

Name : _____

Roll No : _____

6th SEMESTER B.TECH DEGREE MODEL EXAMINATION, MARCH 2018
CS 302 DESIGN & ANALYSIS OF ALGORITHMS

Duration : 3 Hrs

Max. Marks : 100

PART A
(Answer all Questions)

1. State Master theorem for solving recurrence relations. (3)
2. Prove that $\log n$ grows faster than $\sqrt{\log n}$. (3)
3. Write the properties of Red Black Tree. (3)
4. Explain disjoint set operations (3)

PART B
(Answer any two Questions)

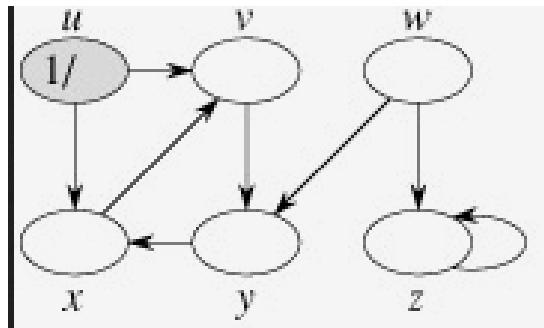
5. Build a B-Tree of degree 2 by inserting the following elements 5, 3, 21, 9, 1, 13, 2, 7, 10, 12, 4, 8 in the given order starting from an empty B-Tree. Show the B-Tree that result from the successive deletion of the keys 2, 21, 10, 3, 4 in that order starting from the constructed B-Tree (9)
6. (a) Use recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 3T(\frac{n}{2}) + n$ (6)
(b) Using Master method solve $T(n) = 2T(\frac{n}{2}) + n^3$ (3)
7. Explain the various asymptotic notations used to represent the rate of growth of running time of algorithms? (9)

PART C
(Answer all Questions)

8. Compare divide-and-conquer and dynamic programming algorithm design techniques (3)
9. What is meant by topological sorting? Where is it used? (3)
10. Write optimality principle. (3)
11. Draw the tree calls of sorting an array of 10 elements using merge sort (3)

PART D
(Answer any two Questions)

12. Write the single source shortest path algorithm. Also analyse the performance of the algorithm (9)
13. Find the optimal parenthesization of a matrix chain product whose sequence of dimension is $\langle 5, 10, 3, 12, 5, 50, 6 \rangle$ (9)
14. Perform a DFS traversal on the following graph and mark the black edge, forward edge, and cross edge



PART E
(Answer any four Questions)

15. Give the algorithm for Knapsack using Greedy strategy. Also find an optimal solution to the Knapsack instance $n=7$, $w=15$,
 $(P_1, P_2, \dots, P_7) = (10, 5, 15, 7, 6, 18, 3)$ and
 $(W_1, W_2, \dots, W_7) = (2, 3, 5, 7, 1, 4, 1)$ (10)
16. Write the Kruskal's algorithm for finding the minimum cost spanning trees? Derive the time complexity of the same algorithm (10)
17. Explain the following complexity classes with examples (10)
 - (a) P
 - (b) NP
 - (c) NP-complete
 - (d) NP-hard
18. What is backtracking? Discuss how backtracking can be used to solve the n'queens problem (10)
19. Prove that Travelling Salesman Problem is NP-complete (10)
20. Explain how 0/1 knapsack problem can be solved using backtracking

6th SEMESTER B.TECH DEGREE MODEL EXAMINATION, MARCH 2018
CS 302 DESIGN & ANALYSIS OF ALGORITHMS
ANSWER KEY

Duration : 3 Hrs

Max. Marks : 100

PART A
(Answer all Questions)

1. State Master theorem for solving recurrence relations.

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n),$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows.

- (a) If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \theta(n^{\log_b a})$
- (b) If $f(n) = \theta(n^{\log_b a})$, then $T(n) = \theta(n^{\log_b a} \cdot \lg n)$
- (c) If $f(n) = \Omega(n^{\log_b a + \epsilon})$, for some constant $\epsilon > 0$, and if $af(\frac{n}{b}) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n , then $T(n) = \theta(f(n))$.

2. Prove that $\log n$ grows faster than $\sqrt{\log n}$

$$f(n) = \log n, g(n) = \sqrt{\log n}$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{\log n}{\sqrt{\log n}} = \lim_{n \rightarrow \infty} \frac{\log n}{\log n^{\frac{1}{2}}} = \lim_{n \rightarrow \infty} \log n^{\frac{1}{2}} = \infty$$

So $f(n)$ grows faster than $g(n)$. That is $\log n$ grows faster than $\sqrt{\log n}$

3. Write the properties of Red Black Tree.

A binary search tree is a red-black tree if it satisfies the following red-black properties:

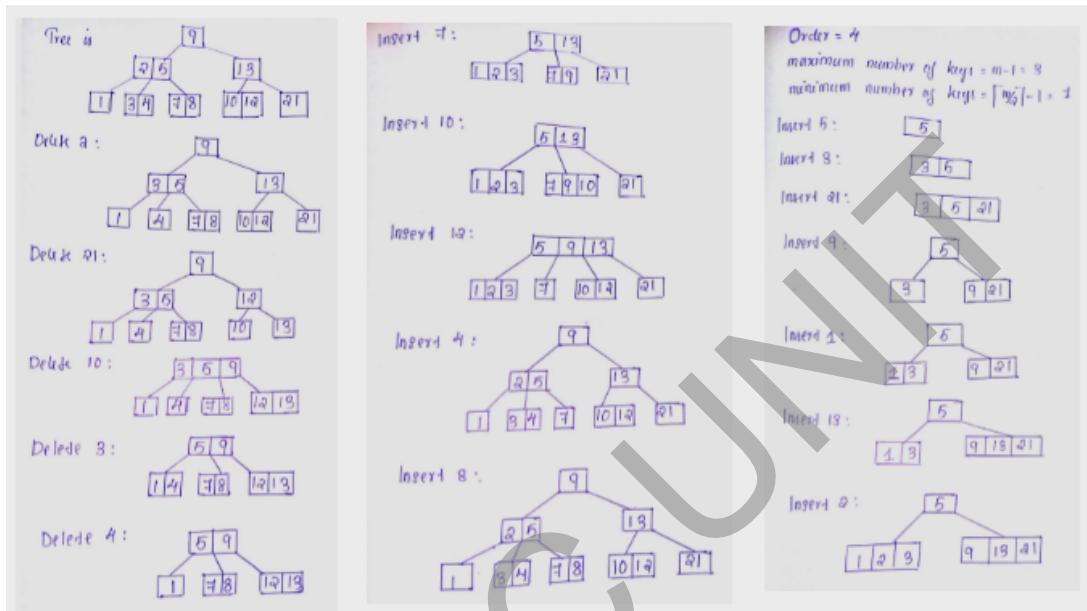
- (a) Every node is either red or black.
- (b) The root is black.
- (c) Every leaf (NIL) is black.
- (d) If a node is red, then both its children are black.
- (e) For each node, all paths from the node to descendant leaves contain the same number of black nodes.

4. Explain disjoint set operations.

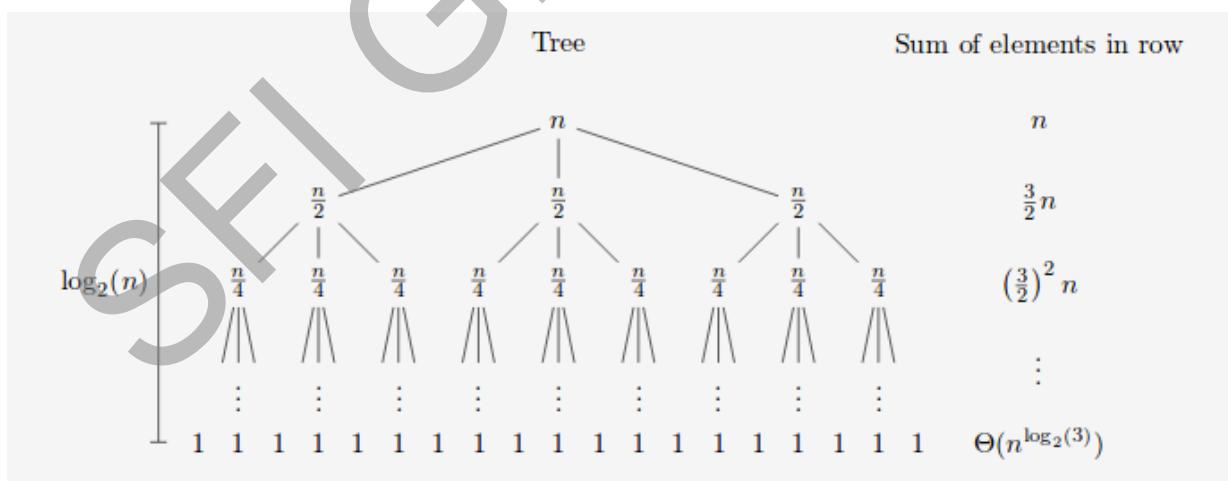
- (a) MAKE-SET(x) :- creates a new set whose only member (and thus representative) is x . Since the sets are disjoint, then x is not already in some other set.
- (b) UNION(x, y) :- unites the dynamic sets that contain x and y , say S_x and S_y , into a new set that is the union of these two sets. The two sets are assumed to be disjoint prior to the operation. The representative of the resulting set is any member of $S_x \cup S_y$. In some case specifically choose the representative of either S_x or S_y as the new representative. Since the sets in the collection to be disjoint, "destroy" sets S_x and S_y , by removing them from the collection .
- (c) FIND-SET(x) returns a pointer to the representative of the (unique) set containing x .

PART B
(Answer any two Questions)

5. Build a B-Tree of degree 2 by inserting the following elements 5, 3, 21, 9, 1, 13, 2, 7, 10, 12, 4, 8 in the given order starting from an empty B-Tree. Show the B-Tree that result from the successive deletion of the keys 2, 21, 10, 3, 4 in that order starting from the constructed B-Tree



6. (a) Use recursion tree to determine a good asymptotic upper bound on the recurrence (6)
 $T(n) = 3T\left(\frac{n}{2}\right) + n$
 Recursion tree is shown below. Level i has 3^i nodes.



$$= 2n \cdot \frac{3^{\log_2 3-1}}{2^{\log_2 3-1}} + n^{\log_2 3}$$

$$= O(n^{\log_2 3})$$

(b) Using Master method solve $T(n) = 8T(\frac{n}{2}) + n^3$

$$a = 8, b = 2, f(n) = n^3$$

$$n^{\log_b a} = n^{\log_2 8} = n^3$$

$f(n)$ and $n^{\log_b a}$ grows in same rate.

$$\text{So case 3, } f(n) = \theta(n^{\log_b a})$$

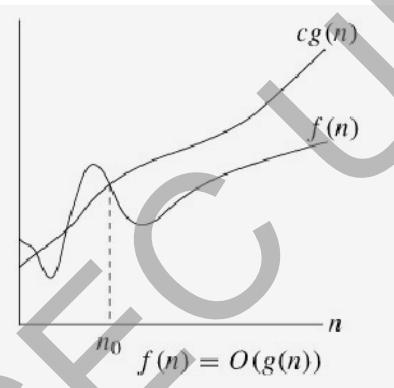
$$n^3 = \theta(n^3).$$

So solution is $T(n) = \theta(n^3 \cdot \log n)$

7. Explain the various asymptotic notations used to represent the rate of growth of running time of algorithms?

O-Notation

$O(g(n)) = \{f(n) : \text{there exists 2 positive constants } c \text{ and } n_0 \text{ such that } f(n) \leq cg(n) \text{ for all } n \geq n_0\}$.



This gives asymptotic an lower bound on a function, to within a constant factor. This asymptotic upper bound provided by O-notation may or may not be asymptotically tight.

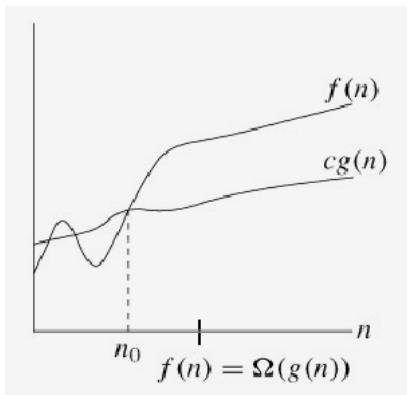
A function $f \in O(g(n))$ if

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c < \infty$$

including the case in which the limit is zero. The set of function in g(n) grows faster than the set of functions in f(n).

Ω -Notation

$\Omega(g(n)) = \{f(n) : \text{there exists 2 positive constants } c \text{ and } n_0 \text{ such that } 0f(n) \geq cg(n) \text{ for all } n \geq n_0\}$.



This gives asymptotic an lower bound

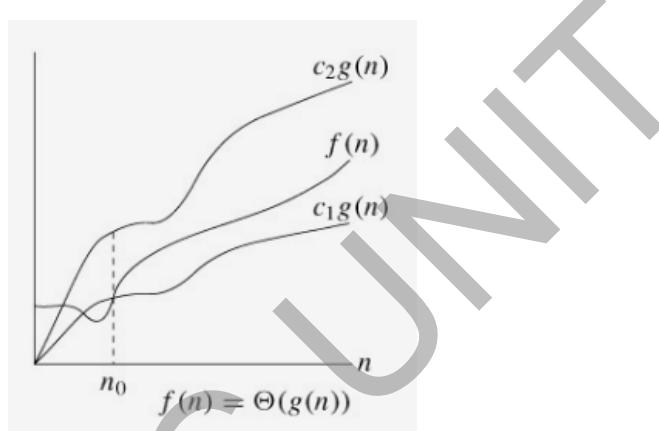
A function $f \in \Omega(g(n))$ if

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c > 0$$

including the case in which the limit is ∞ . The set of function in $f(n)$ grows faster than the set of functions in $g(n)$.

θ -Notation

$\theta(g(n)) = \{f(n) : \text{there exists positive constants } c_1, c_2 \text{ and } n_0 \text{ such that } c_1g(n) \leq f(n) \leq c_2g(n) \text{ for all } n \geq n_0\}$.



For all values of n to the right of n_0 , the value of $f(n)$ lies at or above $c_1g(n)$ and at or below $c_2g(n)$. In other words, for all $n \geq n_0$, the function $f(n)$ is equal to $g(n)$ to within a constant factor. $g(n)$ is an asymptotically tight bound for $f(n)$.

A function $f \in \theta(g(n))$ if

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c$$

The set of function in $f(n)$ grows in same rate as $g(n)$.

o -Notation

o -notation is used to denote an upper bound that is not asymptotically tight.

$o(g(n)) = f(n) : \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } f(n) < cg(n) \text{ for all } n > n_0$. In the o -notation, the function $f(n)$ becomes insignificant relative to $g(n)$ as n approaches infinity. $f \in o(g(n))$ if

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

ω -Notation

ω -notation is used to denote a lower bound that is not asymptotically tight. $\omega(g(n)) = f(n) : \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } f(n) > cg(n) \text{ for all } n \geq n_0$. $f \in o(g(n))$ if

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$$

PART C (Answer all Questions)

8. Compare divide-and-conquer and dynamic programming algorithm design techniques

Divide-and-conquer	Dynamic programming
Top-down	Bottom up
As and when a subproblem is occurred the divide-and-conquer strategy applied on it even though the subproblem might have been solved earlier. This may result in repeated computation of the same subproblem	Repeated computation of the solutions of the same subproblem is avoided. rather when a subproblem is solved its solution is entered in a dictionary. The solution is reused whenever required.
A problem under consideration should not satisfy the principle of optimality	A problem under consideration should satisfy the principle of optimality(Principle of optimal substructure)
Divide-and-conquer call themselves recursively one or more times to deal with closely related sub problems.	There is no recursion
The sub-problems are independent of each other	The sub-problems are not independent of each other (Solution of one sub-problem may be required to solve another sub-problem).

9. What is meant by topological sorting? Where is it used?

- A topological sort of a dag $G = (V, E)$ is a linear ordering of all its vertices such that if G contains an edge (u, v) , then u appears before v in the ordering. If the graph is not acyclic, then no linear ordering is possible. A topological sort of a graph can be viewed as an ordering of its vertices along a horizontal line so that all directed edges go from left to right.
- Topological sorting can be used to schedule tasks under precedence constraints.
A real time example is building a house

10. Write optimality principle.

The principle of optimality states that an optimal sequence of decisions has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision.

11. Draw the tree calls of sorting an array of 10 elements using merge sort.

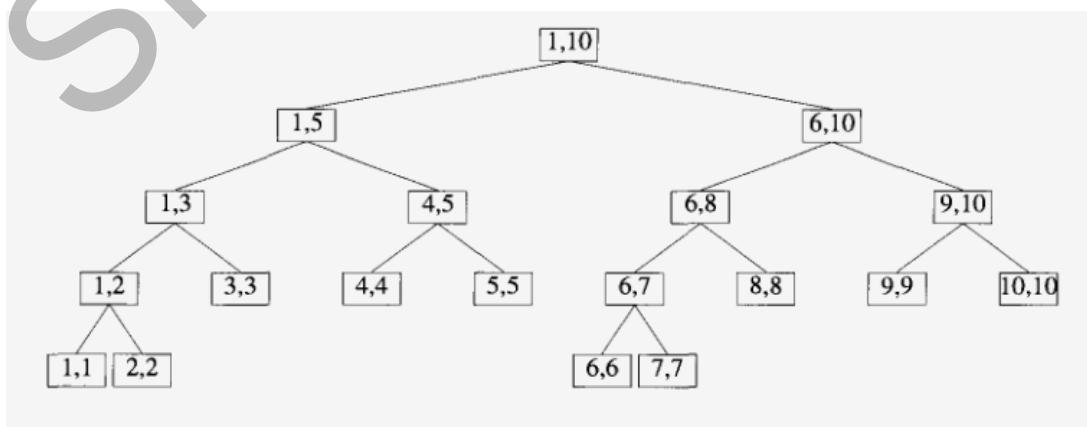


Figure 1: Tree of calls of MergeSort(1, 10)

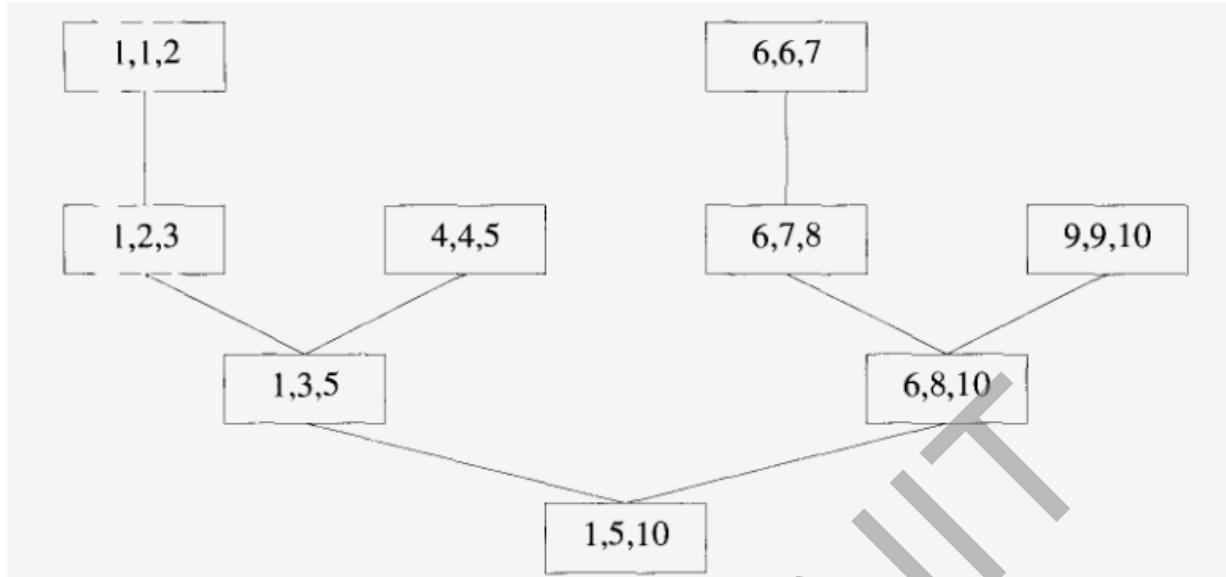


Figure 2: Tree of calls of Merge

PART D
(Answer any two Questions)

12. Write the Dijkstras single source shortest path algorithm. Also analyse the performance of the algorithm.

Dijkstra's algorithm solves the single-source shortest-paths problem on a weighted, directed graph $G = (V, E)$ for the case in which all edge weights are nonnegative. Here assume that $w(u, v) \geq 0$ for each edge $(u, v) \in E$. Dijkstra's algorithm maintains a set S of vertices whose final shortest-path weights from the source s have already been determined. The algorithm repeatedly selects the vertex $u \in V - S$ with the minimum shortest-path estimate, adds u to S , and relaxes all edges leaving u . In the implementation a min-priority queue Q of vertices is used, keyed by their d values. It uses a greedy strategy. $\text{DIJKSTRA}(G, w, s)$

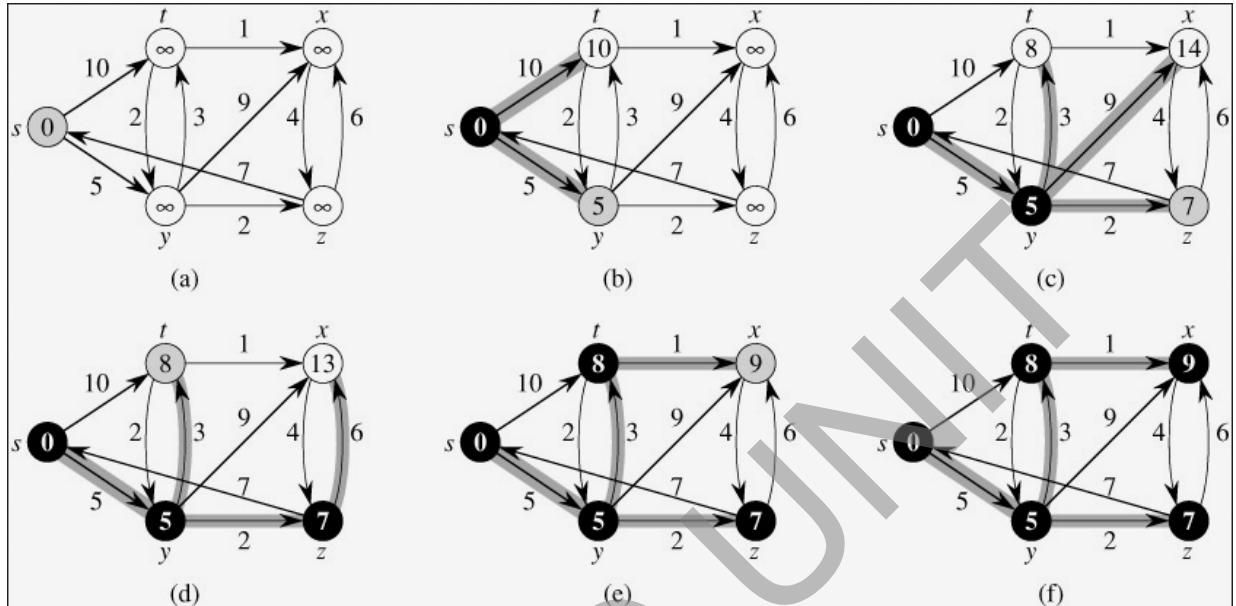
- (a) INITIALIZE-SINGLE-SOURCE(G, s)
- (b) $S = \emptyset$
- (c) $Q = G.V$
- (d) while $Q \neq \emptyset$
- (e) $u = \text{EXTRACT-MIN}(Q)$
- (f) $S = S \cup u$
- (g) for each vertex $v \in G.\text{Adj}[u]$
- (h) $\text{RELAX}(u, v, w)$

INITIALIZE-SINGLE-SOURCE(G, s)

- (a) for each vertex $v \in G.V$
- (b) $v.d = \infty$
- (c) $v.\pi = \text{NIL}$
- (d) $s.d = 0$

RELAX(u, v, w)

- (a) if $v.d > u.d + w(u, v)$
- (b) $v.d = u.d + w(u, v)$
- (c) $v.\pi = u$



Minimum Priority Queue					Vetex Processed
s	t	x	y	z	
0	∞	∞	∞	∞	{s}
	∞	∞	5	∞	{s,y}
	8	14		7	{s,y,z}
	8	13			{s,y,z,t}
		9			{s,y,z,t,x}

Figure 3: Processing of each vertex

Analysis

It maintains the min-priority queue Q by calling three priority-queue operations:

- (a) INSERT (implicit in line 3)
- (b) EXTRACT-MIN (line 5)
- (c) DECREASE-KEY (implicit in RELAX, which is called in line 8).

