defined function. We can call the function when an event occurs.

Example: <input type=”click” onClick=”myFunction();”>

Now the function myFunction() is executed whenever the button is submitted in the form.

- The event handler could also be registered by the assignment to the associated event property on the button object.

Example:
document.getElementById(“myButton1”).onClick=myFunction;

3.Explain different ways of element access in java script with examples.

The different ways of element access in java script are:

1.Using forms and elements array of document object: This is the original way (DOM 0) to access the elements using forms and elements array of document object.

Syntax: document.forms[number].elements[number]

Example:
```
<script type=”text/javascript”>
    function function1(){
    var Name = document.forms[0].elements[0].value;
    var Age = document.forms[0].elements[1].value;
    var Phone = document.forms[0].elements[2].value;
    }
</script>
```

2.Using form names: It’s better to use the names of the forms and elements. In XHTML, we have to give a name to each form and each element.

Example:
```
<form name=”firstForm”>
    <input type=”text”name=”NameText”/>
    <input type=”text”name=”AgeText”/>
    <input type=”text”name=”PhoneText”/>
    <input type=”submit”name=”myButton”/>
</form>
```

- And it can access by these elements by

  Document.firstForm.NameText

  Document.firstForm.AgeText

  Document.firstForm.PhoneText

  Document.firstForm.myButton1

3.Using getElementById() method: By combining the XHTML id attribute with the getElementById() method of the document object, it is easier to get a handle on any XHTML object. This method takes the id of an XHTML element as its argument and returns a reference to that element.

Example: <form name="firstForm">

<input type="text"id="NameText"/>

<input type="text"id="AgeText"/>

<input type="text"id="PhoneText"/>

<input type="submit"id="myButton"/>

</form>

- And it can access by:

var name_element=document.getElementById("NameText");

var age_element=document.getElementById("AgeText");

var phone_element=document.getElementById("PhoneText");

var submit_element=document.getElementById("myButton1");

4.What are events? Explain different categories of events in XHTML.

Event: An event is something that triggers a specific action in the browser.

Different categories of events in XHTML are:

1.User Interface Events: It deals exclusively with the transfer of focus from one object inside the web pages to another.

Event name	**Event handler name/Tag attribute**	**Applicable tags**
focus	onfocus	<a>,<input>,<textarea>,<select>
blur	onblur	<a>,<input>,<textarea>,<select>,<button>

2.Mouse Events: The seven mouse events listed in the following table all relate to actions taken by the user using the mouse. Java script programs have the ability to track mouse movement and button clicks as they relate to the web pages and controls inside.

Event name	**Event handler name/Tag attribute**	**Applicable tags**
Mousedown	onmouse down	Most of the tags support mouse events
Mouseup	onmouseup	Most of the tags support mouse events

Mouseover	onmouseover	Most of the tags support mouse events
Mousemove	onmousemove	Most of the tags support mouse events
Mouseout	onmouseout	Most of the tags support mouse events
Click	Onclick	Most of the tags support mouse events
Dblclick	Ondblclick	Most of the tags support mouse events

3.Key Events: In a few situations where we might want to use keyboard events, this type of events handling is more commonly found in Windows applications. There are three main key events in HTML.

Event name	**Event handler name/Tag attribute**	**Applicable tags**
keypress	onkeypress	<body>,form elements
Keydown	onkeydown	<body>,form elements
keyup	onkeyup	<body>,form elements

4.XHTML Events: It means any events that do not belong in the user interface, mouse, or key event categories. Some XHTML events are triggered directly by the user action, while others are fired only as in indirect result of a user action.

Event name	**Event handler name/Tag attribute**	**Fires When**	**Applicable tags**	
load	onload	Browser finishes loading document	<body>	
unload	onunload	Browser about to unload document	<body>	
submit	onsubmit	Form about to submitted	<form>	
reset	onreset	Form about to be reset	<form>	
select	onselect	Text box contents selected	<input>,<textarea>	
change	onchange	Form control contents changed	<input>,<textarea>,<select>	

[May/June 2016]

1.Write a java script code to find sum of N natural numbers using user defined functions.

```
<?xml version=”1.0”encoding=”ISO-8859-1”?>
<!DOCTYPE html PUBLIC”//W3C//DTD XHTML 1.0 Transitional//EN”
http://www.w3org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>
<html xmlns=”http://www.w3org/1999/xhtml”>
<head>
<title>Sum of Natural Numbers</title>
<script type=”text/javascript”>
function sum(){
var num=window.prompt(“Enter the value of N”);
var n=parseInt(num);
var sum=(n*(n+1))/2;
window.alert(“Sum of First “+n+”Natural Number is: ”+sum);
}
</script>
</head>
<body>
<form>
<input type=”button”value=”Find Sum of Natural Numbers”onClick=”sum()”/>
</form>
</body>
</html>
```

[May/June 2017]

1.Explain XHTML events?