INSERT is invoked once per vertex, as is EXTRACT-MIN. Because each vertex $v \in V$ is added to set S exactly once, each edge in the adjacency list $\text{Adj}[v]$ is examined in the for loop of lines 7-8 exactly once during the course of the algorithm. Since the total number of edges in all the adjacency lists is

$|E|$, there are a total of $|E|$ iterations of this for loop, and thus a total of at most $|E|$ DECREASE-KEY operations.

The running time of Dijkstra's algorithm depends on how the min-priority queue is implemented. Implement the min-priority queue with a binary min-heap. Each EXTRACT-MIN operation then takes time $O(\lg V)$. There are $|V|$ such operations. The time to build the binary min-heap is $O(V)$. Each DECREASE-KEY operation takes time $O(\lg V)$, and there are still at most $|E|$ such operations. The total running time is therefore $O((V + E) \lg V)$, which is $O(E \lg V)$ if all vertices are reachable from the source.

13. Find the optimal parenthesization of a matrix chain product whose sequence of dimension is

$$\langle 5, 10, 3, 12, 5, 50, 6 \rangle \quad (9)$$

Matrix order set $P=5,10,3,12,5,50$

Matrix set $A=A_1, A_2, A_3, A_4, A_5, A_6$

From the algorithm, we have, for all x , $m[x,x] = 0$.

$$m[1,2] = m[1,1] + m[2,2] + p0 * p1 * p2 = 0 + 150 = 150$$

$$m[2,3] = m[2,2] + m[3,3] + p1 * p2 * p3 = 0 + 360 = 360$$

$$m[3,4] = m[3,3] + m[4,4] + p2 * p3 * p4 = 0 + 180 = 180$$

$$m[4,5] = m[4,4] + m[5,5] + p3 * p4 * p5 = 0 + 3000 = 3000$$

$$m[5,6] = m[5,5] + m[6,6] + p4 * p5 * p6 = 0 + 1500 = 1500$$

$$m[1,3] = \min \begin{cases} m[1,1] + m[2,3] + p0 * p1 * p3 = 960 \\ m[1,2] + m[3,3] + p0 * p2 * p3 = 330 \end{cases}$$

$$m[2,4] = \min \begin{cases} m[2,2] + m[3,4] + p1 * p2 * p4 = 330 \\ m[2,3] + m[4,4] + p1 * p3 * p4 = 960 \end{cases}$$

$$m[3,5] = \min \begin{cases} m[3,3] + m[4,5] + p2 * p3 * p5 = 4800 \\ m[3,4] + m[5,5] + p2 * p4 * p5 = 930 \end{cases}$$

$$m[4,6] = \min \begin{cases} m[4,4] + m[5,6] + p3 * p4 * p6 = 1860 \\ m[4,5] + m[6,6] + p3 * p5 * p6 = 6600 \end{cases}$$

$$m[1,4] = \min \begin{cases} m[1,1] + m[2,4] + p0 * p1 * p4 = 580 \\ m[1,2] + m[3,4] + p0 * p2 * p4 = 405 \\ m[1,3] + m[4,4] + p0 * p3 * p4 = 630 \end{cases}$$

$$m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + p1 * p2 * p5 = 2430 \\ m[2,3] + m[4,5] + p1 * p3 * p5 = 9360 \\ m[2,4] + m[5,5] + p1 * p4 * p5 = 2830 \end{cases}$$

$$m[3,6] = \min \begin{cases} m[3,3] + m[4,6] + p2 * p3 * p6 = 2076 \\ m[3,4] + m[5,6] + p2 * p4 * p6 = 1770 \\ m[3,5] + m[6,6] + p2 * p5 * p6 = 1830 \end{cases}$$

$$m[1,5] = \min \begin{cases} m[1,1] + m[2,5] + p0 * p1 * p5 = 4930 \\ m[1,2] + m[3,5] + p0 * p2 * p5 = 1830 \\ m[1,3] + m[4,5] + p0 * p3 * p5 = 6330 \\ m[1,4] + m[5,5] + p0 * p4 * p5 = 1655 \end{cases}$$

$$m[2, 6] = \min \text{ of } \begin{cases} m[2, 2] + m[3, 6] + p_1 * p_2 * p_6 = 1950 \\ m[2, 3] + m[4, 6] + p_1 * p_3 * p_6 = 2920 \\ m[2, 4] + m[5, 6] + p_1 * p_4 * p_6 = 2130 \\ m[2, 5] + m[6, 6] + p_1 * p_5 * p_6 = 5430 \end{cases}$$

$$m[1, 6] = \min \text{ of } \begin{cases} m[1, 1] + m[2, 6] + p_0 * p_1 * p_6 = 2250 \\ m[1, 2] + m[3, 6] + p_0 * p_2 * p_6 = 2010 \\ m[1, 3] + m[4, 6] + p_0 * p_3 * p_6 = 2530 \\ m[1, 4] + m[5, 6] + p_0 * p_4 * p_6 = 2055 \\ m[1, 5] + m[6, 6] + p_0 * p_5 * p_6 = 3155 \end{cases}$$

m table

0	150	330	405	1655	2010
	0	360	330	2430	1950
		0	180	930	1770
			0	3000	1840
				0	1500
					0

s table s[1..5][2..6]

	2	3	4	5	6
1	1	2	2	4	2
2		2	2	2	2
3			3	4	4
4				4	4
5					5

In our problem, n=6. The array s[1..5, 2..6] has been computed above.

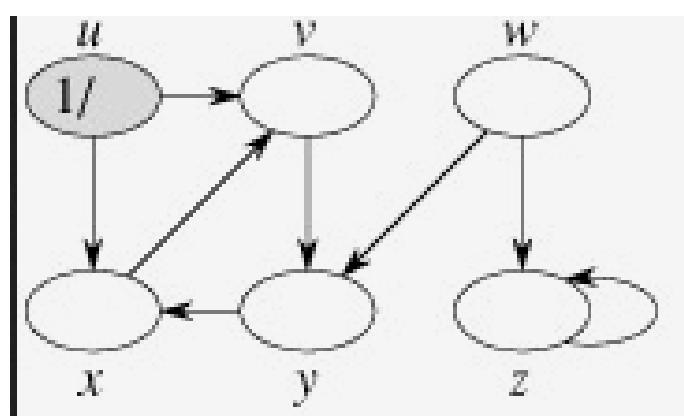
$$s[1,6] = 2 (A_1 A_2)(A_3 A_4 A_5 A_6)$$

$$s[1,2] = 1 (A_1 A_2)$$

$$s[3,6] = 4 (A_3 A_4)(A_5 A_6)$$

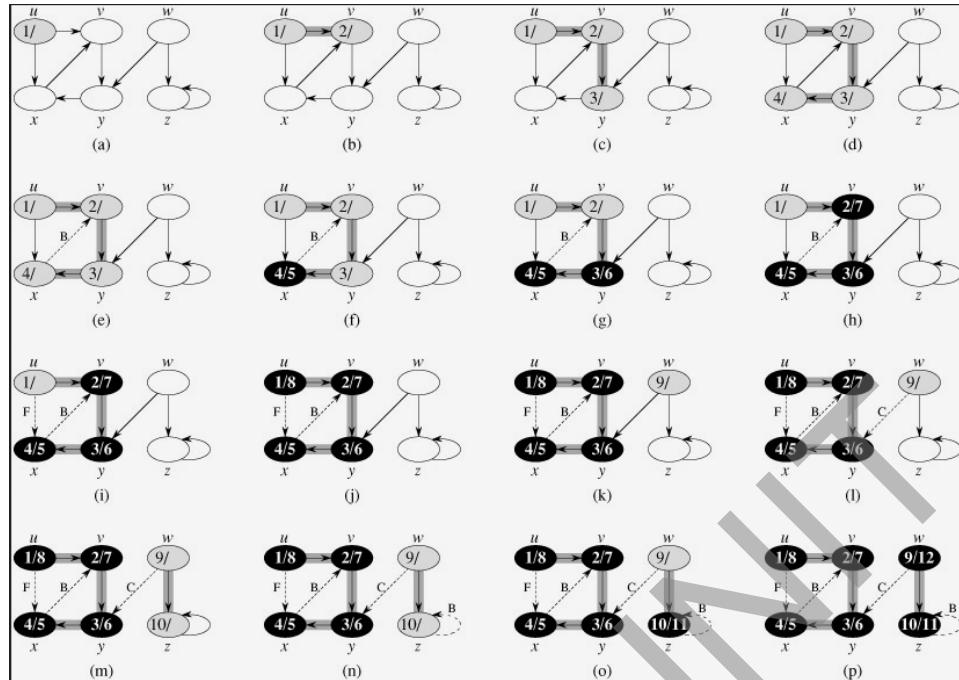
So the final multiplication sequence is $(A_1 A_2)(A_3 A_4)(A_5 A_6)$

14. Perform a DFS traversal on the following graph and mark the black edge, forward edge, and cross edge



PART E

(Answer any four Questions)



15. Give the algorithm for Knapsack using Greedy strategy. Also find an optimal solution to the Knapsack instance $n=7$, $w=15$,

$$(P_1, P_2, \dots, P_7) = (10, 5, 15, 7, 6, 18, 3)$$

$$(W_1, W_2, \dots, W_7) = (2, 3, 5, 7, 1, 4, 1)$$

Algorithm: Greedy-Fractional-Knapsack ($W[1..n]$, $p[1..n]$, w , $x[1..n]$)

- (a) for $i := 1$ to n do
- (b) $x[i] := 0$;
- (c) weight := 0 ;
- (d) for $i := 1$ to n
- (e) if weight + $W[i] \leq w$ then
- (f) $x[i] := 1$;
- (g) weight := weight + $W[i]$;
- (h) else
- (i) $x[i] := (w - \text{weight}) / W[i]$;
- (j) weight := w ;
- (k) break;
- (l) return x

Let (a,b,c,d,e,f,g) represent the items with profit (10,5,15,7,6,18,3) and weight (2, 3, 5, 7, 1, 4,1)

Optimal solution that gives maximum profit.

item	P	W	P/W
a	10	2	5
b	5	3	1.67
c	15	5	3
d	7	7	1
e	6	1	6
f	18	4	4.5
g	3	1	3

Step 1:Find P/W ratio

item	P	W	P/W
e	6	1	6
a	10	2	5
f	18	4	4.5
g	3	1	3
c	15	5	3
b	5	3	1.67
d	7	7	1

Step 2: Arrange this profit/weight ratio in non-increasing order as n values

w=15 x[a,b,c,d,e,f,g,h]=0 weight=0 Take e, weight=0+1=1, x[e]=1, profit=6

Take a, weight=1+2=3, x[a]=1, profit=6+10=16

Take f, weight=3+4=7, x[f]=1, profit=16+18=34

Take g, weight=7+1=8, x[g]=1, profit=34+3=37

Take c, weight=8+5=13, x[c]=1, profit=37+15=52

Take b, w[b] > weight so w=15, x[b]=(5-13)/3=2/3, profit=52+5*2/3=55.33

Profit = 55.33

x={1,2/3,1,0,1,1,1}

16. Write the Kruskal's algorithm for finding the minimum cost spanning trees? Derive the time complexity of the same algorithm

Kruskal's algorithm is a greedy algorithm, because at each step it adds to the forest an edge of least possible weight. It uses a disjoint-set data structure to maintain several disjoint sets of elements. Each set contains the vertices in a tree of the current forest. The operation FIND-SET(u) returns a representative element from the set that contains u . Thus, we can determine whether two vertices u and v belong to the same tree by testing whether FIND-SET(u) equals FIND-SET(v). The combining of trees is accomplished by the UNION procedure.

MST-KRUSKAL(G, w)

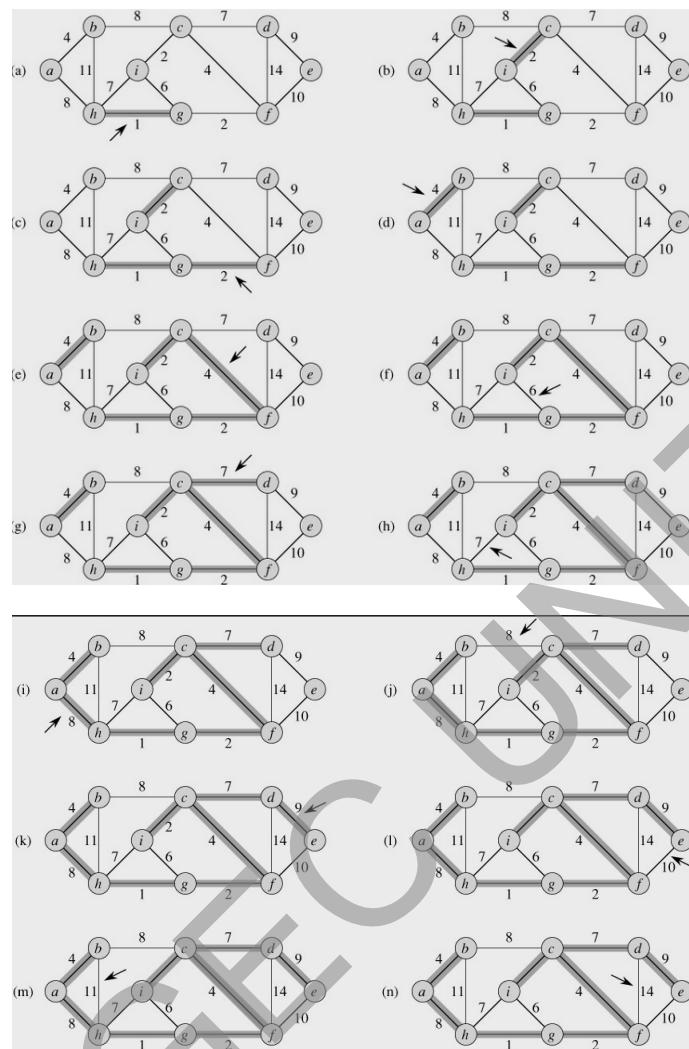
- (a) $A = \emptyset$
- (b) for each vertex $v \in G. V$
- (c) MAKE-SET(v)
- (d) sort the edges of E into nondecreasing order by weight w
- (e) for each edge $(u, v) \in E$, taken in nondecreasing order by weight
- (f) if FIND-SET(u) \neq FIND-SET(v)
- (g) $A = A \cup \{(u, v)\}$
- (h) UNION(u, v)
- (i) return A

Kruskal's algorithm works as shown in Figure

Analysis

The running time of Kruskal's algorithm for a graph $G = (V, E)$ depends on the implementation of the disjoint-set data structure.

- Initializing the set A in line 1 takes $O(1)$ time.
- Line 2-3 takes $|V|$ MAKE-SET operations.
- The time to sort the edges in line 4 is $O(E \lg E)$.
- The for loop of lines 5-8 performs $O(E)$ FIND-SET and



- UNION operations on the disjoint-set forest. FIND-SET operation takes $O(\log V)$ time. So line 5-8 takes $O(E \log V)$ time
- The total running time of Kruskal's algorithm is $O(E \lg V)$.

17. Explain the following complexity classes with examples

- (a) Class P :- Class of all problems which can be solved by polynomial time deterministic algorithms. That is there are problems that can be solved in time $O(n^k)$ for some constant k, where n is the size of the input to the problem.
Eg: Sorting, searching
- (b) Class NP :- NP is the class of decision problems for which there is polynomially bounded non-deterministic algorithm. NP comes from “Non deterministic polynomially bounded”. NP problems means problems that are verifiable in polynomial time. It means that given a “certificate” of a solution then we could verify that the certificate is correct in polynomial time in the size of the input to the problem.
Eg:- Hamiltonian cycle problem
Hamiltonian cycle in an undirected graph is a simple cycle that passes through every vertex exactly once.
- (c) Class NP-hard :- Polynomial time reduction & reducibility :- Let T be a function from the input set for a decision problems P into the input set for decision problem Q. T is a polynomial reduction (polynomial transformation) from P to Q if all of the following hold:

- i. T can be computed in polynomially bounded time.
- ii. For every string x, if x is yes input for P, $T(x)$ is yes input for Q.
- iii. For every string x, if x is a no input for P, then $T(x)$ is a no input for Q.

A problem P is polynomially reducible (polynomially transformable) to Q if there exists a polynomial transformation from P to Q. P is reducible to Q

$$P \leq_p Q$$

A problem Q is said to be NP hard iff every problem P in NP is polynomial time reducible to Q.

Eg :- Circuit satisfiability problem, Travelling salesman problem, Graph coloring problem

- (d) Class NP-complete :- A problem Q is said to be NP complete iff

- i. it is in NP
- ii. It is in NP hard

Eg :- Travelling salesman problem

18. What is backtracking? Discuss how backtracking can be used to solve the n'queens problem

Backtracking is a technique used to solve problems with a large search space, by systematically trying and eliminating possibilities. There is a binary choice “yes” or “no”. Whenever the backtracking have the choice “no” that means the algorithm has encountered a dead end and it backtracks one step and tries a different path for choice “yes”. Backtracking resembles a depth

first search tree in a directed graph, where graph is either a tree or atleast it does not have any cycles
N queens problem

N queens problem is to place n queens in such a manner on an $n \times n$ chessboard that no two queens attack each other by being in the same row, column or diagonal.

when n=1 problem has trivial solution and no solution exists for n=2and n=3.

So consider 4 queens problem and generalise n queens problem

4-queens problem

Given a 4×4 chessboard and numbering the rows and columns of chessboard 1 through 4. Place 4 queens q1, q2, q3 and q4 on a chessboard such that no two queens attack each other.

Place q1 in very acceptable position (1,1).

Next place q2 such that these queens do not attack each other. Place q2 in column (2,3).

But no position left for placing q3. So backtrack one step and place q2 in (2,4).

Then obtain the position for placing q3 which is (3,2). This position leads to dead end and no place is found where q3 can place safely. Backtrack till q1 and place it in (1,2) and then all other queens are placed safely by moving q2 to (2,4), q3 to (3,1) and q4 to (4,3). That is we get the solution as (2,4,1,3). This is one possible solution. Another solutions are also there.

	x		
			x
x			
		x	

SO in general 2 queens are placed at positions (i,j) and (k,l)

then they are on same diagonal only if

$(i-j)=(k-l)$ means $j-l = i-k$ or $i+j = k+l$ means $j-l=k-i$

The algorithm place(k,i) returns a boolean value that is true if the k^{th} queen can be placed in column i. It also tests both whether i is distinct from all previous values x1,x2,x3, ...,xk-1 and whether there is no other queen on the same diagonal

Algorithm place(k, i)

```

(a) for j:=1 to k-1
(b) {
(c)     if (x[j]=i) or abs(x[j])-i=abs(j-k) then
(d)         return false;
(e) }
(f) return true;
(g) }

```

Algorithm N-Queens(k,n)

```

(a) {
(b) for i=1 to n do
(c) {
(d)     if place(k,i) then
(e)     {
(f)         x[k] := i;
(g)         if (k=n) then
(h)             write (x[1..n]);
(i)         else
(j)             N-Queens(k+1,n)
(k)     }
(l) }

```

19. Prove that Travelling Salesman Problem is NP-complete.

- First show that TSP belongs to NP.

– Given an instance of the problem, we use as a certificate the sequence of n vertices in the tour. The verification algorithm checks that this sequence contains each vertex exactly once, sums up the edge costs, and checks whether the sum is at most k . This process can certainly be done in polynomial time.

- Second prove that TSP is NP-hard

– To prove that TSP is NP-hard, we show that HAM-CYCLE \leq_p TSP. Let $G = (V, E)$ be an instance of HAM-CYCLE. We construct an instance of TSP as follows. We form the complete graph $G' = (V, E')$, where $E' = \{i, j\} : i, j \in V$ and $i \neq j$, and we define the cost function c by

(Note that because G is undirected, it has no self-loops, and so $c(v, v) = 1$ for all vertices $v \in V$.) The instance of TSP is then $(G', c, 0)$, which is easily formed in polynomial time.

$$c(i, j) = \begin{cases} 0 & \text{if } (i, j) \in E \\ 1 & \text{if } (i, j) \notin E \end{cases}$$

We now show that graph G has a hamiltonian cycle if and only if graph G' has a tour of cost at most 0. Suppose that graph G has a hamiltonian cycle h . Each edge in h belongs to E and thus has cost 0 in G' . Thus, h is a tour in G' with cost 0. Conversely, suppose that graph G' has a tour h' of cost at most 0. Since the costs of the edges in E' are 0 and 1, the cost of tour h' is exactly 0 and each edge on the tour must have cost 0. Therefore, h' contains only edges in E . We conclude that h' is a hamiltonian cycle in graph G .

First show that TSP belongs to NP.

20. Explain how 0/1 knapsack problem can be solved using backtracking.

The idea of backtracking is to construct solutions one component at a time and evaluate such partially constructed solutions. This partially constructed solution can be developed further without violating the problem constraints. It is convenient to implement this kind of processing by constructing a tree of choices being made called the State Space Tree. Its root represents an initial state before the search for the solution begins. The nodes of the first level in the tree represent the choices for the first component of a solution and the nodes of a second level represent the choices for the second component and so on.

A node in the state space tree is promising if it corresponds to the partials constructed solution that may lead to the complete solution otherwise the nodes are called non-promising. Leaves of the tree represent either the non-promising dead end or complete solution found by the algorithm. In this problem pack the knapsack in such a manner that the total weight of items should not be greater than the knapsack weight W and it should yield a maximum value w_i and v_i indicate weight and value respectively. Here not allowed to take a fraction of item. For example Given three types of items with following weights and values $T = T_1, T_2, T_3$, $w = 2, 3, 4$, $v = 3, 4, 5$

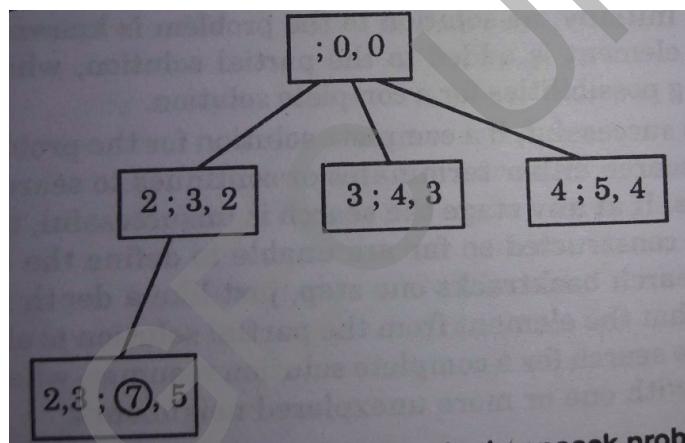


Figure 4: State space diagram

In every node left hand side of semicolon corresponds to the weight chosen and the first entry to the right hand side of the semicolon corresponds to the total value, and the second entry corresponds to total weight with respect to left side of the semicolon. Such a node structure corresponds to partial solution. Initially our partial solution is empty. At every move down to the children select the type of the item to be added in the knapsack.

Here weights are considered in non-increasing order

If we first visit the node $(2, 3; 7, 5)$ then next time we do not visit node $(3, 2; 7, 5)$. The first node visited is $(2; 3, 2)$ then the next is $(2, 3; 7, 5)$. When each node is visited the partial solution also extends. After visiting these two nodes the dead end comes as node $(2, 3; 7, 5)$ has no unvisited successors, since adding more items to this partial solution violates the knapsack capacity constraint. This partial solution produces the optimum solution so far, thus we memorise it. Next backtrack one step and find that new addition $(2, 4; 8, 6)$ will also violate the capacity. In the same manner we backtrack one step and proceed the search and it will continue until we get optimum solution

MODEL QUESTION PAPER

SIXTH SEMESTER B.TECH DEGREE EXAMINATION,MARCH 2018
COURSE CODE:CS302
COURSE NAME:DESIGN AND ANALYSIS OF ALGORITHM

Max Mark:100

Duration:3 hrs

PART A

(Answer all questions.Each question carries 4 Marks.)

1. State masters theorem
 2. Define recursion tree?
 3. Give a recurrence for merge sort?
 4. What is the maximum no of internal nodes in a red-black tree? What is the minimum?
- (4*4=12 marks)

PART B

(Answer any 2 questions.Each carries 9 Marks.)

5. a) Solve the following recurrence?
 - i) $T(n)=3T(n/2)+n$
- b) Write the recursion version of insertion sort, which can be expressed as follows: inorder to sort $A[1...n-1]$. We recursively sort $A[1...n-1]$ and then insert $A[n]$ into sorted array $A[1...n-1]$. write and solve recurrence.
6. a) Define red-black tree?
 - b) Write an algorithm for insertion in RB tree.
7. a) write an algorithm to search an element in a list using divide and conquer method.
 - b) Explain properties of B-tree

(9*2=18 marks)

PART C

(Answer all questions.Each carries 3 Marks.)

8. Write an algorithm for breadth first search? Explain
9. What you mean by topological sort?
10. What is strongly connected component?
11. What is the difference b/w dynamic programming and recursion?

(4*4=12marks.)

PART D

(Answer any 2 questions.Each carries 9 Marks.)

12. a) Apply DFS & BFS search to complete graph on 4 vertices. List the vertex in the order they visited.
- b) Given a directed graph $G=(V,E)$, a weighting function $w(e)$ for the edges of G and source vertex V_0 . Write an algorithm for determine the shortest path from V_0 to all remaining nodes.
- c) Perform a DFS traversal on the tree and mark the back,forward & cross edges.
13. Explain dynamic programming with example
14. What is 0-1 knapsack problem?

PART E
(Answer any 4 questions. Each carries 10 Marks)

- 15.Explain prim's Algorithim with example?
- 16.Explain Fractional knapsack algorithm
- 17.Explain travelling salesman problem with example
- 18.Explain an algorithm to solve the N-Queens problems
- 19.a)Difference between tractable and intractable problem
b)Explain NP-Hard and Np-Complete Classes

SFI GEC UNIT

Answer key

1.The master theorem can be employed to solve recursive equations of the form

$$T(n) = aT(n/b) + f(n)$$

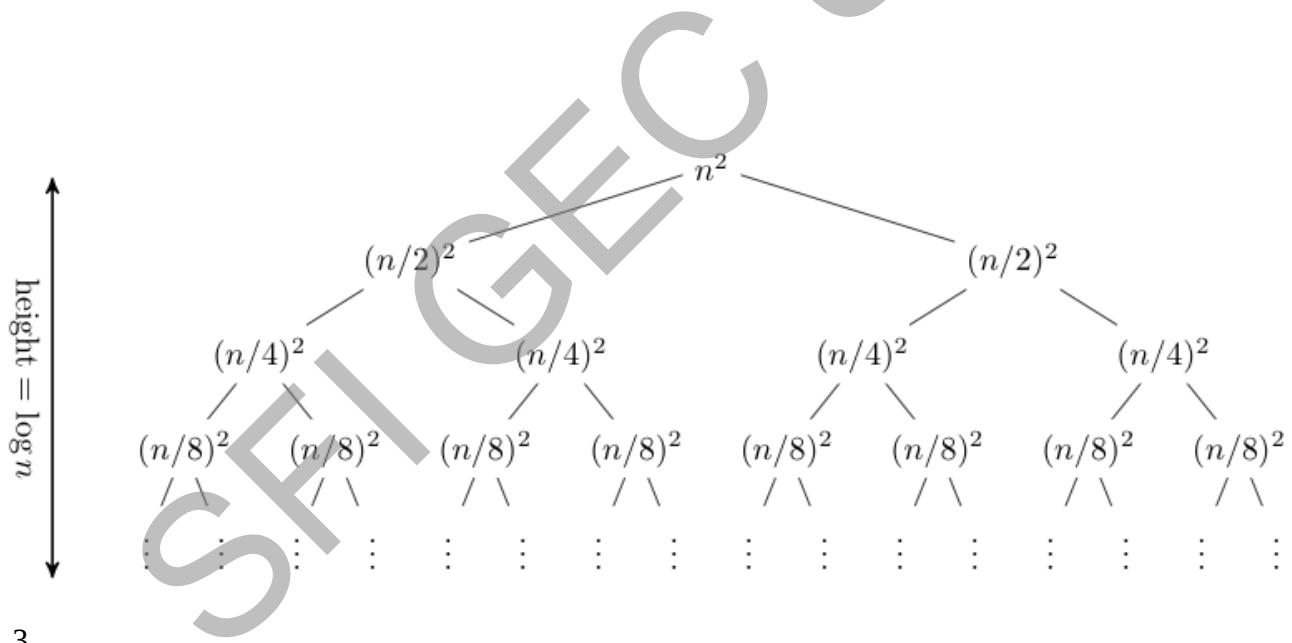
where $a \geq 1$, $b > 1$, and $f(n)$ is *asymptotically positive*. Intuitively for divide and conquer algorithms, this equation represents dividing the problem up into a subproblems of size n/b with a combine time of $f(n)$. For example, for merge sort $a = 2$, $b = 2$, and $f(n) = \Theta(n)$. Note that floors and ceilings for n/b do not affect the asymptotic behavior or the results derived using the theorem.

2.A *recursion tree* is useful for visualizing what happens when a recurrence is iterated. It diagrams the tree of recursive calls and the amount of work done at each call.

For instance, consider the recurrence

$$T(n) = 2T(n/2) + n^2.$$

The recursion tree for this recurrence has the following form:



3.

Divide-and-conquer solution for sorting an array gives an algorithm known as mergesort:

Mergesort:

{

Divide: Divide an array of n elements into two arrays of $n=2$ elements each.

{

Conquer: Sort the two arrays recursively.

{

Combine: Merge the two sorted arrays.

Assume we have procedure Merge($= A$; p ; q ; r)which merges sorted $A[p..q]$ with sorted $A[q+1....r]$
We can sort $A[p...r]$ as follows (initially $p=0$ and $r=n-1$):

Merge Sort(A,p,r)

If

$p < r$

then

$q=b$

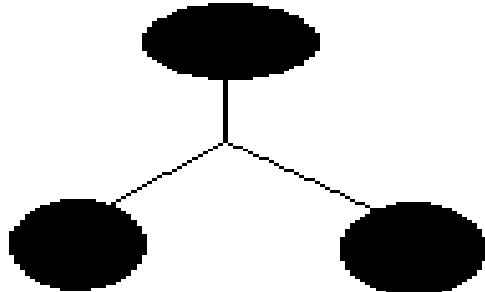
$(p+r)=2c$

MergeSort(A,p,q)

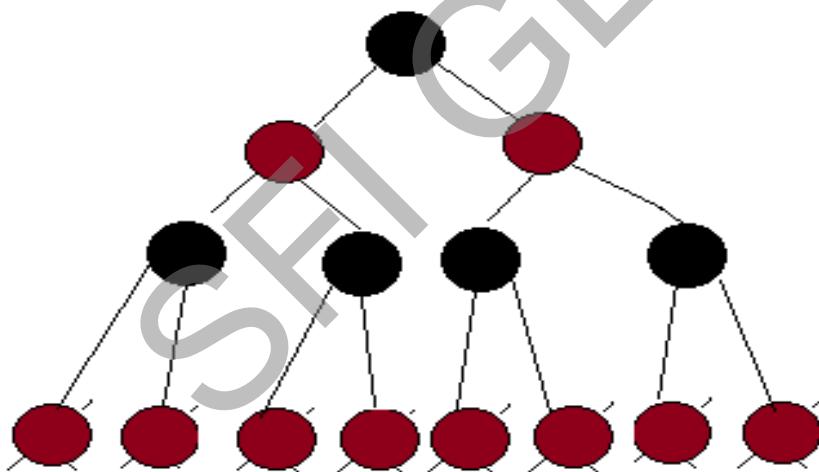
MergeSort(A,q+1,r)

Merge(A,p,q,r)

4. The smallest number of internal nodes in a red-black tree with black height of k is $2^k - 1$ which is one in the following image:



The largest number of internal nodes with black height of k is $2^{2k} - 1$ which, if the black height is 2, should be $2^4 - 1 = 15$. However, consider this image:



5.a) Given recurrence

$$\begin{aligned} T(n) &= 3T(n/2) + O(n) \\ \text{let } cn &= O(n) \end{aligned}$$

for some constant c I can bound

$$T(n)$$

in terms of

$$T(n/2)$$

so I have

$$T(n) \leq 3T(n/2) + cn, \quad k=1 \text{ call}$$

So I expanded the recurrence a few times and got

$$T(n) \leq 9T(n/4) + 2cn, \quad k=2 \text{ calls}$$

$$T(n) \leq 27T(n/8) + 3cn, \quad k=3 \text{ calls}$$

The emerged pattern is

$$T(n) \leq 3kT(n/2^k) + kcn, \quad k \text{th call}$$

5.b) Insertion sort can be expressed as a recursive procedure as follows. In order to sort $A[1..n]$, we recursively sort $A[1..n-1]$ and then insert $A[n]$ into the sorted array $A[1..n-1]$

. Write a recurrence for the running time of this recursive version of insertion sort.

There are two steps in this recursive sorting algorithm:

1. Sort the sub-array $A[1..n-1]$
 - Insert $A[n]$
2. into the sorted sub-array from step 1 in proper position

For $n=1$

, step 1 doesn't take any time as the sub-array is an empty array and step 2 takes constant time, i.e. the algorithm runs in $\Theta(1)$

time.

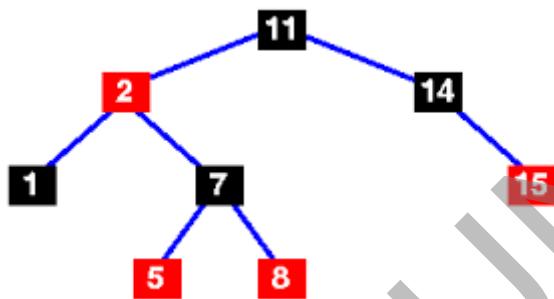
For $n > 1$

, step 1 again calls for the recursion for $n-1$ and step 2 runs in $\Theta(n)$ time.

So, we can write the recurrence as:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n=1 \\ T(n-1) + \Theta(n) & \text{if } n > 1 \end{cases}$$

6. a) A red-black tree is a binary search tree in which each node is colored red or black such that. The root is black. The children of a red node are black. Every path from the root to a 0-node or a 1-node has the same number of black nodes.



- 6.b) Insertion of a node into an n -node red-black tree can be accomplished in $O(\lg n)$ time. We use the TREE-INSERT procedure (Section 13.3) to insert node x into the tree T as if it were an ordinary binary search tree, and then we color x red. To guarantee that the red-black properties are preserved, we then fix up the modified tree by recoloring nodes and performing rotations. Most of the code for RB-INSERT handles the various cases that can arise as we fix up the modified tree.

```

RB-INSERT( $T, x$ )
  1  TREE-INSERT( $T, x$ )
  2   $color[x] \leftarrow RED$ 
  3  while  $x \neq root[T]$  and  $color[p[x]] = RED$ 
  4      do if  $p[x] = left[p[p[x]]]$ 
  5          then  $y \leftarrow right[p[p[x]]]$ 
  6          if  $color[y] = RED$ 
  7              then  $color[p[x]] \leftarrow BLACK$            ▷ Case 1
  8                   $color[y] \leftarrow BLACK$            ▷ Case 1
  9                   $color[p[p[x]]] \leftarrow RED$            ▷ Case 1
 10                  $x \leftarrow p[p[x]]$            ▷ Case 1
 11             else if  $x = right[p[x]]$ 
 12                 then  $x \leftarrow p[x]$            ▷ Case 2
 13                     LEFT-ROTATE( $T, x$ )           ▷ Case 2
 14                      $color[p[x]] \leftarrow BLACK$            ▷ Case 3
 15                      $color[p[p[x]]] \leftarrow RED$            ▷ Case 3
 16                     RIGHT-ROTATE( $T, p[p[x]]$ )           ▷ Case 3
 17             else (same as then clause
 18                 with "right" and "left" exchanged)
 19      $color[root[T]] \leftarrow BLACK$ 
  
```

7.a)

Given an array A of n elements with values or records A_0, A_1, \dots, A_{n-1} , sorted such that $A_0 \leq A_1 \leq \dots \leq A_{n-1}$, and target value T, the following subroutine uses binary search to find the index of T in A.[7]

1. Set L to 0 and R to $n - 1$.
2. If $L > R$, the search terminates as unsuccessful.
3. Set m (the position of the middle element) to the floor (the largest previous integer) of $(L + R)/2$.
4. If $A_m < T$, set L to $m + 1$ and go to step 2.
5. If $A_m > T$, set R to $m - 1$ and go to step 2.

Now $A_m = T$, the search is done; return m

7.b) For a B-tree of order m:

- All data is in leaves. Keys (only) can be replicated in interior nodes.
- The root is either
 - a leaf, or
 - an interior node with $2 \dots m$ children
- All other nodes have children
- All leaves are at the same depth.

8.BFS:

```

unmark all vertices
choose some starting vertex x
mark x
list L = x
tree T = x
while L nonempty
choose some vertex v from front of list
visit v
for each unmarked neighbor w
  mark w
  add it to end of list
  add edge vw to T

```

9. A **topological sort** or **topological ordering** of a [directed graph](#) is a [linear ordering](#) of its [vertices](#) such that for every directed edge uv from vertex u to vertex v , u comes before v in the ordering. For instance, the vertices of the graph may represent tasks to be performed, and the edges may represent constraints that one task must be performed before another; in this application, a topological ordering is just a valid sequence for the tasks. A topological ordering is possible if and only if the graph has no [directed cycles](#), that is, if it is a [directed acyclic graph](#) (DAG). Any DAG has at least one topological ordering, and [algorithms](#) are known for constructing a topological ordering of any DAG in [linear time](#).

10. In the mathematical theory of [directed graphs](#), a graph is said to be **strongly connected** or **diconnected** if every vertex is [reachable](#) from every other vertex. The **strongly connected components** or **diconnected components** of an arbitrary directed graph form a [partition](#) into subgraphs that are themselves strongly connected. It is possible to test the strong connectivity of a graph, or to find its strongly connected components, in [linear time](#).

11. Recursion :

During recursion, there may exist a case where same sub-problems are solved multiple times.

Consider the example of calculating nth fibonacci number.

$$\text{fibo}(n) = \text{fibo}(n-1) + \text{fibo}(n-2)$$

$$\text{ffibo}(n-1) = \text{fibo}(n-2) + \text{fibo}(n-3)$$

$$\text{fibo}(n-2) = \text{fibo}(n-3) + \text{ffibo}(n-4)$$

.....

.....

$$\text{fibo}(2) = \text{fibo}(1) + \text{fibo}(0)$$

In Dynamic programming, once a result of a function is evaluated , it gets stored in a table and can be retrieved from there whenever required in further iterations.

Whereas , in recursion, the functions maybe calculated many times as required by the program

During recursion, there may exist a case where same sub-problems are solved multiple times.

But **Dynamic programming** is a technique where you store the result of previous calculation to avoid calculating the same once again. DP is basically a memorization technique which uses a table to store the results of sub-problem so that if same sub-problem is encountered again in future, it could directly return the result instead of re-calculating it

13 a) Dynamic Programming is also used in optimization problems. Like divide-and-conquer method, Dynamic Programming solves problems by combining the solutions of subproblems. Moreover, Dynamic Programming algorithm solves each sub-problem just once and then saves its answer in a table, thereby avoiding the work of re-computing the answer every time.

Two main properties of a problem suggest that the given problem can be solved using Dynamic Programming. These properties are **overlapping sub-problems and optimal substructure**.

Overlapping Sub-Problems

Similar to Divide-and-Conquer approach, Dynamic Programming also combines solutions to sub-problems. It is mainly used where the solution of one sub-problem is needed repeatedly. The computed solutions are stored in a table, so that these don't have to be re-computed. Hence, this technique is needed where overlapping sub-problem exists.

For example, Binary Search does not have overlapping sub-problem. Whereas recursive program of Fibonacci numbers have many overlapping sub-problems.

Optimal Sub-Structure

A given problem has Optimal Substructure Property, if the optimal solution of the given problem can be obtained using optimal solutions of its sub-problems.

For example, the Shortest Path problem has the following optimal substructure property –

If a node **x** lies in the shortest path from a source node **u** to destination node **v**, then the shortest path from **u** to **v** is the combination of the shortest path from **u** to **x**, and the shortest path from **x** to **v**.

The standard All Pair Shortest Path algorithms like Floyd-Warshall and Bellman-Ford are typical examples of Dynamic Programming.

Steps of Dynamic Programming Approach

Dynamic Programming algorithm is designed using the following four steps –

- Characterize the structure of an optimal solution.
- Recursively define the value of an optimal solution.
- Compute the value of an optimal solution, typically in a bottom-up fashion.
- Construct an optimal solution from the computed information.

Applications of Dynamic Programming Approach

- Matrix Chain Multiplication
- Longest Common Subsequence
- Travelling Salesman Problem

14.0-1 Knapsack cannot be solved by Greedy approach. Greedy approach does not ensure an optimal solution. In many instances, Greedy approach may give an optimal solution.

The following examples will establish our statement.

Example

Let us consider that the capacity of the knapsack is $W = 25$ and the items are as shown in the following table.

Item	A	B	C	D
Profit	24	18	18	10
Weight	24	10	10	7

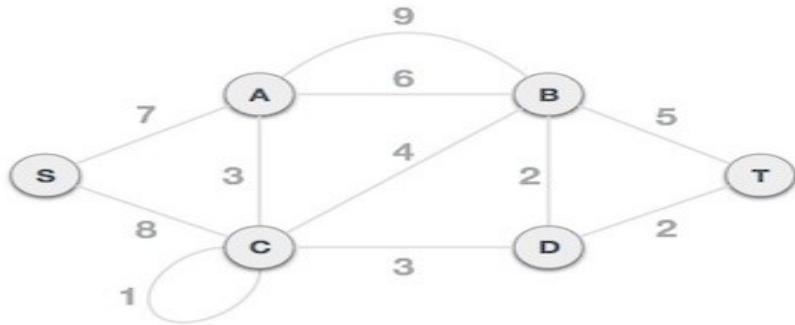
Without considering the profit per unit weight (p_i/w_i), if we apply Greedy approach to solve this problem, first item A will be selected as it will contribute maximum profit among all the elements.

After selecting item A, no more item will be selected. Hence, for this given set of items total profit is **24**. Whereas, the optimal solution can be achieved by selecting items, **B** and **C**, where the total profit is $18 + 18 = 36$.

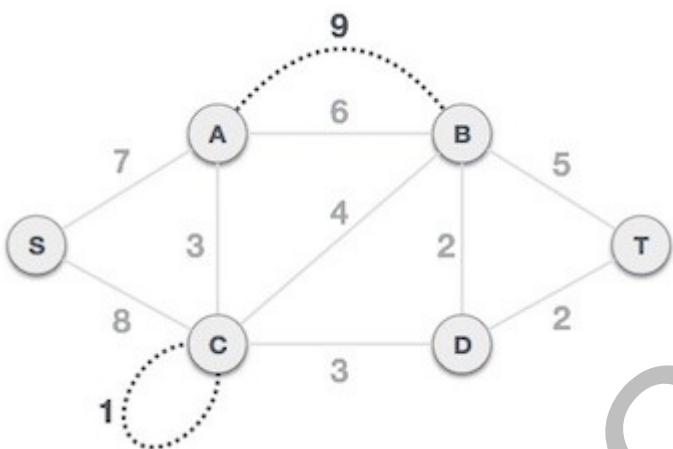
15. Prim's algorithm to find minimum cost spanning tree (as Kruskal's algorithm) uses the greedy approach. Prim's algorithm shares a similarity with the **shortest path first** algorithms.

Prim's algorithm, in contrast with Kruskal's algorithm, treats the nodes as a single tree and keeps on adding new nodes to the spanning tree from the given graph.

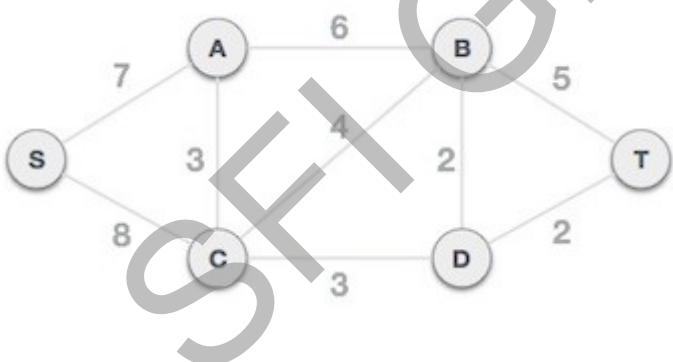
To contrast with Kruskal's algorithm and to understand Prim's algorithm better, we shall use the same example –



Step 1 - Remove all loops and parallel edges



Remove all loops and parallel edges from the given graph. In case of parallel edges, keep the one which has the least cost associated and remove all others.