XHTML Events: It means any events that do not belong in the user interface, mouse, or key event categories. Some XHTML events are triggered directly by the user action, while others are fired only as in indirect result of a user action.

Event name	Event handler name/Tag attribute	Fires When	Applicable tags	
load	onload	Browser finishes loading document	<body>	
unload	onunload	Browser about to unload document	<body>	
submit	onsubmit	Form about to submitted	<form>	
reset	onreset	Form about to be reset	<form>	
select	onselect	Text box contents selected	<input>,<textarea>	
change	onchange	Form control contents changed	<input>,<textarea>,<select>	

Load event: The onload event handler is called when the HTML document finishes loading into the browser window.

Unload event: The unload event fires when a browser is leaving the current document.

Submit event: The submit event occurs just before a form is submitted to a web server.

Reset event: The reset event occurs when the reset button on a form is clicked.

Select event: The select event occurs when the user selects text inside a text box or text area.

Change event: The change event occurs when a control loses focus and its value has been altered.

[May/June 2018]

1.Write a java script program to implement keyboard events?

<html>

<head>

<script>

```
function myFunction(element, clr) {
element.style.color = clr;
}
</script>
</head>
<body>
<h1 onkeypress="style.color='red'" onkeydown="style.color='black'">Get the Key press to this text</h1>
<h1 onkeypress="myFunction(this,'red')" onkeydown="myFunction(this,'green')">
Click the text to change the color.<br/>A function, with parameters is triggered when the key button is down , and again,<br/> with other parameters, when the key button is pressed.
</h1>
</body>
</html>
```

[May/June 2019]

MENTIONED ABOVE QUESTIONS ARE REPEATED

UNIT V

[May/June 2015]

1.Write the differences between XML schema and DTD?

DTD	XML Schema
Own format	XML format
Compact notation	Very verbose
Simple data types	Advanced data types
From SGML	Invented for XML
Support entities	Does not support entities
No support namespaces	Support namespaces

2.What are the different types of positioning techniques? Explain.

The different types of positioning techniques are:

1.Absolute positioning: An element with position, absolute is positioned at the specified coordinates relative to your screen top-left corner.

Example:

```
<html>

<head>- - - - - - -</head>

<body>

<div style = "position:absolute; left:80px; top:20px; background-color:yellow;">

    This div has absolute positioning.

</div>

</body>

</html>
```

2.Relative positioning: Relative positioning changes the position of the HTML element relative to where it normally appears.

Example:

```
 <html>

 <head>- - - - - -</head>

 <body>

 <div style = "position:relative; left:80px; top:2px; background-color:yellow;">

    This div has relative positioning.

 </div>

</body>

</html>
```

3.Fixed positioning: Fixed positioning allows you to fix the position of an element to a particular spot on the page, regardless of scrolling. Specified coordinates will be relative to the browser window.

Example:

```
<html>

<head>- - - - - - -</head>
```

```
<body>

<div style = "position:fixed; left:80px; top:20px; background-color:yellow;">

        This div has fixed positioning.

</div>

</body>

</html>
```

4.Static positioning: The default or 'natural' position of block elements on the page. This is the normal positioning scheme in which elements are positioned as they occur in the normal document flow.

Example:

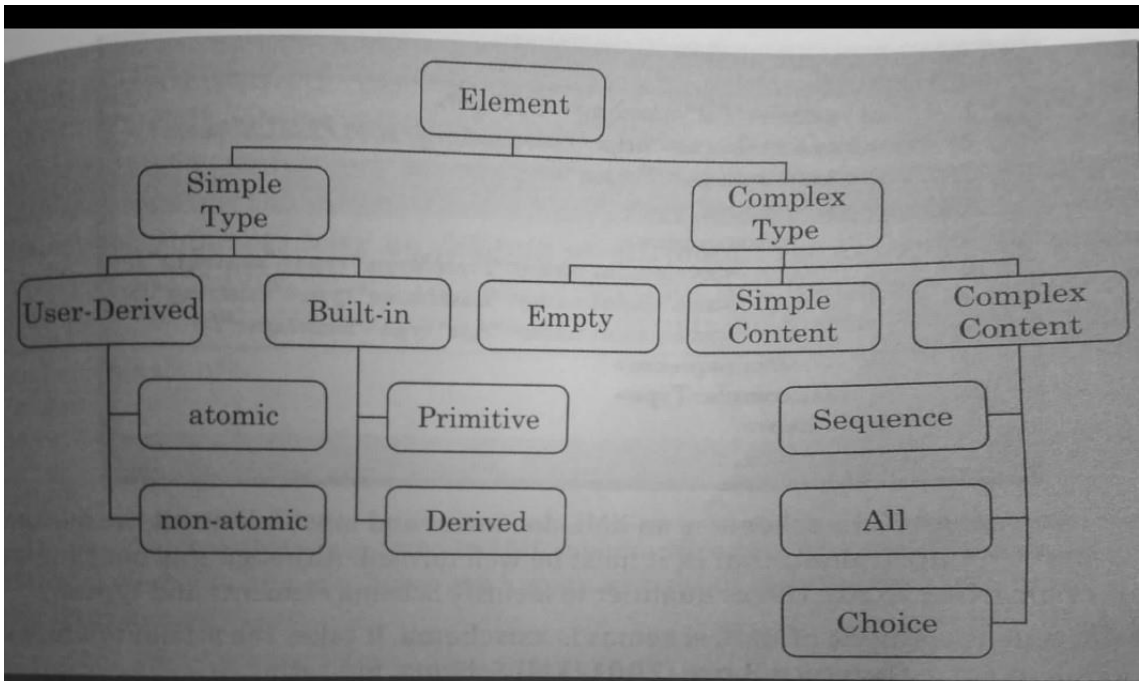
```
#box-1 {
width: 150px;
}

#box-2 {
width: 150px;
}

#box-3 {
width: 150px;
}
```