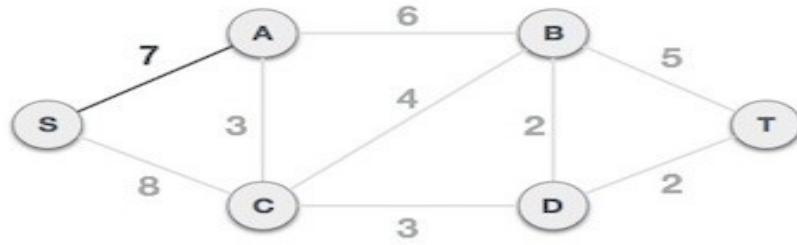


Step 2 - Choose any arbitrary node as root node

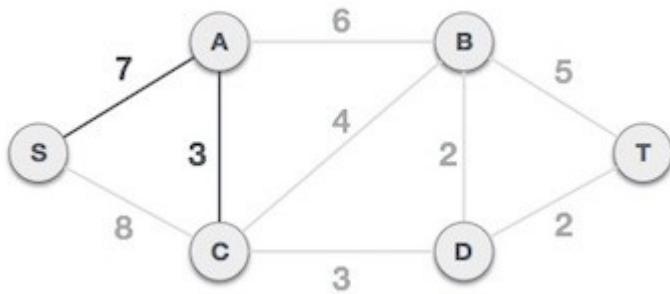
In this case, we choose S node as the root node of Prim's spanning tree. This node is arbitrarily chosen, so any node can be the root node. One may wonder why any video can be a root node. So the answer is, in the spanning tree all the nodes of a graph are included and because it is connected then there must be at least one edge, which will join it to the rest of the tree.

Step 3 - Check outgoing edges and select the one with less cost

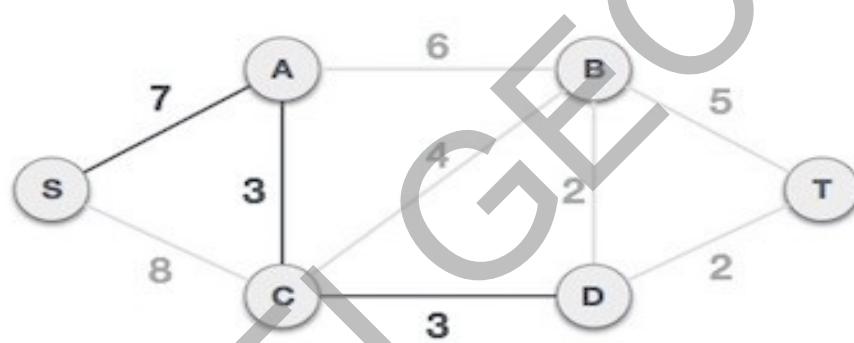
After choosing the root node S, we see that S,A and S,C are two edges with weight 7 and 8, respectively. We choose the edge S,A as it is lesser than the other.



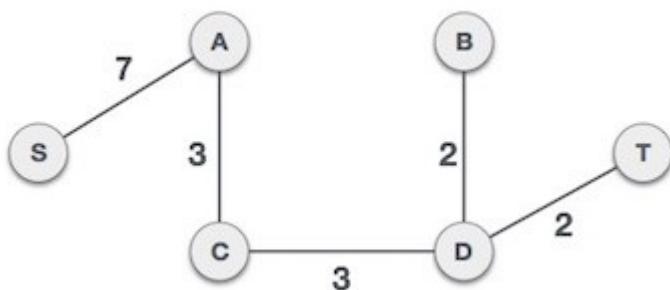
Now, the tree S-7-A is treated as one node and we check for all edges going out from it. We select the one which has the lowest cost and include it in the tree.



After this step, S-7-A-3-C tree is formed. Now we'll again treat it as a node and will check all the edges again. However, we will choose only the least cost edge. In this case, C-3-D is the new edge, which is less than other edges' cost 8, 6, 4, etc.



After adding node D to the spanning tree, we now have two edges going out of it having the same cost, i.e. D-2-T and D-2-B. Thus, we can add either one. But the next step will again yield edge 2 as the least cost. Hence, we are showing a spanning tree with both edges included.



16. Items can be broken into smaller pieces, hence the thief can select fractions of items.

According to the problem statement,

- There are n items in the store
- Weight of i^{th} item $w_i > 0$
- Profit for i^{th} item $p_i > 0$
- and
- Capacity of the Knapsack is W

In this version of Knapsack problem, items can be broken into smaller pieces. So, the thief may take only a fraction x_i of i^{th} item.

$$0 \leq x_i \leq 1$$

The i^{th} item contributes the weight $x_i \cdot w_i$

to the total weight in the knapsack and profit $x_i \cdot p_i$

to the total profit.

Hence, the objective of this algorithm is to

$$\text{maximize } \sum_{i=1}^n (x_i \cdot p_i)$$

subject to constraint,

$$\sum_{i=1}^n (x_i \cdot w_i) \leq W$$

It is clear that an optimal solution must fill the knapsack exactly, otherwise we could add a fraction of one of the remaining items and increase the overall profit.

Thus, an optimal solution can be obtained by

$$\sum_{i=1}^n (x_i \cdot w_i) = W$$

In this context, first we need to sort those items according to the value of p_i / w_i

, so that $p_{i+1} / w_{i+1} \leq p_i / w_i$

. Here, x is an array to store the fraction of items.

```
Algorithm: Greedy-Fractional-Knapsack (w[1..n], p[1..n], W)
for i = 1 to n
    do x[i] = 0
weight = 0
for i = 1 to n
    if weight + w[i] ≤ W then
        x[i] = 1
        weight = weight + w[i]
    else
        x[i] = (W - weight) / w[i]
        weight = W
        break
return x
```

Analysis

If the provided items are already sorted into a decreasing order of **piwi**, then the whileloop takes a time in $O(n)$; Therefore, the total time including the sort is in $O(n \log n)$.

Example

Let us consider that the capacity of the knapsack $W = 60$ and the list of provided items are shown in the following table –

Item	A	B	C	D
Profit	280	100	120	120
Weight	40	10	20	24
Ratio (<i>piwi</i>)	7	10	6	5

As the provided items are not sorted based on **piwi**,

. After sorting, the items are as shown in the following table.

Item	B	A	C	D
Profit	100	280	120	120
Weight	10	40	20	24
Ratio (<i>piwi</i>)	10	7	6	5

Solution

After sorting all the items according to *piwi*,

. First all of **B** is chosen as weight of **B** is less than the capacity of the knapsack. Next, item **A** is chosen, as the available capacity of the knapsack is greater than the weight of **A**. Now, **C** is chosen as the next item. However, the whole item cannot be chosen as the remaining capacity of the knapsack is less than the weight of **C**.

Hence, fraction of **C** (i.e. $(60 - 50)/20$) is chosen.

Now, the capacity of the Knapsack is equal to the selected items. Hence, no more item can be selected.

The total weight of the selected items is $10 + 40 + 20 * (10/20) = 60$

And the total profit is $100 + 280 + 120 * (10/20) = 380 + 60 = 440$

This is the optimal solution. We cannot gain more profit selecting any different combination of items.

In this case, items can be broken into smaller pieces, hence the thief can select fractions of items.

According to the problem statement,

- There are **n** items in the store
- Weight of **ith** item $w_i > 0$

- Profit for i^{th} item $p_i > 0$
- and
- Capacity of the Knapsack is W

In this version of Knapsack problem, items can be broken into smaller pieces. So, the thief may take only a fraction x_i of i^{th} item.

$$0 \leq x_i \leq 1$$

The i^{th} item contributes the weight $x_i \cdot w_i$

to the total weight in the knapsack and profit $x_i \cdot p_i$

to the total profit.

Hence, the objective of this algorithm is to

$$\text{maximize } \sum_{i=1}^n (x_i \cdot p_i)$$

subject to constraint,

$$\sum_{i=1}^n (x_i \cdot w_i) \leq W$$

It is clear that an optimal solution must fill the knapsack exactly, otherwise we could add a fraction of one of the remaining items and increase the overall profit.

Thus, an optimal solution can be obtained by

$$\sum_{i=1}^n (x_i \cdot w_i) = W$$

In this context, first we need to sort those items according to the value of $p_i w_i$

, so that $p_i + 1 w_i + 1 \leq p_i w_i$

. Here, x is an array to store the fraction of items.

```
Algorithm: Greedy-Fractional-Knapsack (w[1..n], p[1..n], W)
for i = 1 to n
    do x[i] = 0
weight = 0
for i = 1 to n
    if weight + w[i] ≤ W then
        x[i] = 1
        weight = weight + w[i]
    else
        x[i] = (W - weight) / w[i]
        weight = W
        break
return x
```

Analysis

If the provided items are already sorted into a decreasing order of $p_i w_i$

, then the whileloop takes a time in $O(n)$; Therefore, the total time including the sort is in $O(n \log n)$.

Example

Let us consider that the capacity of the knapsack $W = 60$ and the list of provided items are shown in the following table –

Item	A	B	C	D
Profit	280	100	120	120
Weight	40	10	20	24
Ratio ($piwi$)	7	10	6	5

As the provided items are not sorted based on $piwi$

. After sorting, the items are as shown in the following table.

Item	B	A	C	D
Profit	100	280	120	120
Weight	10	40	20	24
Ratio ($piwi$)	10	7	6	5

Solution

After sorting all the items according to $piwi$

. First all of **B** is chosen as weight of **B** is less than the capacity of the knapsack. Next, item **A** is chosen, as the available capacity of the knapsack is greater than the weight of **A**. Now, **C** is chosen as the next item. However, the whole item cannot be chosen as the remaining capacity of the knapsack is less than the weight of **C**.

Hence, fraction of **C** (i.e. $(60 - 50)/20$) is chosen.

Now, the capacity of the Knapsack is equal to the selected items. Hence, no more item can be selected.

The total weight of the selected items is $10 + 40 + 20 * (10/20) = 60$

And the total profit is $100 + 280 + 120 * (10/20) = 380 + 60 = 440$

This is the optimal solution. We cannot gain more profit selecting any different combination of items.

17. A traveler needs to visit all the cities from a list, where distances between all the cities are known and each city should be visited just once. What is the shortest possible route that he visits each city

exactly once and returns to the origin city?

Solution

Travelling salesman problem is the most notorious computational problem. We can use brute-force approach to evaluate every possible tour and select the best one. For n number of vertices in a graph, there are $(n - 1)$

number of possibilities.

Instead of brute-force using dynamic programming approach, the solution can be obtained in lesser time, though there is no polynomial time algorithm.

Let us consider a graph $G = (V, E)$, where V is a set of cities and E is a set of weighted edges. An edge $e(u, v)$ represents that vertices u and v are connected. Distance between vertex u and v is $d(u, v)$, which should be non-negative.

Suppose we have started at city 1 and after visiting some cities now we are in city j . Hence, this is a partial tour. We certainly need to know j , since this will determine which cities are most convenient to visit next. We also need to know all the cities visited so far, so that we don't repeat any of them. Hence, this is an appropriate sub-problem.

For a subset of cities $S \in \{1, 2, 3, \dots, n\}$ that includes 1 , and $j \in S$, let $C(S, j)$ be the length of the shortest path visiting each node in S exactly once, starting at 1 and ending at j .

When $|S| > 1$, we define $C(S, 1) = \infty$ since the path cannot start and end at 1 .

Now, let express $C(S, j)$ in terms of smaller sub-problems. We need to start at 1 and end at j . We should select the next city in such a way that

$$C(S, j) = \min_{i \in S} C(S - \{j\}, i) + d(i, j) \text{ where } i \in S \text{ and } i \neq j$$

Algorithm: Traveling-Salesman-Problem

```

C ({1}, 1) = 0
for s = 2 to n do
    for all subsets S ∈ {1, 2, 3, ..., n} of size s and containing 1
        C (S, 1) = ∞
        for all j ∈ S and j ≠ 1
            C (S, j) = min {C (S - {j}, i) + d(i, j) for i ∈ S and i ≠ j}
Return minj C ({1, 2, 3, ..., n}, j) + d(j, 1)

```

Analysis

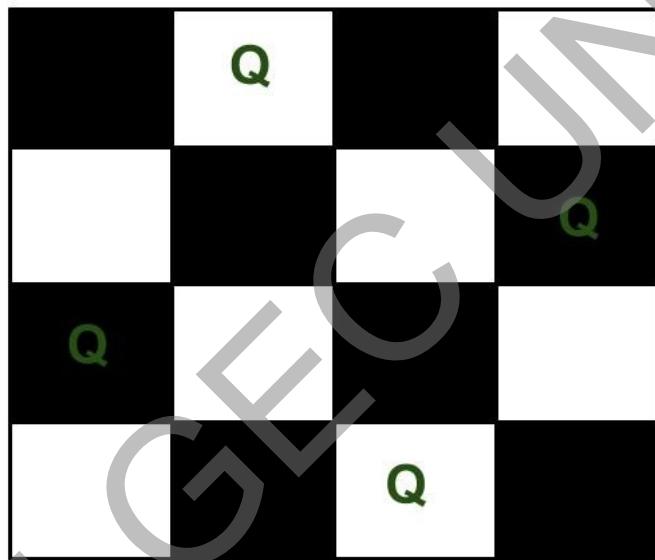
There are at the most $2n.n$

sub-problems and each one takes linear time to solve. Therefore, the total running time is $O(2n.n^2)$

.

Example

18. The N Queen is the problem of placing N chess queens on an $N \times N$ chessboard so that no two queens attack each other. For example, following is a solution for 4 Queen problem.



The expected output is a binary matrix which has 1s for the blocks where queens are placed. For example following is the output matrix for above 4 queen solution.

```
{ 0,  1,  0,  0}  
{ 0,  0,  0,  1}  
{ 1,  0,  0,  0}  
{ 0,  0,  1,  0}
```

19 a) Examples of intractable problems (ones that have been proven to have no polynomial-time algorithm).

– Some of them require a non-polynomial amount of output, so they clearly will take a non-polynomial amount of time, e.g.:

* Towers of Hanoi: we can prove that any algorithm that solves this problem must have a worst-case running time that is at least $2n - 1$.

* List all permutations (all possible orderings) of n numbers.

– Others have polynomial amounts of output, but still cannot be solved in polynomial time:

* For an $n \times n$ draughts board with an arrangement of pieces, determine whether there is a winning strategy for White (i.e. a sequence of moves so that, no matter what Black does, White is guaranteed to win). We can prove that any algorithm that solves this problem must have a worst-case running time that is at least $2n$.

Examples of tractable problems (ones with known polynomial-time algorithms):

- Searching an unordered list
- Searching an ordered list
- Sorting a list
- Multiplication of integers (even though there's a gap)
- Finding a minimum spanning tree in a graph (even though there's a gap)

19.b) A NP problem (not NP-Hard problem) is a **decision problem** which can be verified in polynomial time. Maybe they are solvable in polynomial time, since all problems in P are also in NP.

A NP-complete problem is a **decision problem**, which all NP problems can be reduced to in polynomial time. They are the hardest problems in the class NP.

The NP-hard class is the class of the problems which are at least as hard as the NP-complete problem. They are not necessarily a decision problem. Given that we don't know whether NP = P or not, it would be hard to say whether we can verify a NP-hard problem in polynomial time.

SFI GEC UNIT

SIXTH SEMESTER BTECH DEGREE EXAMINATION, MARCH 2018

CS302: DESIGN AND ANALYSIS OF ALGORITHMS

Time: 3 hours

Max. Marks: 100

PART A

(Answer all questions. Each carries 3 marks)

1. Define Time Complexity and Space Complexity of Algorithms.
2. State and Explain Master Theorem.
3. Insert the following keys in order starting with an empty Red Black Tree: 8,18,5,15, 17, 25, 40 and 80.
4. What is a recursion tree? Construct one for $T(n) = 2T(n/2) + c$.

4x3=12

PART B

(Answer any two. Each carries 9 marks)

5. (a). Analyse the complexity of insertion sort algorithm. (7)
(b). Write a short note on red black tree. (2)
6. (a). Explain various asymptotic methods used to represent the rate of growth of running Time of algorithms. (6)
(b). Explain the properties of a B tree. (3)
7. Solve the following recurrence equations (3*3)
(i) $T(n) = 3T(n/4) + n \log n$.
(ii) $T(n) = 4T(n/2) + n^2$.
(iii) $T(n) = 2T(\sqrt{n}) + 1$.

2*9=18

PART C

(Answer all questions. Each carries 3 marks)

8. What do you mean by strongly connected components?
9. Explain depth first search.
10. Write a short note on optimal matrix multiplication.
11. Apply merge sort to the list 54, 26, 93, 17, 77, 31, 44, 55, 20.

4*3=12

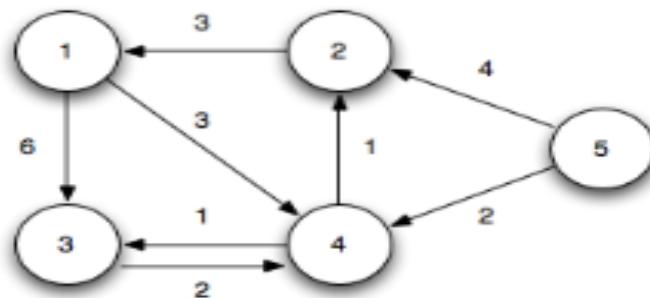
PART D

(Answer any two. Each carries 9 marks)

12. (a). Explain topological sorting algorithm. (4)
(b). Use Strassen's matrix multiplication algorithm to compute the matrix Product

$$\begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix} \quad \begin{bmatrix} 6 & 8 \\ 4 & 2 \end{bmatrix} \quad (5)$$

13. Write an algorithm for BFS traversal on a graph and explain it.
14. Write Bellman - Ford algorithm. Using this algorithm find the shortest path from source S to remaining vertex for the graph shown below.

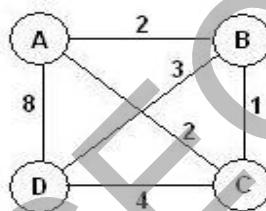


2*9=18

PART E

(Answer any **four**. Each carries 10 marks)

15. Solve travelling salesman problem using greedy algorithm.
16. (a). Compare divide and conquer and dynamic programming strategies (4)
 (b). Explain fractional knapsack problem with example (6)
17. Discuss about the algorithm to find the cost of minimum spanning tree using prim's algorithm. Calculate and draw the minimum cost spanning tree for the graph shown below. Solve



18. Find an optimal solution to the Knapsack instance $n=4$, $w=5$ by greedy method: $(P_1, P_2, P_3, P_4) = (100, 20, 60, 40)$ and $(W_1, W_2, W_3, W_4) = (3, 2, 4, 1)$. Can 0-1 Knapsack problem be solved by greedy method? Justify.
19. Explain 8-queens problem. How back tracking technique can be used to solve the problem?
20. What are NP- Complete problems? Discuss about different NP- Complete problems.

4*10=40

ANSWER KEY

1. Analysis of Algorithms

The complexity of an algorithm is a function describing the efficiency of the algorithm in terms of the amount of data the algorithm must process. Usually there are natural units for the domain and range of this function. Algorithm can be classified by the amount of time they need to complete compared to their input size. The analysis of an algorithm focuses on the complexity of algorithm which depends on time or space.

There are two main complexity measures of the efficiency of an algorithm:

1. Time Complexity: The time complexity is a function that gives the amount of time required by an algorithm to run to completion.

Worst case time complexity: It is the function defined by the maximum amount of time needed by an algorithm for an input of size n.

Average case time complexity: The average-case running time of an algorithm is an estimate of the running time for an "average" input. Computation of average-case running time entails knowing all possible input sequences, the probability distribution of occurrence of these sequences, and the running times for the individual sequences.

Amortized Running Time: It is the time required to perform a sequence of (related) operations is averaged over all the operations performed. Amortized analysis guarantees the average performance of each operation in the worst case.

Best case time complexity: It is the minimum amount of time that an algorithm requires for an input of size n.

```

int count = 0;
for (int i=0; i<n; i++)
    for (int j=0; j<i; j++)
        count++;
Let's see how many times count++ will run.
when i=0 , it will run 0 times
when i=1 , it will run 1 times
when i=2 , it will run 2 times
when i=3 , it will run 3 times and so on
Total number of times count++ will run as
0+1+2+...+(N-1)
=  $\frac{N*(N-1)}{2} - \frac{N^2-N}{2}$ 
So the time complexity will be  $O(N^2)$ 

```

Space Complexity: Total amount of computer memory required by an algorithm to complete its execution is called as space complexity of that algorithm. Generally, when a program is under execution it uses the computer memory for THREE reasons. They are as follows

- Instruction Space: It is the amount of memory used to store compiled version of instructions.
- Environmental Stack: It is the amount of memory used to store information of partially executed functions at the time of function call.
- Data Space: It is the amount of memory used to store all the variables and constants.

Consider the following piece of code...

```
int sum(int A[], int n)
{
    int sum = 0, i;
    for(i = 0; i < n; i++)
        sum = sum + A[i];
    return sum;
}
```

In above piece of code it requires

- 'n*2' bytes of memory to store array variable 'a[]'
- 2 bytes of memory for integer parameter 'n'
- 4 bytes of memory for local integer variables 'sum' and 'i' (2 bytes each)
- 2 bytes of memory for return value.

That means, totally it requires ' $2n+8$ ' bytes of memory to complete its execution. Here, the amount of memory depends on the input value of 'n'. This space complexity is said to be Linear Space Complexity.

2. Master Theorem

The master theorem can be employed to solve recursive equations of the form

$$T(n) = aT(n/b) + f(n)$$

where $a \geq 1$, $b > 1$, and $f(n)$ is *asymptotically positive*. Intuitively for divide and conquer algorithms, this equation represents dividing the problem up into a subproblems of size n/b with a combine time of $f(n)$. For example, for merge sort $a = 2$, $b = 2$, and $f(n) = \Theta(n)$. Note that floors and ceilings for n/b do not affect the asymptotic behavior or the results derived using the theorem.

If the recursion is in the form shown above, then the recurrence can be solved depending on one of three cases (which relate the dominance of the recursive term to the combine term):

Case 1

If

$$f(n) = O(n^{\log_b a - \epsilon})$$

for some **constant** $\epsilon > 0$, then the solution to the recurrence is given by

$$T(n) = \Theta(n^{\log_b a})$$

In this case, $f(n)$ is *polynomially bounded* above by $n^{\log_b a}$ (which represents the run time of the recursive term), i.e. the recursive term dominates the run time.

Case 2

If

$$f(n) = \Theta(n^{\log_a a})$$

then the solution to the recurrence is given by

$$T(n) = \Theta(n^{\log_a a} \lg n)$$

In this case, $f(n)$ is "equal" to $n^{\log_a a}$, i.e. neither term dominates thus the extra term to get the bound.

Case 3

If

$$f(n) = \Omega(n^{\log_a a + \epsilon})$$

for some **constant** $\epsilon > 0$ **AND** if $a f(n/b) \leq c f(n)$ for some **constant** $c < 1$ and *sufficiently large n* (this additional constraint is known as the *regularity condition*), then the solution to the recurrence is given by

$$T(n) = \Theta(f(n))$$

In this case, $f(n)$ is *polynomially bounded* below by $n^{\log_a a}$ (which represents the run time of the recursive term) with the additional regularity condition, i.e. the combine term dominates the run time.

Thus in all three cases it is important to compute the *recursive term run time* $n^{\log_a a}$ and compare it *asymptotically* to the *combine term run time* to determine which case holds. If the recursive equation satisfies either case 1 or case 2, the solution can then be written by inspection. If the recursive equation satisfies case 3, then the regularity condition must be verified in order to write down the solution.

3. Red Black Tree is a BST with more complex algorithms to ensure balance. Each node is labeled as Red or Black. Path: A unique series of links (edges) traverses from the root to each node. The number of edges (links) that must be followed is the path length. In Red Black trees paths from the root to elements with 0 or 1 child are of particular interest.

Red Black Tree Rules

- Every node is colored either Red or black
- The root is black
- If a node is red its children must be black. (a.k.a. the red rule)
- Every path from a node to a null link must contain the same number of black nodes.

KAITHANG

insert (8)

Tree is Empty. So insert newNode as Root node with black color.



insert (18)

Tree is not Empty. So insert newNode with red color.



insert (5)

Tree is not Empty. So insert newNode with red color.



insert (15)

Tree is not Empty. So insert newNode with red color.



Here there are two consecutive Red nodes (18 & 15).
The newnode's parent sibling color is Red
and parent's parent is root node.
So we use RECOLOR to make it Red Black Tree.

After RECOLOR

After Recolor operation, the tree is satisfying all Red Black Tree properties.

insert (17)

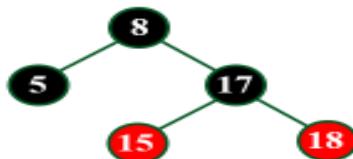
Tree is not Empty. So insert newNode with red color.



Here there are two consecutive Red nodes (15 & 17).
The newnode's parent sibling is NULL. So we need rotation.
Here, we need LR Rotation & Recolor.

After Left Rotation

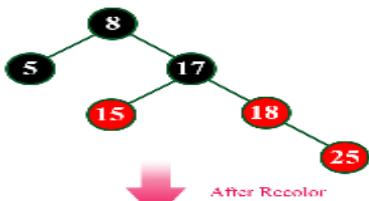
After Right Rotation & Recolor



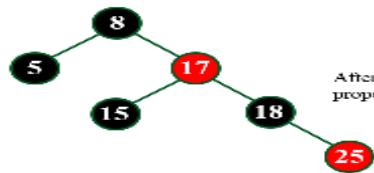
KAITHANG

insert (25)

Tree is not Empty. So insert newNode with red color.



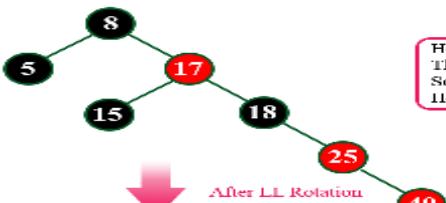
Here there are two consecutive Red nodes (18 & 25).
The newnode's parent sibling color is Red.
and parent's parent is not root node.
So we use RECOLOR and Recheck.



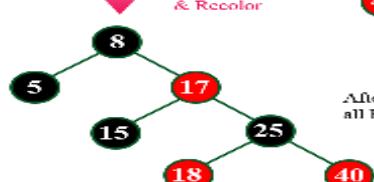
After Recolor operation, the tree is satisfying all Red Black Tree properties.

insert (40)

Tree is not Empty. So insert newNode with red color.



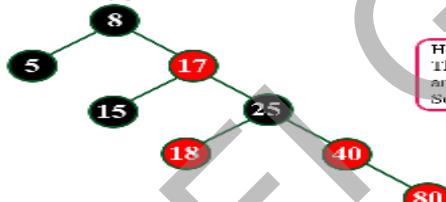
Here there are two consecutive Red nodes (25 & 40).
The newnode's parent sibling is NULL.
So we need a Rotation & Recolor.
Here, we use LL Rotation and Recheck.



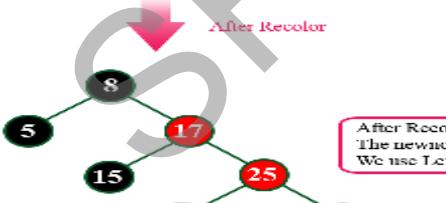
After LL Rotation & Recolor operation, the tree is satisfying all Red Black Tree properties.

insert (80)

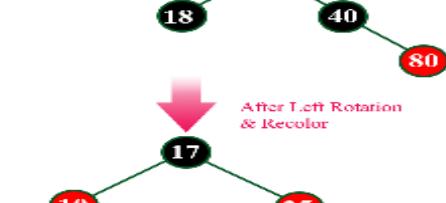
Tree is not Empty. So insert newNode with red color.



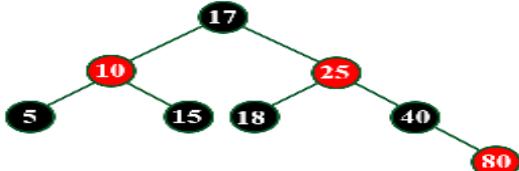
Here there are two consecutive Red nodes (18 & 80).
The newnode's parent sibling color is Red
and parent's parent is not root node.
So we use RECOLOR and Recheck.



After Recolor again there are two consecutive Red nodes (17 & 25).
The newnode's parent sibling color is Black. So we need Rotation.
We use Left Rotation & Recolor.



After Left Rotation & Recolor



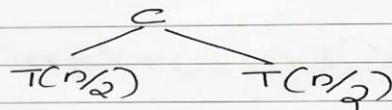
4.

2. Recursion-tree method for Solving recurrences.

A recursion tree is a tree where each node represents the cost of a single subproblem. We sum the costs within each level of the tree to obtain a set of pre-level costs, and then we sum all the pre-level costs to determine the total cost of all levels of the recursion.

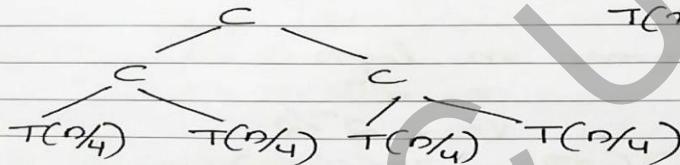
$$T(n) = \begin{cases} 2T(n/2) + c & n > 1 \\ c & n = 1 \end{cases}$$

(a)



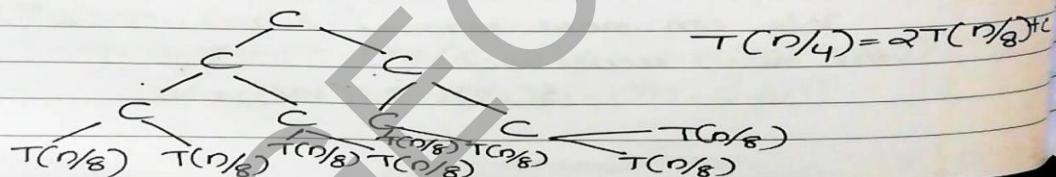
$$T(n) = 2T(n/2) + c$$

(b)



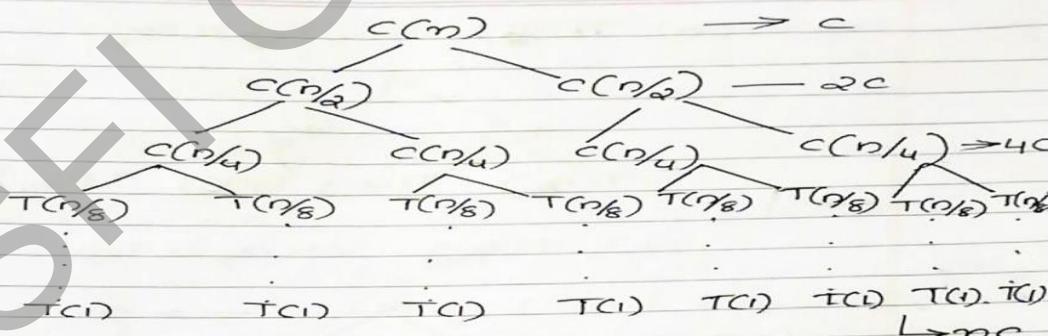
$$T(n/2) = 2T(n/4) + c$$

(c)



$$T(n/4) = 2T(n/8) + c$$

(d)



The total cost is

$$c + 2c + 4c + \dots + nc$$

$$= c(1 + 2 + 4 + \dots + n)$$

Assume that $n = 2^k$

$$\begin{aligned} \underline{\text{G.P}} \quad \frac{a(x^m - 1)}{x - 1} &= \frac{c(1(x^{k+1} - 1))}{x - 1} \\ &= c(x^{k+1} - 1) = c(x^k \cdot x - 1) \\ &= c(x^{2m} - 1) = c(2^m - 1) \\ &= \underline{\underline{O(n)}} \end{aligned}$$

5. a). In insertion sort the element is inserted at an appropriate place similar to card insertion. Here the list is divided into two parts sorted and unsorted sub-lists. In each pass, the first element of unsorted sub list is picked up and moved into the sorted sub list by inserting it in suitable position. Suppose we have ‘n’ elements, we need n-1 passes to sort the elements.

Insertion sort works this way:

It works the way you might sort a hand of playing cards:

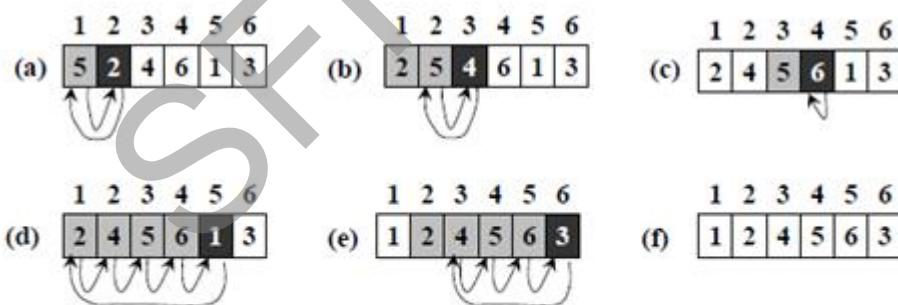
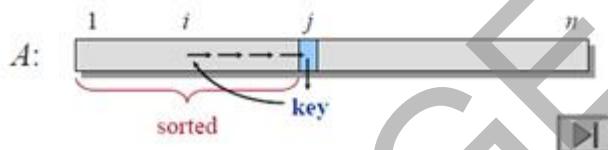
1. We start with an empty left hand [sorted array] and the cards face down on the table [unsorted array].
2. Then remove one card [key] at a time from the table [unsorted array], and insert it into the correct position in the left hand [sorted array].
3. To find the correct position for the card, we compare it with each of the cards already in the hand, from right to left.

INSERTION SORT

```

INSERTION-SORT(A)
1 for j ← 2 to length(A)
2   do key ← A[j]
3     // insert A[j] into the sorted sequence A[1..j-1]
4     i ← j - 1
5     while i > 0 and A[i] > key
6       do A[i+1] ← A[i] // move item back
7       i ← i - 1
8   A[i+1] ← key //find the insertion position

```



- The operation of **INSERTION-SORT** on the array $A = \langle 5, 2, 4, 6, 1, 3 \rangle$.

ANALYSYS OF INSERTION SORT

To compute $T(n)$, the running time

Statement	Effort
<code>InsertionSort(A, n) {</code>	
<code>for i = 2 to n {</code>	c_1n
<code>key = A[i]</code>	$c_2(n-1)$
<code>j = i - 1</code>	$c_3(n-1)$
<code>while (j > 0) and (A[j] > key) {</code>	c_4C
<code>A[j+1] = A[j]</code>	$c_5(C-(n-1))$
<code>j = j - 1</code>	$c_6(C-(n-1))$
<code>A[j+1] = key</code>	$c_7(n-1)$
<code>}</code>	

$C = t_2 + t_3 + \dots + t_n$ where t_i is number of while expression evaluations for the i^{th} for loop iteration

$$\begin{aligned} \text{Body of the while statement is executed} &= (t_2 - 1) + (t_3 - 1) + \dots + (t_{n-1} - 1) \\ &= t_2 + t_3 + \dots + t_n - (n-1) = C - (n-1) \end{aligned}$$

$$T(n) = c_1n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^n t_j + c_6 \sum_{j=2}^n (t_j - 1) + c_7 \sum_{j=2}^n (t_j - 1) + c_8(n-1)$$

The best-case occurs if the array is already sorted.

$$\begin{aligned} T(n) &= c_1n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) \end{aligned}$$

– The running time is a linear function of n

The worst-case results if the array is in reverse sorted order – that is, in decreasing order.

$$\begin{aligned} T(n) &= c_1n + c_2(n-1) + c_4(n-1) + c_5(n(n+1)/2 - 1) + c_6(n(n-1)/2) + c_7(n(n-1)/2) + c_8(n-1) \\ &= (c_5/2 + c_6/2 + c_7/2)n^2 + (c_1 + c_2 + c_4 + c_5/2 - c_6/2 - c_7/2 + c_8)n - (c_2 + c_4 + c_5 + c_8) \end{aligned}$$

– The running time is a quadratic function of n

$$\sum_{j=2}^n t_j = \sum_{j=2}^n j = n(n+1)/2 - 1$$

$$\sum_{j=2}^n t_j - 1 = \sum_{j=2}^n (j-1) = n(n-1)/2$$

Upper bound on the running time for any input For some algorithms, worst-case occur fairly often.e.g. Searching in a database for a particular piece of information Average case often as bad as worst case.

5 (b).

Red Black Tree is a BST with more complex algorithms to ensure balance. Each node is labelled as Red or Black. Path: A unique series of links (edges) traverses from the root to each node. The number of edges (links) that must be followed is the path length. In Red Black trees paths from the root to elements with 0 or 1 child are of particular interest.

Red Black Tree Rules

1. Every node is colored either Red or black
2. The root is black
3. If a node is red its children must be black. (a.k.a. the red rule)
4. Every path from a node to a null link must contain the same number of black nodes.

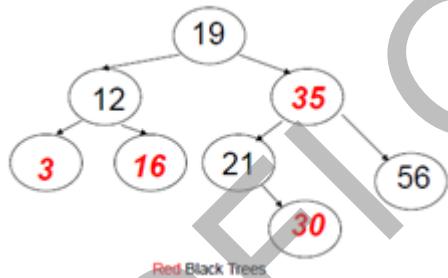
Example of a Red Black Tree

Rule 1: The root of a Red Black tree is black

Rule 2: Every other node in the tree follows these rules:

Rule 3: If a node is Red, all of its children are Black

Rule 4: The number of Black nodes must be the same in all paths from the root node to null nodes



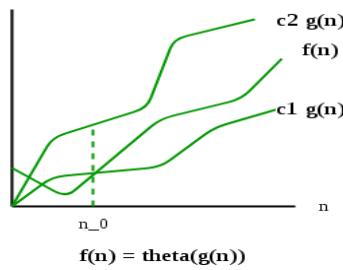
Time Complexity of Red-Black Trees

All non-modifying BST operations (min, max, successor, predecessor, search) run in $O(h) = O(\log n)$ time on red-black trees. Small storage issue per node to include a color flag. Insertion and deletion must maintain rules of red-black trees and are therefore more complex: still $O(\log_2 n)$ time but a bit slower empirically than in ordinary BST. Efficiency of Red Black Trees are. Insertions and removals require additional time due to requirements to recolor and rotate.

Most insertions require on average a single rotation: still $O(\log_2 n)$ time but a bit slower empirically than in ordinary BST.

6. (a). The main idea of asymptotic analysis is to have a measure of efficiency of algorithms that doesn't depend on machine specific constants, and doesn't require algorithms to be implemented and time taken by programs to be compared. Asymptotic notations are mathematical tools to represent time complexity of algorithms for asymptotic analysis. The following 3 asymptotic notations are mostly used to represent time complexity of algorithms.

1) Θ Notation: The theta notation bounds functions from above and below, so it defines exact asymptotic behavior. A simple way to get Theta notation of an expression is to drop low order terms and ignore leading constants.



For example, consider the following expression.

$$3n^3 + 6n^2 + 6000 = \Theta(n^3)$$

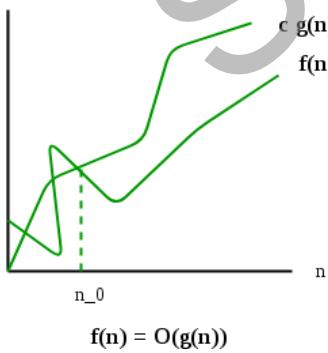
Dropping lower order terms is always fine because there will always be a n_0 after which $\Theta(n^3)$ has higher values than $\Theta(n^2)$ irrespective of the constants involved.

For a given function $g(n)$, we denote $\Theta(g(n))$ is following set of functions.

$$\Theta(g(n)) = \{f(n): \text{there exist positive constants } c_1, c_2 \text{ and } n_0 \text{ such that } 0 \leq c_1*g(n) \leq f(n) \leq c_2*g(n) \text{ for all } n \geq n_0\}$$

The above definition means, if $f(n)$ is theta of $g(n)$, then the value $f(n)$ is always between $c_1*g(n)$ and $c_2*g(n)$ for large values of n ($n \geq n_0$). The definition of theta also requires that $f(n)$ must be non-negative for values of n greater than n_0 .

2) Big O Notation: The Big O notation defines an upper bound of an algorithm, it bounds a function only from above. For example, consider the case of Insertion Sort. It takes linear time in best case and quadratic time in worst case. We can safely say that the time complexity of Insertion sort is $O(n^2)$. Note that $O(n^2)$ also covers linear time.



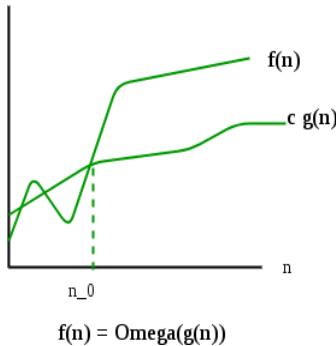
If we use Θ notation to represent time complexity of Insertion sort, we have to use two statements for best and worst cases:

1. The worst case time complexity of Insertion Sort is $\Theta(n^2)$.
2. The best case time complexity of Insertion Sort is $\Theta(n)$.

The Big O notation is useful when we only have upper bound on time complexity of an algorithm. Many times we easily find an upper bound by simply looking at the algorithm.

$O(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \}$

- 3) **Ω Notation:** Just as Big O notation provides an asymptotic upper bound on a function, Ω notation provides an asymptotic lower bound.



Ω Notation can be useful when we have lower bound on time complexity of an algorithm. As discussed in the previous post, the best case performance of an algorithm is generally not useful, the Omega notation is the least used notation among all three.

For a given function $g(n)$, we denote by $\Omega(g(n))$ the set of functions.

$\Omega(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0 \}.$

Let us consider the same Insertion sort example here. The time complexity of Insertion Sort can be written as $\Omega(n)$, but it is not a very useful information about insertion sort, as we are generally interested in worst case and sometimes in average case.

4) Little oh

Asymptotic upper bound is provided by big oh notation may not be asymptotically tight so little oh notation is used to denote an upper bound ie, asymptotically tight.

$o(g(n)) = f(n)$; for any +ve constant $c > 0$, if a constant $n_0 > 0$; such that $0 <= f(n) < c * g(n)$ for all $n > n_0$.

5) Little omega

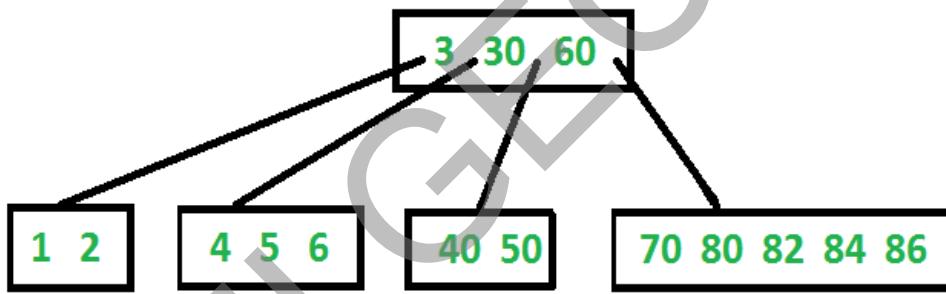
little omega notation is used to denote an lower bound ie, asymptotically tight.

$w(g(n)) = f(n)$; for any +ve constant $c > 0$, if a constant $n_0 > 0$; such that $0 < c * g(n) < f(n)$ for all $n > n_0$.

6. (b). Properties of B-Tree

- 1) All leaves are at same level.
- 2) A B-Tree is defined by the term *minimum degree* ‘t’. The value of t depends upon disk block size.
- 3) Every node except root must contain at least $t-1$ keys. Root may contain minimum 1 key.
- 4) All nodes (including root) may contain at most $2t - 1$ keys.
- 5) Number of children of a node is equal to the number of keys in it plus 1.
- 6) All keys of a node are sorted in increasing order. The child between two keys k_1 and k_2 contains all keys in range from k_1 and k_2 .
- 7) B-Tree grows and shrinks from root which is unlike Binary Search Tree. Binary Search Trees grow downward and also shrink from downward.
- 8) Like other balanced Binary Search Trees, time complexity to search, insert and delete is $O(\log n)$.

Following is an example B-Tree of minimum degree 3. Note that in practical B-Trees, the value of minimum degree is much more than 3.



7. (i)

$$T(n) = 3T(n/4) + n \lg n$$

For this equation $a = 3$, $b = 4$, and $f(n) = n \lg n$. Intuitively this equation would represent an algorithm that divides the original inputs into three groups each consisting of a quarter of the elements and takes linearlog time to combine the results. Computing

$$n^{\log_a a} = n^{\log_4 3} = n^{0.79}$$

Then

$$f(n) = n \lg n = \Omega(n) = \Omega(n^{0.793+\epsilon})$$

which is satisfied for any $\epsilon \leq 0.21$, e.g. choose $\epsilon = 0.2$. Hence the equation *might* satisfy *Case 3* so checking regularity

$$af(n/b) = 3(n/4)\lg(n/4)$$

$$\leq (3/4)n\lg n$$

$$\leq (3/4)f(n)$$

Thus regularity holds by choosing $c = 3/4 < 1$. Therefore the equation satisfies *Case 3* so the solution is

$$T(n) = \Theta(f(n)) = \Theta(n \lg n)$$

7 (ii) Assume the recurrence equation is $T(n) = 4T(n/2) + n^2$

Let us compare this recurrence with our eligible recurrence for Master Theorem $T(n) = aT(n/b) + f(n)$. We find that $a = 4$, $b = 2$ and $f(n) = n^2$

Let us find out $n^{\log_b a}$, which is the work done at last level, using the above values. It is equal to $n^{\log_2 4}$, which is equal to n^2 .