3.What are the different schema elements in XML?

An XML schema describes the structure of an XML instance document by defining what each element must or may contain. An element is limited by its type. The element can be of simple type or complex type.



- **The following is a high level overview of schema types:**

1.Elements can be simple or complex type.

2.Simple type elements can only contain text. They cannot have child element or attributes.

3.All the built-in types are simple types.

4.Schema developers can derive simple types by restricting another simple type.

5.Simple types can be atomic or non-atomic.

6.Complex-type elements can contain child elements and attributes as well as text.

7.By default, complex-type elements have complex content, meaning that they have child elements.

8.Complex-type elements can be limited to having simple content, meaning they only contain text. They are different from simple type elements in that they have attributes.

9. Complex types can be limited to having no content, meaning they are empty, but they have attributes.

10.Complex types may have mixed content-a combination of text and child elements.

[May/June 2016]

1.Write a note on XML name spaces.

- In XML, a namespace is used to prevent any conflicts with element names. Because XML allows to create our own element names, there's always the possibility of naming an element exactly the same as one in another XML document, this is possible when they never use both documents together.
- We can create a name conflict by creating a namespace for the XML document. To use XML namespace, elements are given qualified names. These qualified names consists of two parts:

  1.The local part is the same as the names we have been given elements.

  2.The namespace prefix specifies to which namespace this name belongs.

  Ex: <bk:books xmlns:bk="http://www.mysite.com"/>

Where, xmlns stands for XML Namsespace.

bk is the namespace prefix.

http:// www.mysite.com is the URL of the namespace.

Example:

<?xml version="1.0"encoding="UTF-8"?>

<!DOCTYPE books SYSTEM "bookrules dtd">

<bk:books xmlns:bk="http://www.musite.com">

<bk:book type="programming">

<bk:title>Web programming</bk:title>

<bk:author>Srikanth.S</bk:author>

</bk:book>

</bk:books>

[May/June 2017]

1.What is DTD? Explain internal DTD and external DTD.

A DTD is a Document Type Definition. A DTD defines the structure and the legal elements and attributes of an XML document.

1.Internal DTD: An internal DTD is the DTD is defined between the square brackets within the XML documents.

RULES:

1. The document type declaration must be written in between the XML declaration and the root element.
2. Keyword DOCTYPE must be followed by the root element.
3. Keyword DOCTYPE must be in upper case.

Example:

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tom</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

2.External DTD: An external DTD is nothing different from internal DTD except that defining DTD in an external file. It can be use with more than one XML document.

Example:

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tom</to>
  <from>Jani</from>
  <heading>Reminder</heading>
```

```
  <body>Don't forget me this weekend!</body>
</note>
```

[May/June 2018]

1.Describe XML processors.

XML parsers or processors support for many tasks, including managing configuration files, communicating customer data and archiving data. Given such diversity, it is important that applications are able to property interpret and manipulate XML files. An XML parser accepts an XML documents as its input or processes it to interpret the data that it contains. The parser may even manipulate the data and re-create yet another XML documents for further use by another applications.

There are two major techniques available for XML parsing:

1.Simple API for XML (SAX): It is an event based parsing. The parser generates an application event whenever it encounters an element or data in the document being parsed and it is simple API for processing XML documents.

2.Document Object Model (DOM): In this model, the parser builds an in-memory structure for the entire document that is parsed. We can use the DOM API to navigate the tree, modify the document contents and store into another XML files.

[May/June 2019]

MENTIONED ABOVE QUESTIONS ARE REPEATED