Now let us compare the work done at first and last level. Which means comparing $f(n)$ with $n^{\log_b a}$. $f(n) = n^2$ and $n^{\log_b a} = n^2$.

We know that n^2 is equal to n^2 . Hence, it is the second case of Master Theorem. Therefore the solution to this recurrence is $\theta(n^{\log_b a} * \log_{k+1} n)$. So $T(n) = \theta(n^2 \log_2 n)$.

7 (iii)

Let $n = 2^m$ $m = \log n$
 $T(2^m) = 2T(2^{m/2}) + 1$
 Let $s(m) = T(2^m)$
 $s(m) = 2s(\frac{m}{2}) + 1$
 Apply masters Theorem $a=2$ $b=2$, $f(m)=1=m^0$
 $m^{\log_2 2} = m^1$
 $m^{\log_2 2} = m^1 = \underline{m}$
 $f(m) = 1 = m^0$
 $m^0 < m^1$
 $s(m) = \Theta(m) = \Theta(\log n)$

8. Strongly Connected Components: Connectivity in undirected graphs is rather straightforward: A graph that is not connected is naturally and obviously decomposed in several connected components (Figure 1). As we have seen, depth-first search does this handily: Each restart of the algorithm marks a new connected component.

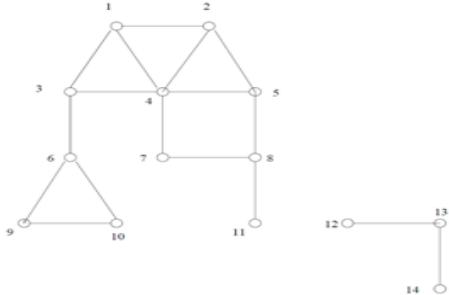


Figure 1: An undirected graph

In directed graphs, however, connectivity is more subtle. In some primitive sense, the directed graph in Figure 2 is “connected” (no part of it can be pulled apart,” so to speak, without breaking” any edges). But this notion is hardly interesting or informative. This graph is certainly not connected, in that there is no path from node 12 to 6, or from 6 to 1. The only meaningful way to define connectivity in directed graphs is this: Call two nodes u and v of a directed graph $G = (V; E)$ connected if there is a path from u to v , and one from v to u . This relation between nodes is reflexive, symmetric, and transitive (check!), so it is an equivalence relation on the nodes. As such, it partitions V into disjoint sets, called the strongly connected components of the graph. In the directed graph of Figure 2 there are four strongly connected components. If we shrink each of these strongly connected components down to a single node, and draw an edge between two of them if there is an edge from some node in the first to some node in the second, the resulting directed graph has to be a directed acyclic graph (dag) |that is to say, it can have no cycles (see Figure 2(b)). The reason is simple: A cycle containing several strongly connected components would merge them all to a single strongly connected component. We can restate this observation as follows:

Every directed graph is a dag of its strongly connected components.

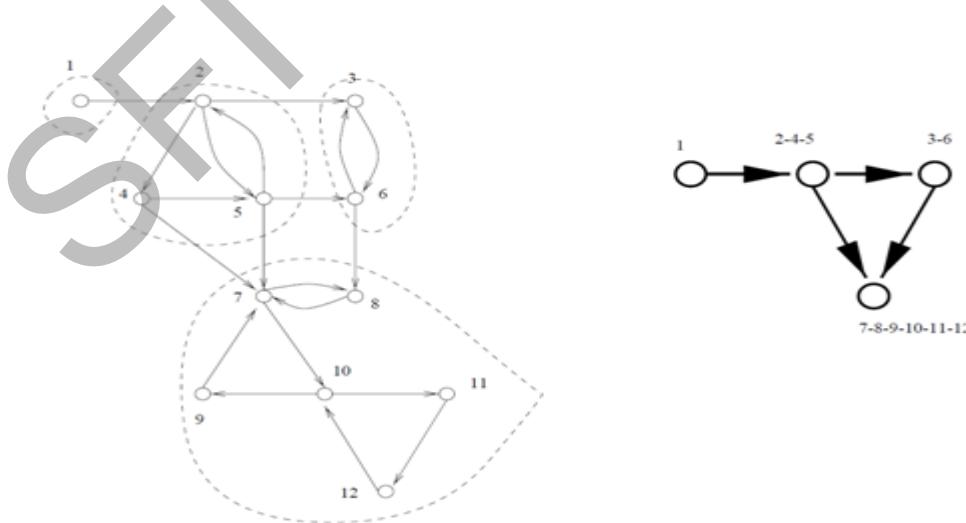


Figure 2: A directed graph and its strongly connected components

This important decomposition theorem allows one to fathom the subtle connectivity information of a directed graph in a two-level manner: At the top level we have a dag [a rather simple structure. For example, we know that a dag is guaranteed to have at least one source (a node without incoming edges) and at least one sink (a node without outgoing edges), and can be topologically sorted. If we want more details, we could look inside a node of the dag to see the full-edged strongly connected component [a tightly connected graph] that lies there. This decomposition is extremely useful and informative; it is thus very fortunate that we have a very client algorithm, based on depth-first search that finds it in linear time! We motivate this algorithm next. It is based on several interesting and slightly subtle properties of depth-first search:

Property 1: If depth-first search of a graph is started at a node u , then it will get stuck and restarted precisely when all nodes that are reachable from u have been visited. Therefore, if depth first search is started at a node of a sink strongly connected component (a strongly connected component that has no edges leaving it in the dag of strongly connected components),then it will get stuck after it visits precisely the nodes of this strongly connected component.

Property 2: The node with the highest post number in depth_rst search (that is, the node where the depth_rst search ends) belongs to a source strongly connected component. Property 2 follows from a more basic fact:

Property 3: Let C and C_0 be two strongly connected components of a graph, and suppose that there is an edge from a node of C to a node of C_0 . Then the node of C visited first by

9.

Dynamic programming, like divide-and-conquer method, solves problems by combining the solutions to sub problems. Meanwhile, dynamic programming avoids the repetition of computing the common sub problems through programming. Here, programming means tabular rather than coding, e.g. Dynamic programming, linear programming, non-linear programming, semi-definite programming, Dynamic programming is a technique for solving problems with the following

properties:

- An instance is solved using the solutions for smaller instances.
- The solution for a smaller instance might be needed multiple times.
- The solutions to smaller instances are stored in a table, so that each smaller instance is solved only once.
- Additional space is used to save time.

Dynamic programming (and greedy technique) is typically used to solve an optimization problem. An optimization problem usually has multiple feasible solutions, and each solution is associated with a value. The goal is to find the solution with the minimum/maximum value. Dynamic Programming Convert memoized algorithm from a recursive one to an iterative one. Dynamic Programming Give a solution of a problem using smaller sub-problems where the parameters of all the possible sub-problems are determined in advance Useful when the same sub-problems show up again and again in the solution.

10.

Breadth first search is the variant of search that is guided by a queue, instead of depth-first search's stack. There is one stylistic difference: One does not restart breadth first search, because breadth first search only makes sense in the context of exploring the part of the graph that is reachable from a particular node (s in the algorithm below). Also, although BFS does not have the wonderful and subtle properties of depth first search, it does provide useful information of another kind: Since it tries to be fair in its choice of the next node, it visits nodes in order of increasing distance from s . In fact, our breadth first search algorithm below labels each node with the shortest distance from s , that is, the number of edges in the shortest path from s to the node. The algorithm is this:

```

Algorithm dfs(G=(V,E): graph, s: node);
v, w: nodes; Q: queue of nodes, initially {s};
dist: array[V] of integer, initially  $\infty$ 
dist[s]:=0
while Q is not empty do
{v:= eject(Q),
for all edges (v,w) out of v do
{if dist[w] =  $\infty$  then
{inject(w,Q), dist[w]:=dist[v]+1}}}

```

For example, applied to the graph in Figure 1, this algorithm labels the nodes (by the array $dist$) as shown. Why are we sure that the $dist[v]$ is the shortest-path distance of v from s ? It is certainly true if $dist[v]$ is zero (this happens only at s). And, if it is true for $dist[v] = d$, then it can be easily shown to be true for values of $dist$ equal to $d + 1$ [any node that receives this value has an edge from a node with $dist d$, and from no node with lower $dist$. Notice that nodes not reachable from s will not be visited or labeled.

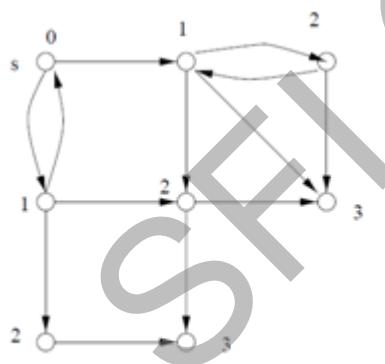


Figure 1: BFS of a directed graph

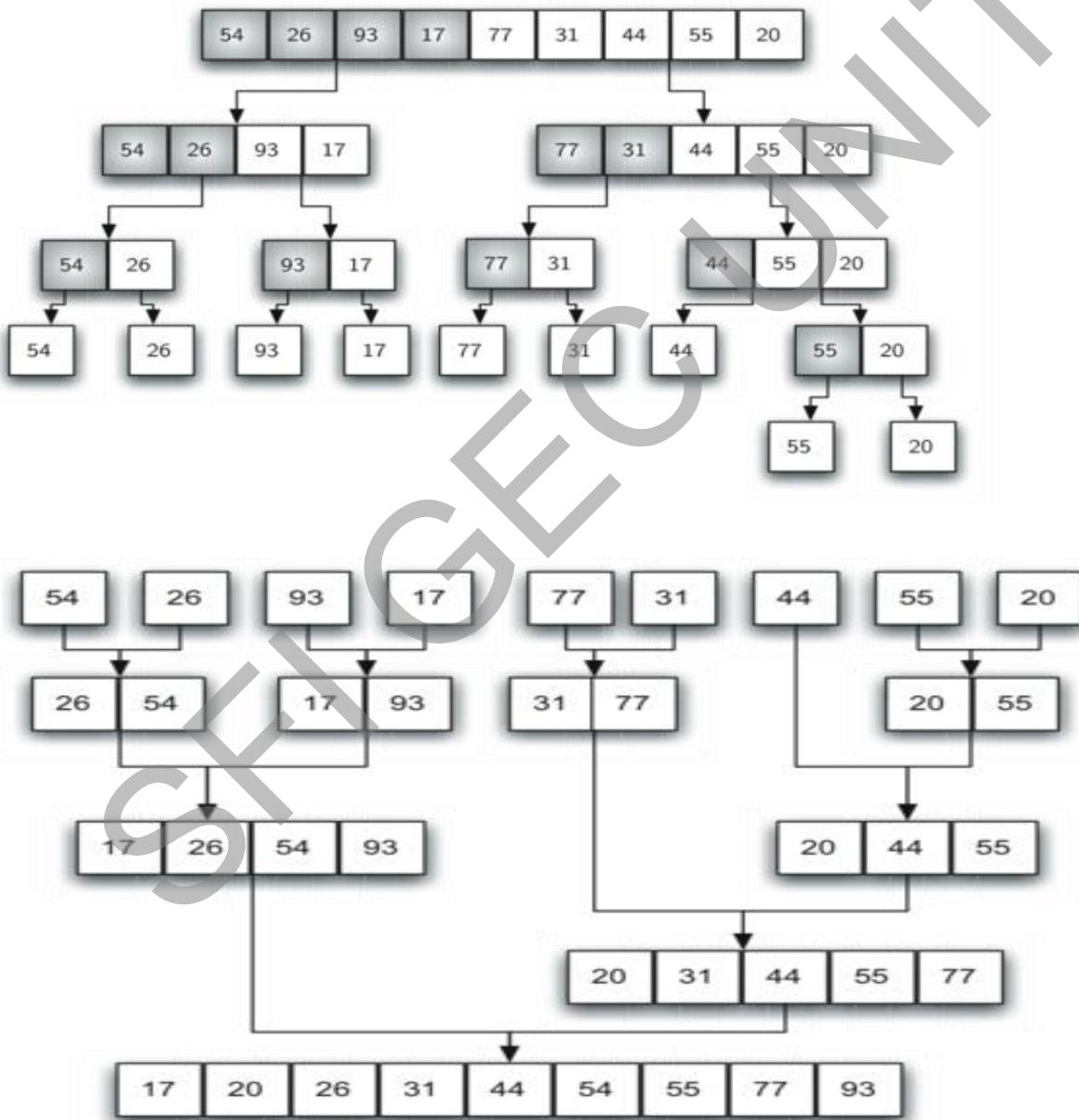
11.

Merge sort is a divide-and-conquer algorithm based on the idea of breaking down a list into several sublists until each sublist consists of a single element and merging those sublists in a manner that results into a sorted list.

Idea:

- Divide the unsorted list into N sublists, each containing 1 element.
- Take adjacent pairs of two singleton lists and merge them to form a list of 2 elements. N will now convert into $N/2$ lists of size 2.
- Repeat the process till a single sorted list of obtained.

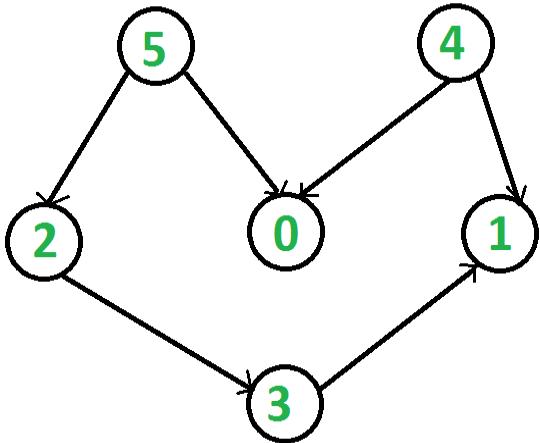
While comparing two sublists for merging, the first element of both lists is taken into consideration. While sorting in ascending order, the element that is of a lesser value becomes a new element of the sorted list. This procedure is repeated until both the smaller sublists are empty and the new combined sublist comprises all the elements of both the sublists.



12. (a).

Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge uv , vertex u comes before v in the ordering. Topological Sorting for a graph is not possible if the graph is not a DAG.

For example, a topological sorting of the following graph is “5 4 2 3 1 0”. There can be more than one topological sorting for a graph. For example, another topological sorting of the following graph is “4 5 2 3 1 0”. The first vertex in topological sorting is always a vertex with in-degree as 0 (a vertex with no incoming edges).



Topological Sorting vs Depth First Traversal (DFS):

In DFS, we print a vertex and then recursively call DFS for its adjacent vertices. In topological sorting, we need to print a vertex before its adjacent vertices. For example, in the given graph, the vertex ‘5’ should be printed before vertex ‘0’, but unlike DFS, the vertex ‘4’ should also be printed before vertex ‘0’. So Topological sorting is different from DFS. For example, a DFS of the shown graph is “5 2 3 1 0 4”, but it is not a topological sorting

12. (b)

$$A = \begin{bmatrix} 1 & 3 \end{bmatrix} \quad B = \begin{bmatrix} 6 & 8 \\ 7 & 5 \end{bmatrix} \quad [4 & 2]$$

$$C = A * B = ?$$

// Step 1: Split A and B into half-sized matrices of size 1x1 (scalars).

```

def a11 = 1
def a12 = 3
def a21 = 7
def a22 = 5
def b11 = 6
def b12 = 8
def b21 = 4
def b22 = 2
  
```

```
// Define the "S" matrix.  
def s1 = b12 - b22 // 6  
def s2 = a11 + a12 // 4  
def s3 = a21 + a22 // 12  
def s4 = b21 - b11 // -2  
def s5 = a11 + a22 // 6  
def s6 = b11 + b22 // 8  
def s7 = a12 - a22 // -2  
def s8 = b21 + b22 // 6  
def s9 = a11 - a21 // -6  
def s10 = b11 + b12 // 14  
  
// Define the "P" matrix.  
def p1 = a11 * s1 // 6  
def p2 = s2 * b22 // 8  
def p3 = s3 * b11 // 72  
def p4 = a22 * s4 // -10  
def p5 = s5 * s6 // 48  
def p6 = s7 * s8 // -12  
def p7 = s9 * s10 // -84  
  
// Fill in the resultant "C" matrix.  
def c11 = p5 + p4 - p2 + p6 // 18  
def c12 = p1 + p2 // 14  
def c21 = p3 + p4 // 62  
def c22 = p5 + p1 - p3 - p7 // 66
```

/* RESULT: =====
 C = [18 14]
 [62 66]

13.

Graph traversals

Graph traversal means visiting every vertex and edge exactly once in a well-defined order. While using certain graph algorithms, you must ensure that each vertex of the graph is visited exactly once. The order in which the vertices are visited are important and may depend upon the algorithm or question that you are solving.

During a traversal, it is important that you track which vertices have been visited. The most common way of tracking vertices is to mark them.

Breadth First Search (BFS)

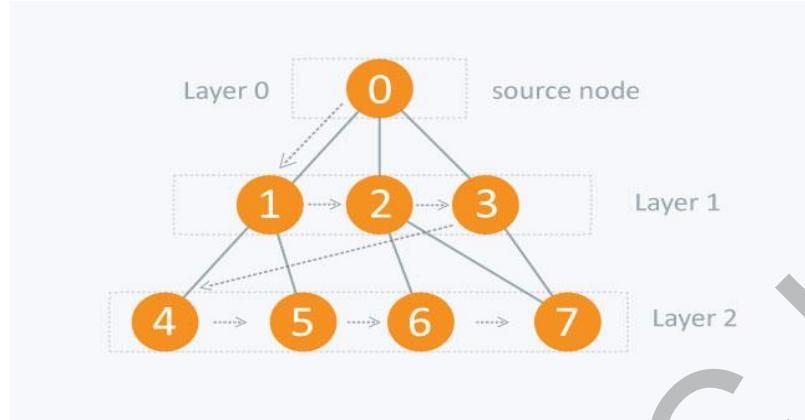
There are many ways to traverse graphs. BFS is the most commonly used approach.

BFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layerwise thus exploring the neighbour nodes (nodes which are directly connected to source node). You must then move towards the next-level neighbour nodes.

As the name BFS suggests, you are required to traverse the graph breadthwise as follows:

1. First move horizontally and visit all the nodes of the current layer
2. Move to the next layer

Consider the following diagram.



The distance between the nodes in layer 1 is comparatively lesser than the distance between the nodes in layer 2. Therefore, in BFS, you must traverse all the nodes in layer 1 before you move to the nodes in layer 2.

Traversing child nodes

A graph can contain cycles, which may bring you to the same node again while traversing the graph. To avoid processing of same node again, use a boolean array which marks the node after it is processed. While visiting the nodes in the layer of a graph, store them in a manner such that you can traverse the corresponding child nodes in a similar order.

In the earlier diagram, start traversing from 0 and visit its child nodes 1, 2, and 3. Store them in the order in which they are visited. This will allow you to visit the child nodes of 1 first (i.e. 4 and 5), then of 2 (i.e. 6 and 7), and then of 3 (i.e. 7) etc.

To make this process easy, use a queue to store the node and mark it as 'visited' until all its neighbours (vertices that are directly connected to it) are marked. The queue follows the First In First Out (FIFO) queuing method, and therefore, the neighbors of the node will be visited in the order in which they were inserted in the node i.e. the node that was inserted first will be visited first, and so on.

Pseudocode

```
BFS (G, s)           //Where G is the graph and s is the source node  
let Q be queue.  
Q.enqueue( s ) //Inserting s in queue until all its neighbour vertices are marked.
```

mark s as visited.

while (Q is not empty)

//Removing that vertex from queue,whose neighbour will be visited now

v = Q.dequeue()

//processing all the neighbours of v

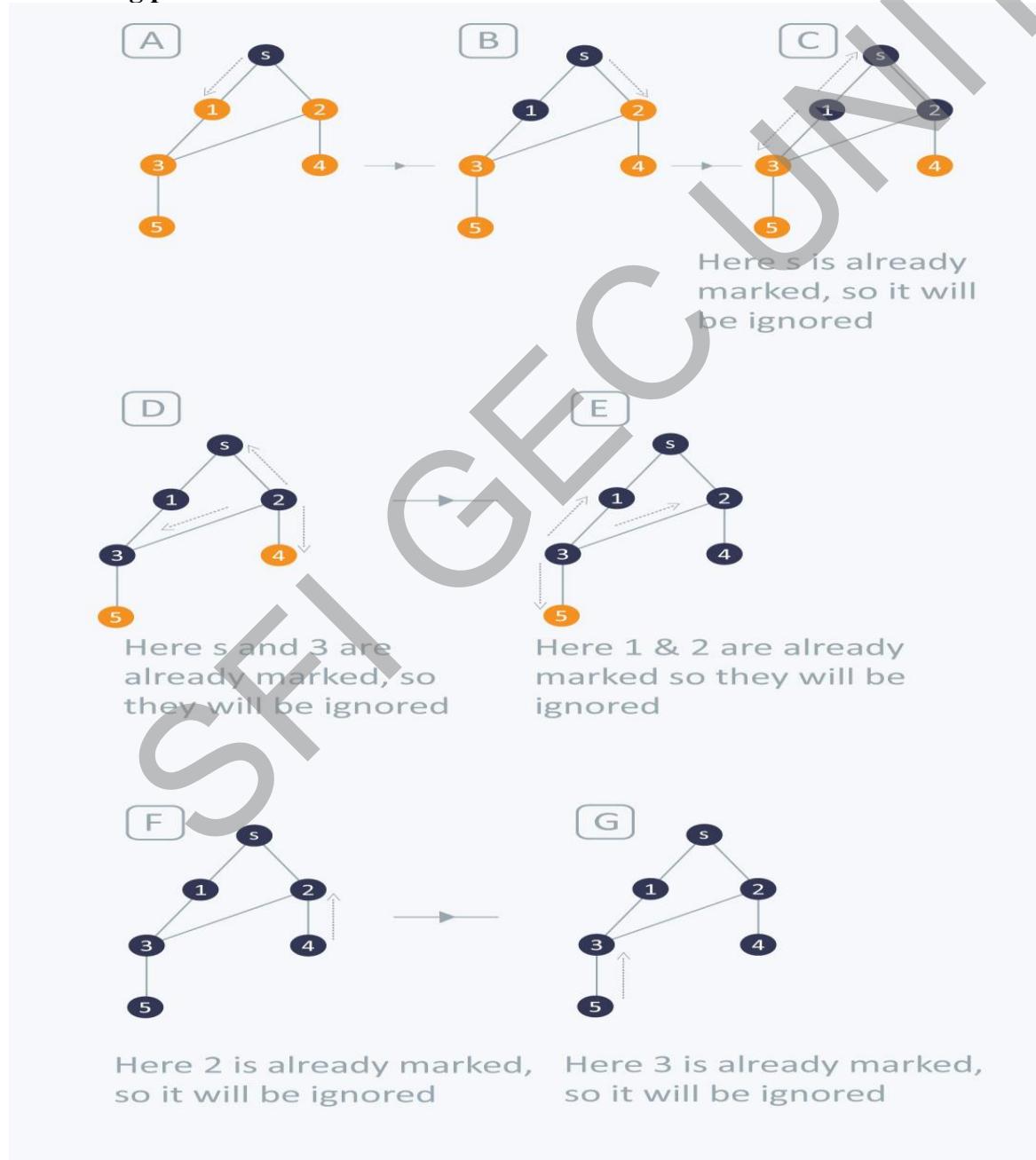
for all neighbours w of v in Graph G

if w is not visited

Q.enqueue(w) //Stores w in Q to further visit its neighbour

mark w as visited.

Traversing process



14. Bellman-Ford Algorithm

The Bellman-Ford algorithm uses relaxation to find single source shortest paths on directed graphs that may contain negative weight edges. The algorithm will also detect if there are any negative weight cycles (such that there is no solution).

BELLMAN-FORD(G,w,s)

1. INITIALIZE-SINGLE-SOURCE(G,s)
2. for $i = 1$ to $|G.V|-1$
3. for each edge $(u,v) \in G.E$
4. RELAX(u,v,w)
5. for each edge $(u,v) \in G.E$
6. if $v.d > u.d + w(u,v)$
7. return FALSE
8. return TRUE

INITIALIZE-SINGLE-SOURCE(G,s)

1. for each vertex $v \in G.V$
2. $v.d = \infty$
3. $v.pi = NIL$
4. $s.d = 0$

RELAX(u,v,w)

1. if $v.d > u.d + w(u,v)$
2. $v.d = u.d + w(u,v)$
3. $v.pi = u$

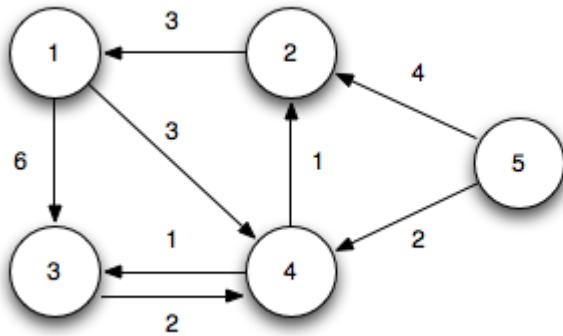
Basically the algorithm works as follows:

1. Initialize d 's, π 's, and set $s.d = 0 \Rightarrow O(V)$
2. Loop $|V|-1$ times through all edges checking the relaxation condition to compute minimum distances $\Rightarrow (|V|-1) O(E) = O(VE)$
3. Loop through all edges checking for negative weight cycles which occurs if any of the relaxation conditions fail $\Rightarrow O(E)$

The run time of the Bellman-Ford algorithm is $O(V + VE + E) = O(VE)$.

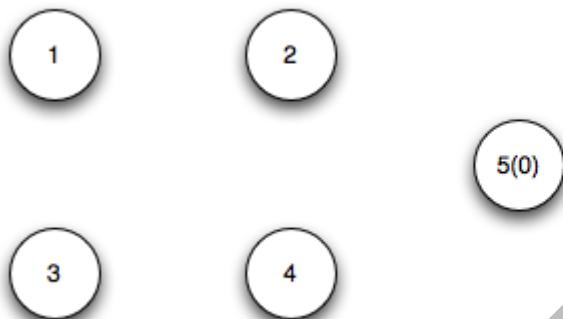
Note that if the graph is a DAG (and thus is known to not have any cycles), we can make Bellman-Ford more efficient by first *topologically sorting* G ($O(V+E)$), performing the same initialization ($O(V)$), and then simply looping through each vertex u in *topological order* relaxing only the edges in $\text{Adj}[u]$ ($O(E)$). This method only takes $O(V + E)$ time. This procedure (with a few slight modifications) is useful for finding *critical paths* for PERT charts.

Given the following directed graph



$$\begin{aligned}
 (1,3) &= 6 \\
 (1,4) &= 3 \\
 (2,1) &= 3 \\
 (3,4) &= 2 \\
 (4,2) &= 1 \\
 (4,3) &= 1 \\
 (5,2) &= 4 \\
 (5,4) &= 2
 \end{aligned}$$

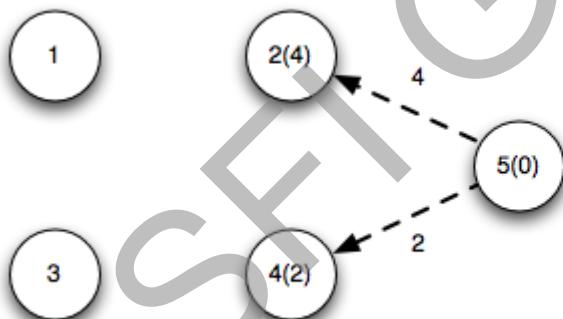
Using vertex 5 as the source (setting its distance to 0), we initialize all the other distances to ∞ .



$$\begin{aligned}
 (1,3) &= 6 \\
 (1,4) &= 3 \\
 (2,1) &= 3 \\
 (3,4) &= 2 \\
 (4,2) &= 1 \\
 (4,3) &= 1 \\
 (5,2) &= 4 \\
 (5,4) &= 2
 \end{aligned}$$

	1	2	3	4	5
d	∞	∞	∞	∞	0
π	/	/	/	/	/

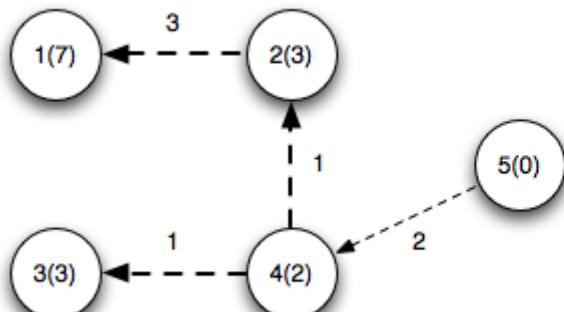
Iteration 1: Edges (u_5, u_2) and (u_5, u_4) relax updating the distances to 2 and 4



$$\begin{aligned}
 (1,3) &= 6 \\
 (1,4) &= 3 \\
 (2,1) &= 3 \\
 (3,4) &= 2 \\
 (4,2) &= 1 \\
 (4,3) &= 1 \\
 (5,2) &= 4 \\
 (5,4) &= 2
 \end{aligned}$$

	1	2	3	4	5
d	∞	4	∞	2	0
π	/	5	/	5	/

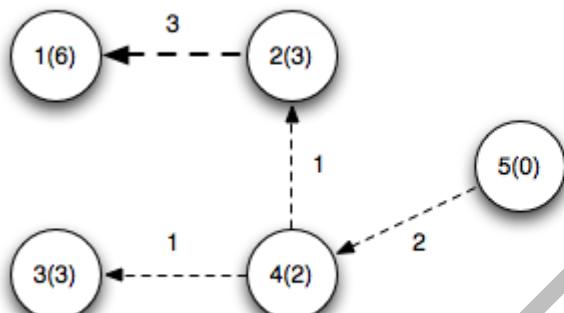
Iteration 2: Edges (u_2, u_1) , (u_4, u_2) and (u_4, u_3) relax updating the distances to 1, 2, and 4 respectively. Note edge (u_4, u_2) finds a shorter path to vertex 2 by going through vertex 4



$$\begin{aligned} (1,3) &= 6 \\ (1,4) &= 3 \\ (2,1) &= 3 \\ (3,4) &= 2 \\ (4,2) &= 1 \\ (4,3) &= 1 \\ (5,2) &= 4 \\ (5,4) &= 2 \end{aligned}$$

	1	2	3	4	5
d	7	3	3	2	0
π	2	4	4	5	/

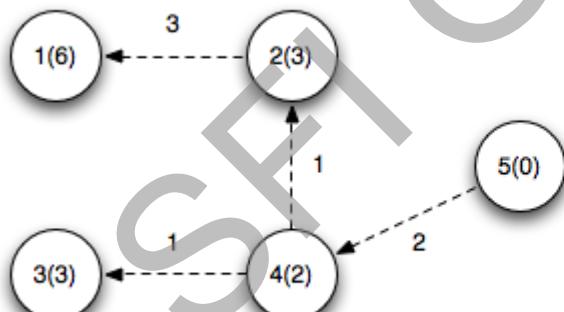
Iteration 3: Edge (u_2, u_1) relaxes (since a shorter path to vertex 2 was found in the previous iteration) updating the distance to 1



$$\begin{aligned} (1,3) &= 6 \\ (1,4) &= 3 \\ (2,1) &= 3 \\ (3,4) &= 2 \\ (4,2) &= 1 \\ (4,3) &= 1 \\ (5,2) &= 4 \\ (5,4) &= 2 \end{aligned}$$

	1	2	3	4	5
d	6	3	3	2	0
π	2	4	4	5	/

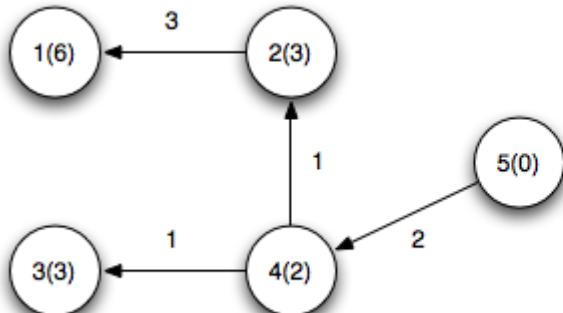
Iteration 4: No edges relax



$$\begin{aligned} (1,3) &= 6 \\ (1,4) &= 3 \\ (2,1) &= 3 \\ (3,4) &= 2 \\ (4,2) &= 1 \\ (4,3) &= 1 \\ (5,2) &= 4 \\ (5,4) &= 2 \end{aligned}$$

	1	2	3	4	5
d	6	3	3	2	0
π	2	4	4	5	/

The final shortest paths from vertex 5 with corresponding distances is



	1	2	3	4	5
d	6	3	3	2	0
π	2	4	4	5	/

Negative cycle checks: We now check the relaxation condition one additional time for each edge. If any of the checks pass then there exists a negative weight cycle in the graph.

$$v_3.d > u_1.d + w(1,3) \Rightarrow 4 \not> 6 + 6 = 12$$

$$v_4.d > u_1.d + w(1,4) \Rightarrow 2 \not> 6 + 3 = 9$$

$$v_1.d > u_2.d + w(2,1) \Rightarrow 6 \not> 3 + 3 = 6$$

$$v_4.d > u_3.d + w(3,4) \Rightarrow 2 \not> 3 + 2 = 5$$

$$v_2.d > u_4.d + w(4,2) \Rightarrow 3 \not> 2 + 1 = 3$$

$$v_3.d > u_4.d + w(4,3) \Rightarrow 3 \not> 2 + 1 = 3$$

$$v_2.d > u_5.d + w(5,2) \Rightarrow 3 \not> 0 + 4 = 4$$

$$v_4.d > u_5.d + w(5,4) \Rightarrow 2 \not> 0 + 2 = 2$$

Note that for the edges *on the shortest paths* the relaxation criteria gives equalities.

Additionally, the path to any reachable vertex can be found by starting at the vertex and following the π 's back to the source. For example, starting at vertex 1, $u_1.\pi = 2$, $u_2.\pi = 4$, $u_4.\pi = 5 \Rightarrow$ the shortest path to vertex 1 is {5,4,2,1}.

15.

Travelling salesman problem.

The traveling salesman problem involves a salesman who must make a tour of a number of cities using the shortest path available and visit each city exactly once and only once and return to the original starting point. For each number of cities n , the number of paths which must be explored is $n!$, causing this problem to grow exponentially rather than as a polynomial. There are bunch of algorithms offering comparably fast running time and still yielding near optimal solutions.

The traveling salesman problem can be described as follows:

TSP = { (G, f, t): G = (V, E) a complete graph, f is a function $V \times V \rightarrow t \in Z$,

G is a graph that contains a traveling salesman tour with cost that does not exceed t}.

different strategies used for travelling salesman problem are:

1. ***Nearest Neighbor strategy.***

This is perhaps the simplest and most straight forward TSP heuristic. The key to this algorithm is to always visit the nearest city, then return to the starting city when all the other cities are visited.

Nearest Neighbor, $O(n^2)$

Step 1. Select a random city.

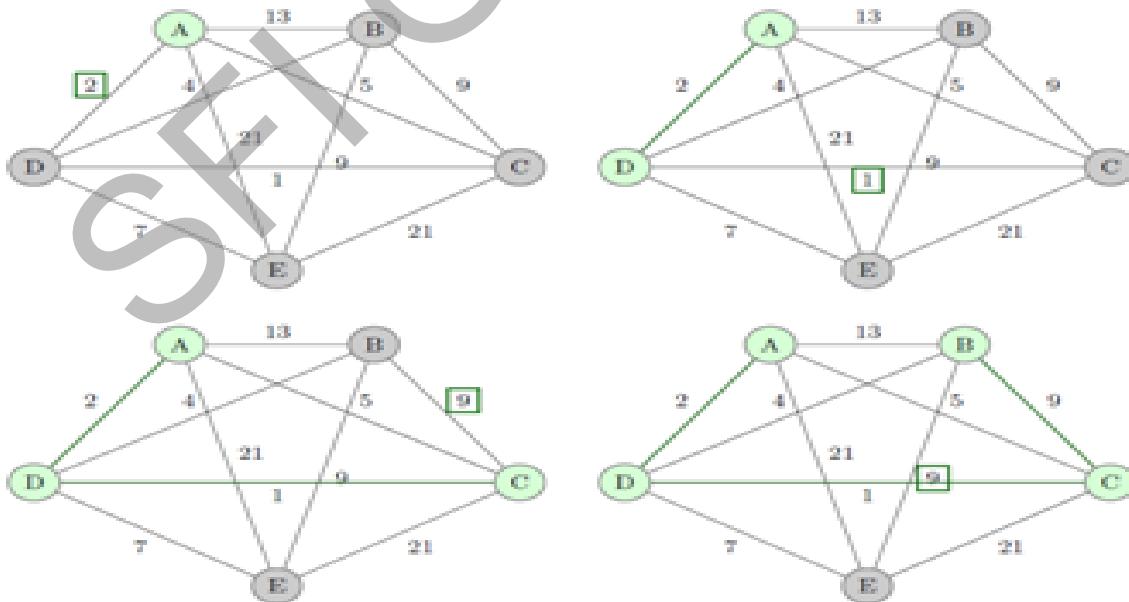
Step 2. Find the nearest unvisited city and go there.

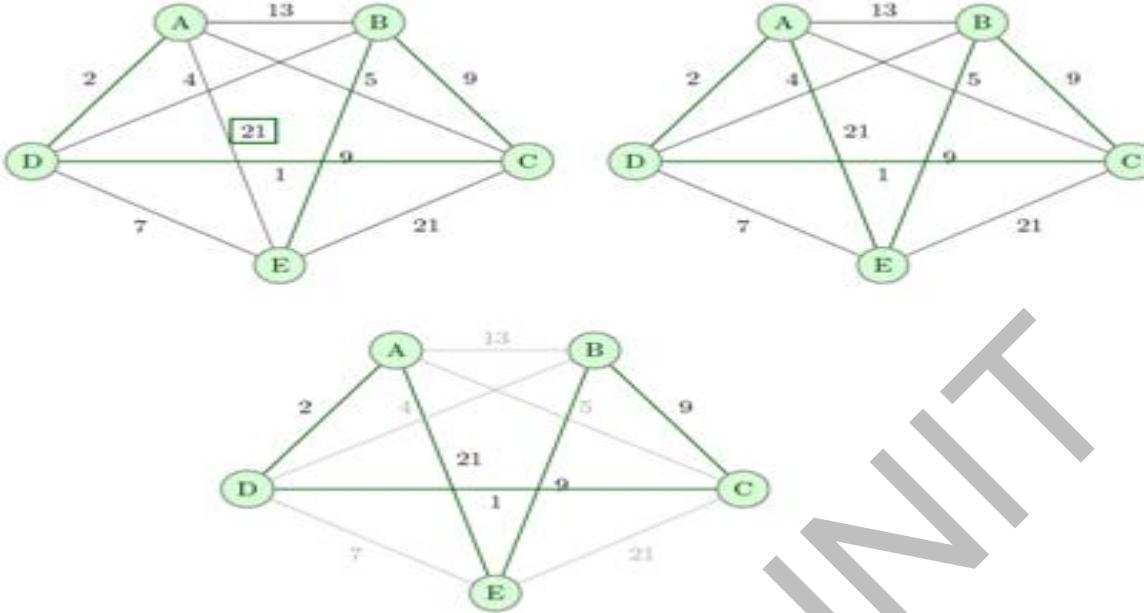
Step 3. Are there any unvisited cities left? If yes, repeat step 2.

Step 4. Return to the first city.

Example for Nearest Neighbor Method

This is the step-wise approximate solution by nearest neighbor method. This case has 5 nodes. We start with the node A and perform the nearest neighbor algorithm.





The total distance of the path A ->D-> C ->B -> E -> A obtained using the nearest neighbor method is $2 + 1 + 9 + 9 + 21 = 42$.

2.Greedy approach.

Greedy algorithm is the simplest improvement algorithm. It starts with the departure Node 1. Then the algorithm calculates all the distances to other n-1 nodes. Go to the next closest node. Take the current node as the departing node, and select the next nearest node from the remaining

n – 2 nodes. The process continues until all the nodes are visited once and only once then back to Node 1. When the algorithm is terminated, the sequence is returned as the best tour.

Greedy, $O(n^2 \log_2(n))$

Greedy method for optimization problem consist of making choices in sequences such that each individual choice is best according to some limited short term criterion that is relatively easy to evaluate.once a choice is made it cannot be undone even if it becomes evident later that it was poor choice.In general greedy strategies are heuristics.they seem to make good sense but many of them don't always lead to optimal solutions or are'nt always efficient.

Prims algorithm begins at an arbitrary start vertex and grows a tree from there.At each iteration of the main loop it chooses an edge from tree vertex to a fringe vertex.it greedily chooses such an edge with minimum weight.

3.Shortest link strategy

Shortest link strategy for undirected graph.it need small changes if the graph is directed.At each iteration of its main loop,the shortest link strategy for TSP grabs a lowest weighted edge from among all remaining edges anywhere in the graph.however the Shortest link strategy must discard an edge if it could not be part of a tour with the other edge already chosen.the sub graph consisting of the edges already chosen at

any point in the algorithm forms a collection of simple paths. there must be no cycles and no vertices incident with more than two chosen edges. the algorithm terminate when all edge been processed.

shortestLinkTSP(V,E,W)

R=E; // R is remaining edges.

C=O; // C is cycle edges

While R is not empty;

Remove the lightest(shortest)edge,vw,from R.

If vw does not make a cycle with edges in C

And vw would not be the third edge in C incident on v or w

Add vw to C//continue loop

Add the edge connecting the endpoints of the path in C.

Return C;

The while loop could be made to end n-1 edges have been chosen. it is easy to maintain a count of the shortest link strategy is about the same kruskal's algorithm.

16. (a) Divide & Conquer

1. The divide-and-conquer paradigm involves three steps at each level of the recursion

- Divide the problem into a number of sub problems.
- Conquer the sub problems by solving them recursively. If the sub problem sizes are small enough, however, just solve the sub problems in a straightforward manner
- Combine the solutions to the sub problems into the solution for the original problem.

2. They call themselves recursively one or more times to deal with closely related sub problems.

3. D&C does more work on the sub-problems and hence has more time consumption.

4. In D&C the sub problems are independent of each other.

5. Example: Merge Sort, Binary Search

Dynamic Programming

1. The development of a dynamic-programming algorithm can be broken into a sequence of four steps.

- a. Characterize the structure of an optimal solution.
 - b. Recursively define the value of an optimal solution.
 - c. Compute the value of an optimal solution in a bottom-up fashion.
 - d. Construct an optimal solution from computed information
2. Dynamic Programming is not recursive.
 3. DP solves the sub problems only once and then stores it in the table.
 4. In DP the sub-problems are not independent.
 5. Example : Matrix chain multiplication

16 (b)

The knapsack problem is a combinatorial optimization problem. It appears as a subproblem in many, more complex mathematical models of real-world problems. One general approach to difficult problems is to identify the most restrictive constraint, ignore the others, solve a knapsack problem, and somehow adjust the solution to satisfy the ignored constraints.

Fractional Knapsack

In this case, items can be broken into smaller pieces, hence the thief can select fractions of items.

According to the problem statement,

- There are n items in the store
- Weight of ith item $w_i > 0$
- Profit for ith item $p_i > 0$ and
- Capacity of the Knapsack is W

In this version of Knapsack problem, items can be broken into smaller pieces. So, the thief may take only a fraction x_i of ith item. $0 \leq x_i \leq 1$. The ith item contributes the weight $x_i \cdot w_i$ to the total weight in the knapsack and profit $x_i \cdot p_i$ to the total profit. Hence, the objective of this algorithm is to maximize $\sum_{i=1}^n (x_i \cdot p_i)$ subject to constraint,

$$\sum_{i=1}^n (x_i \cdot w_i) \leq W$$

It is clear that an optimal solution must fill the knapsack exactly, otherwise we could add a fraction of one of the remaining items and increase the overall profit.

Thus, an optimal solution can be obtained by

$$\sum_{i=1}^n (x_i \cdot w_i) = W$$

In this context, first we need to sort those items according to the value of p_i/w_i , so that $p_{i+1}w_{i+1} \leq p_i/w_i$. Here, x is an array to store the fraction of items.

Algorithm: Greedy-Fractional-Knapsack ($w[1..n]$, $p[1..n]$, W)

```

for i = 1 to n
    do x[i] = 0
    weight = 0
    for i = 1 to n
        if weight + w[i] ≤ W then
            x[i] = 1
            weight = weight + w[i]
        else
            x[i] = (W - weight) / w[i]
            weight = W
            break
    return x
  
```

Analysis

If the provided items are already sorted into a decreasing order of p_i/w_i , then the whileloop takes a time in $O(n)$; Therefore, the total time including the sort is in $O(n \log n)$.

Example

Let us consider that the capacity of the knapsack $W = 60$ and the list of provided items are shown in the following table –

Item	A	B	C	D
Profit	280	100	120	120
Weight	40	10	20	24
Ratio (p_i/w_i)	7	10	6	6

As the provided items are not sorted based on p_i/w_i . After sorting, the items are as shown in the following table.

Item	B	A	C	D
Profit	100	280	120	120
Weight	10	40	20	24
Ratio (p_i/w_i)	10	7	6	6

After sorting all the items according to p_i/w_i . First all of B is chosen as weight of B is less than the capacity of the knapsack. Next, item A is chosen, as the available capacity of the knapsack is greater than the weight of A. Now, C is chosen as the next item. However, the whole item cannot be chosen as the remaining capacity of the knapsack is less than the weight of C.

Hence, fraction of C (i.e. $(60 - 50)/20$) is chosen.

Now, the capacity of the Knapsack is equal to the selected items. Hence, no more item can be selected.

The total weight of the selected items is $10 + 40 + 20 * (10/20) = 60$

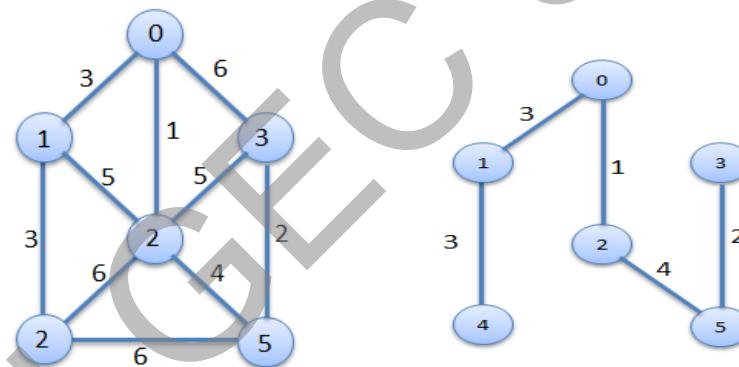
And the total profit is $100 + 280 + 120 * (10/20) = 380 + 60 = 440$

This is the optimal solution. We cannot gain more profit selecting any different combination of items.

17. Prim's Algorithm

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a connected weighted undirected graph. It finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. This algorithm is directly based on the MST(minimum spanning tree) property.

Example



A Simple Weighted Graph

Minimum-Cost Spanning Tree

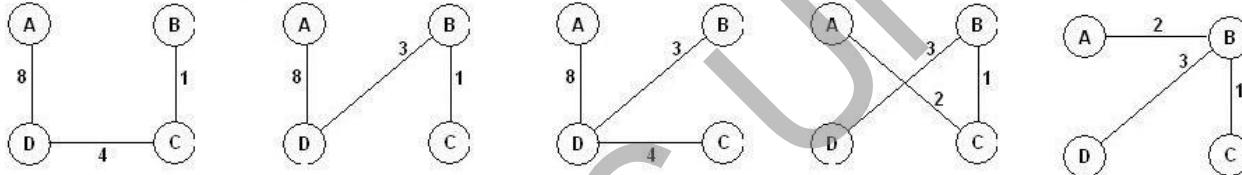
Prim's Algorithm

MST-PRIM(G, w, r)

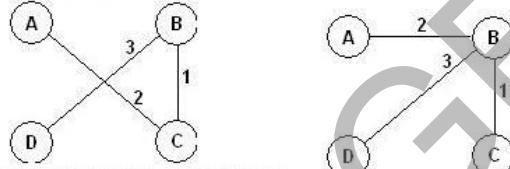
1. for each $u \in V[G]$
2. do $\text{key}[u] \leftarrow \infty$
3. $\pi[u] \leftarrow \text{NIL}$
4. $\text{key}[r] \leftarrow 0$

5. $Q \leftarrow V[G]$
6. while $Q \neq \emptyset$
7. do $u \leftarrow \text{EXTRACT-MIN}(Q)$
8. for each $v \in \text{Adj}[u]$
9. do if $v \in Q$ and $w(u, v) < \text{key}[v]$
10. then $\pi[v] \leftarrow u$
11. $\text{key}[v] \leftarrow w(u, v)$

Has 16 spanning trees. Some are:



The graph has two minimum-cost spanning trees, each with a cost of 6:



27.4

18.

In this problem we have a Knapsack that has a weight limit W . There are items i_1, i_2, \dots , in each having weight w_1, w_2, \dots, w_n and some benefit (value or profit) associated with it v_1, v_2, \dots, v_n . Our objective is to maximise the benefit such that the total weight inside the knapsack is at most W . Since this is a 0-1 Knapsack problem so we can either take an entire item or reject it completely. We can not break an item and fill the knapsack.

Problem

Assume that we have a knapsack with max weight capacity $W = 5$. Our objective is to fill the knapsack with items such that the benefit (value or profit) is maximum.

Following table contains the items along with their value and weight.

Item i	1	2	3	4
Value val	100	20	60	40
Weight wt	3	2	4	1

KAITHANG

Total items $n = 4$

Total capacity of the knapsack $W = 5$

Now we create a value table $V[i,w]$ where, i denotes number of items and w denotes the weight of the items. Rows denote the items and columns denote the weight. As there are 4 items so, we have 5 rows from 0 to 4. And the weight limit of the knapsack is $W = 5$ so, we have 6 columns from 0 to 5

$V[i,w]$	$W=0$	1	2	3	4	5
$i=0$						
1						
2						
3						
4						

We fill the first row $i = 0$ with 0. This means when 0 item is considered weight is 0. Then we fill the first column $w = 0$ with 0. This means when weight is 0 then items considered is 0.

Rule to fill the $V[i,w]$ table.

if $wt[i] > w$ then

$$V[i,w] = V[i-1,w]$$

else if $wt[i] \leq w$ then

$$V[i,w] = \max(V[i-1,w], val[i] + V[i-1, w - wt[i]])$$

After calculation, the value table V

$V[i,w]$	$W=0$	1	2	3	4	5
$i=0$	0	0	0	0	0	0
1	0	0	0	100	100	100
2	0	0	20	100	100	120
3	0	0	20	100	100	120
4	0	40	40	100	140	140

Maximum value earned

$$\text{Max Value} = V[n,W]$$

$$= V[4,5]$$

$$= 140$$

Items that were put inside the knapsack are found using the following rule

set $i = n$ and $w = W$

while i and $w > 0$ do

 if $V[i,w] \neq V[i-1,w]$ then

 mark the i th item

 set $w = w - wt[i]$

 set $i = i - 1$

 else

 set $i = i - 1$

 endif

endwhile

So, items we are putting inside the knapsack are 4 and 1.

19.

The eight queens problem is the problem of placing eight queens on an 8×8 chessboard such that none of them attack one another (no two are in the same row, column, or diagonal). More generally, the n queens problem places n queens on an $n \times n$ chessboard.

There are different solutions for the problem.

- Backtracking
- Branch and Bound

Solving 8 queen problem by backtracking

The 8 queen problem is a case of more general set of problems namely “ n queen problem”. The basic idea: How to place n queen on n by n board, so that they don’t attack each other. As we can expect the complexity of solving the problem increases with n . We will briefly introduce solution by backtracking.

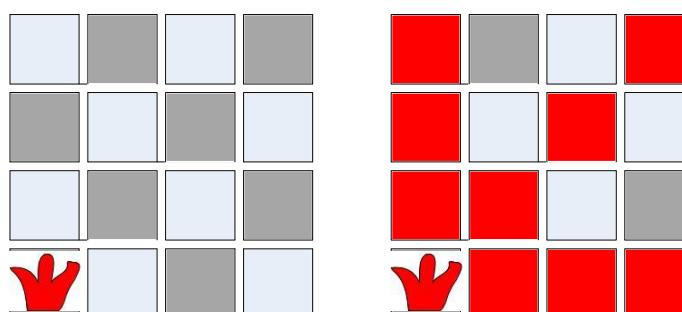
First let’s explain what is backtracking? The board should be regarded as a set of constraints and the solution is simply satisfying all constraints. For example: Q1 attacks some positions, therefore Q2 has to comply with these constraints and take place, not directly attacked by Q1. Placing Q3 is harder, since we have to satisfy constraints of Q1 and Q2. Going the same way we may reach point, where the constraints make the placement of the next queen impossible. Therefore we need to relax the constraints and find new solution. To do this we are going backwards and finding new admissible solution. To keep everything in order we keep the simple rule: last placed, first displaced. In other words if we place successfully queen on the i^{th} column but cannot find solution for $(i+1)^{\text{th}}$ queen, then going backwards we will try to find other admissible solution for the i^{th} queen first. This process is called backtrack

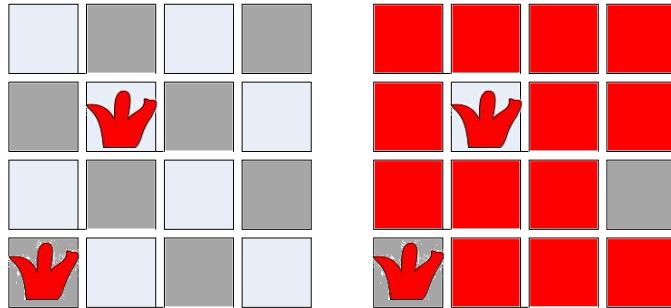
For the purpose of this handout we will find solution of 4 queen problem.

Algorithm:

- Start with one queen at the first column first row
- Continue with second queen from the second column first row
- Go up until find a permissible situation
- Continue with next queen

We place the first queen on A1:

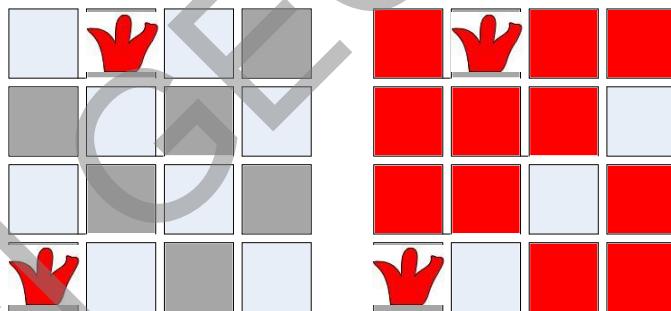




Note the positions which Q1 is attacking. So the next queen Q2 has to options: B3 or B4.
We choose the first one B3

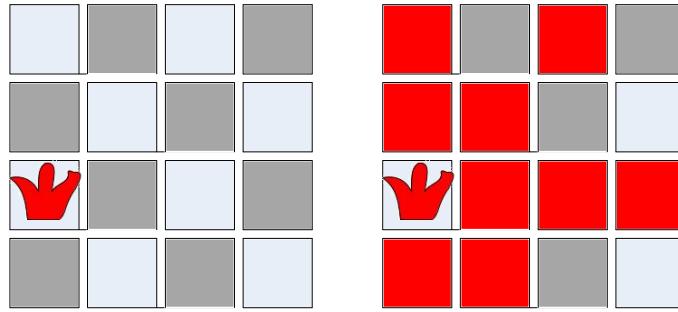
Again with red we show the prohibited positions. It turned out that we cannot place the third queen on the third column (we have to have a queen for each column!). In other words we imposed a set of constraints in a way that we no longer can satisfy them in order to find a solution. Hence we need to revise the constraints or rearrange the board up to the state which we were stuck. Now we may ask a question what we have to change. Since the problem happened after placing Q2 we are trying first with this queen.

OK we know that there were two possible places for Q2. B3 gives problem for the third queen, so there is only one position left – B4:

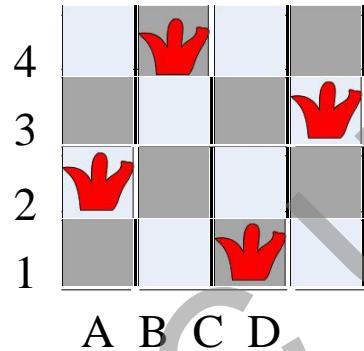


As you can see from the new set of constraints (the red positions) now we have admissible position for Q3, but it will make impossible to place Q4 since the only place is D3. Hence placing Q2 on the only one left position B4 didn't help. Therefore the one step backtrack was not enough. We need to go for second backtrack. Why? The reason is that there is no position for Q2, which will satisfy any position for Q4 or Q3. Hence we need to deal with the position of Q1.

We have started from Q1 so we will continue upward and placing the queen at A2

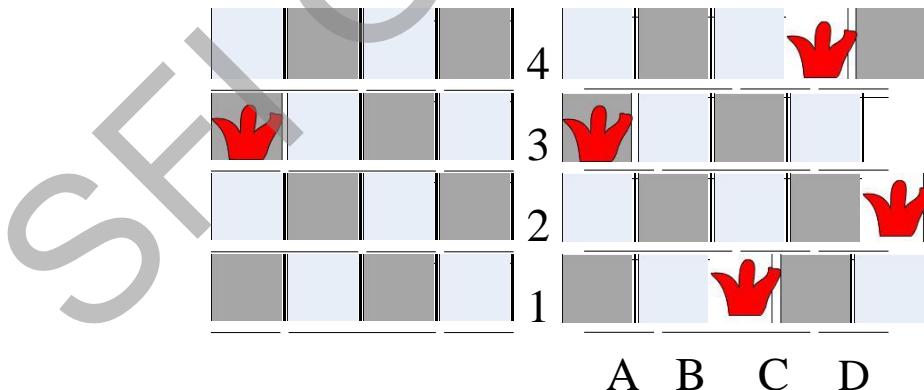


Now it is easy to see that Q2 goes to B4, Q3 goes to C1 and Q4 takes D3:



To find this solution we had to perform two backtracks. So what now? In order to find all solutions we use as you can guess – backtrack!

Start again in reverse order we try to place Q4 somewhere up, which is not possible. We backtrack to Q3 and try to find admissible place different from C1. Again we need to backtrack. Q2 has no other choice and finally we reach Q1. We place Q1 on A3:



Continuing further we will reach the solution on the right. Is this distinct solution? No it is rotated first solution. In fact for 4x4 board there is only one unique solution. Placing Q1 on A4 has the same effect as placing it on A1. Hence we explored all solutions.

How implement backtrack in code. Remember that we used backtrack when we cannot find admissible position for a queen in a column. Otherwise we go further with the next column until we place a queen on the last column. Therefore your code must have fragment:

```
int PlaceQueen(int board[8], int row)

If (Can place queen on ith column)
    PlaceQueen(newboard, 0)
Else
    PlaceQueen(oldboard,oldplace+1)
End
```

If you can place queen on ith column try to place a queen on the next one, or backtrack and try to place a queen on position above the solution found for i-1 column.

20.

NP-Completeness

NP-completeness is a form of bad news: evidence that many important problems can't be solved quickly. These NP-complete problems really come up all the time. Knowing they're hard lets you stop beating your head against a wall trying to solve them, and do something better:

- Use a heuristic. If you can't quickly solve the problem with a good worst case time, maybe you can come up with a method for solving a reasonable fraction of the common cases.
- Solve the problem approximately instead of exactly. A lot of the time it is possible to come up with a provably fast algorithm, that doesn't solve the problem exactly but comes up with a solution you can prove is close to right.
- Use an exponential time solution anyway. If you really have to solve the problem exactly, you can settle down to writing an exponential time algorithm and stop worrying about finding a better solution.
- Choose a better abstraction. The NP-complete abstract problem you're trying to solve presumably comes from ignoring some of the seemingly unimportant details of a more complicated real world problem. Perhaps some of those details shouldn't have been ignored, and make the difference between what you can and can't solve.

Classification of problems

The subject of *computational complexity theory* is dedicated to classifying problems by how hard they are. There are many different classifications; some of the most common and useful are the following. (One technical point: these are all really defined in terms of yes-or-no problems -- does a certain structure exist rather than how do I find the structure.)

- P. Problems that can be solved in polynomial time. ("P" stands for polynomial.) These problems have formed the main material of this course.

KAITHANG

- NP. This stands for "nondeterministic polynomial time" where nondeterministic is just a fancy way of talking about guessing a solution. A problem is in NP if you can quickly (in polynomial time) test whether a solution is correct (without worrying about how hard it might be to find the solution). Problems in NP are still relatively easy: if only we could guess the right solution, we could then quickly test it.

NP does not stand for "non-polynomial". There are many complexity classes that are much harder than NP.

- PSPACE. Problems that can be solved using a reasonable amount of memory (again defined formally as a polynomial in the input size) without regard to how much time the solution takes.
- EXPTIME. Problems that can be solved in exponential time. This class contains most problems you are likely to run into, including everything in the previous three classes. It may be surprising that this class is not all-inclusive: there are problems for which the best algorithms take even more than exponential time.
- Undecidable. For some problems, we can prove that there is no algorithm that always solves them, no matter how much time or space is allowed. One very uninformative proof of this is based on the fact that there are as many problems as there are real numbers, and only as many programs as there are integers, so there are not enough programs to solve all the problems. But we can also define explicit and useful problems which can't be solved.

Although defined theoretically, many of these classes have practical implications. For instance P is a very good approximation to the class of problems which can be solved quickly in practice -- usually if this is true, we can prove a polynomial worst case time bound, and conversely the polynomial time bounds we can prove are usually small enough that the corresponding algorithms really are practical. NP-completeness theory is concerned with the distinction between the first two classes, P and NP.

Examples of problems in different classes

Example 1: Long simple paths.

A *simple path* in a graph is just one without any repeated edges or vertices. To describe the problem of finding long paths in terms of complexity theory, we need to formalize it as a yes-or-no question: given a graph G, vertices s and t, and a number k, does there exist a simple path from s to t with at least k edges? A solution to this problem would then consist of such a path.

Why is this in NP? If you're given a path, you can quickly look at it and add up the length, double-checking that it really is a path with length at least k. This can all be done in linear time, so certainly it can be done in polynomial time.

However we don't know whether this problem is in P; I haven't told you a good way for finding such a path (with time polynomial in m,n, and K). And in fact this problem is NP-complete, so we believe that no such algorithm exists.

There are algorithms that solve the problem; for instance, list all 2^m subsets of edges and check whether any of them solves the problem. But as far as we know there is no algorithm that runs in polynomial time.

Example 2: Cryptography.

Suppose we have an encryption function e.g.

code=RSA(key,text)

KAITHANG

The "RSA" encryption works by performing some simple integer arithmetic on the code and the key, which consists of a pair (p,q) of large prime numbers. One can perform the encryption only knowing the product pq ; but to decrypt the code you instead need to know a different product, $(p-1)(q-1)$.

A standard assumption in cryptography is the "known plaintext attack": we have the code for some message, and we know (or can guess) the text of that message. We want to use that information to discover the key, so we can decrypt other messages sent using the same key.

Formalized as an NP problem, we simply want to find a key for which $\text{code} = \text{RSA}(\text{key}, \text{text})$. If you're given a key, you can test it by doing the encryption yourself, so this is in NP.

The hard question is, how do you find the key? For the code to be strong we hope it isn't possible to do much better than a brute force search.

Another common use of RSA involves "public key cryptography": a user of the system publishes the product pq , but doesn't publish p , q , or $(p-1)(q-1)$. That way anyone can send a message to that user by using the RSA encryption, but only the user can decrypt it. Breaking this scheme can also be thought of as a different NP problem: given a composite number pq , find a factorization into smaller numbers.

One can test a factorization quickly (just multiply the factors back together again), so the problem is in NP. Finding a factorization seems to be difficult, and we think it may not be in P. However there is some strong evidence that it is not NP-complete either; it seems to be one of the (very rare) examples of problems between P and NP-complete in difficulty.

Example 3: Chess.

We've seen in the news recently a match between the world chess champion, Gary Kasparov, and a very fast chess computer, Deep Blue. The computer lost the match, but won one game and tied others.

What is involved in chess programming? Essentially the sequences of possible moves form a tree: The first player has a choice of 20 different moves (most of which are not very good), after each of which the second player has a choice of many responses, and so on. Chess playing programs work by traversing this tree finding what the possible consequences would be of each different move.

The tree of moves is not very deep -- a typical chess game might last 40 moves, and it is rare for one to reach 200 moves. Since each move involves a step by each player, there are at most 400 positions involved in most games. If we traversed the tree of chess positions only to that depth, we would only need enough memory to store the 400 positions on a single path at a time. This much memory is easily available on the smallest computers you are likely to use.

So perfect chess playing is a problem in PSPACE. (Actually one must be more careful in definitions. There is only a finite number of positions in chess, so in principle you could write down the solution in constant time. But that constant would be very large. Generalized versions of chess on larger boards are in PSPACE.)

The reason this deep game-tree search method can't be used in practice is that the tree of moves is very bushy, so that even though it is not deep it has an enormous number of vertices. We won't run out of space if we try to traverse it, but we will run out of time before we get even a small fraction of the way through. Some pruning methods, notably "alpha-beta search" can help reduce the portion of the tree that needs to be examined, but not enough to solve this difficulty. For this reason, actual chess programs instead only search a much smaller depth (such as up to 7 moves), at which point they don't have enough information to evaluate the true consequences of the moves and are forced

KAITHANG

to guess by using heuristic "evaluation functions" that measure simple quantities such as the total number of pieces left.

Example 4: Knots.

If I give you a three-dimensional polygon (e.g. as a sequence of vertex coordinate triples), is there some way of twisting and bending the polygon around until it becomes flat? Or is it knotted?

There is an algorithm for solving this problem, which is very complicated and has not really been adequately analyzed. However it runs in at least exponential time.

One way of proving that certain polygons are not knots is to find a collection of triangles forming a surface with the polygon as its boundary. However this is not always possible (without adding exponentially many new vertices) and even when possible it's NP-complete to find these triangles.

There are also some heuristics based on finding a non-Euclidean geometry for the space outside of a knot that work very well for many knots, but are not known to work for all knots. So this is one of the rare examples of a problem that can often be solved efficiently in practice even though it is theoretically not known to be in P.

Certain related problems in higher dimensions (is this four-dimensional surface equivalent to a four-dimensional sphere) are provably undecidable.

Example 5: Halting problem.

Suppose you're working on a lab for a programming class, have written your program, and start to run it. After five minutes, it is still going. Does this mean it's in an infinite loop, or is it just slow?

It would be convenient if your compiler could tell you that your program has an infinite loop. However this is an undecidable problem: there is no program that will always correctly detect infinite loops.

Some people have used this idea as evidence that people are inherently smarter than computers, since it shows that there are problems computers can't solve. However it's not clear to me that people can solve them either. Here's an example:

```
main()
{
    int x = 3;
    for (;;) {
        for (int a = 1; a <= x; a++)
            for (int b = 1; b <= x; b++)
                for (int c = 1; c <= x; c++)
                    for (int i = 3; i <= x; i++)
                        if(pow(a,i) + pow(b,i) == pow(c,i))
                            exit;
        x++;
    }
}
```

This program searches for solutions to Fermat's last theorem. Does it halt? (You can assume I'm using a multiple-precision integer package instead of built in integers, so don't worry about arithmetic overflow complications.) To be able to answer this, you have to understand the recent

KAITHANG

proof of Fermat's last theorem. There are many similar problems for which no proof is known, so we are clueless whether the corresponding programs halt.

Problems of complexity theory

The most famous open problem in theoretical science is whether $P = NP$. In other words, if it's always easy to check a solution, should it also be easy to find the solution?

We have no reason to believe it should be true, so the expectation among most theoreticians is that it's false. But we also don't have a proof...

So we have this nice construction of complexity classes P and NP but we can't even say that there's one problem in NP and not in P . So what good is the theory if it can't tell us how hard any particular problem is to solve?

NP-completeness

The theory of NP-completeness is a solution to the practical problem of applying complexity theory to individual problems. NP-complete problems are defined in a precise sense as the hardest problems in P . Even though we don't know whether there is any problem in NP that is not in P , we can point to an NP-complete problem and say that if there are any hard problems in NP , that problem is one of the hard ones.

(Conversely if everything in NP is easy, those problems are easy. So NP-completeness can be thought of as a way of making the big $P=NP$ question equivalent to smaller questions about the hardness of individual problems.)

So if we believe that P and NP are unequal, and we prove that some problem is NP-complete, we should believe that it doesn't have a fast algorithm.

For unknown reasons, most problems we've looked at in NP turn out either to be in P or NP-complete. So the theory of NP-completeness turns out to be a good way of showing that a problem is likely to be hard, because it applies to a lot of problems. But there are problems that are in NP , not known to be in P , and not likely to be NP-complete; for instance the code-breaking example I gave earlier.

Reduction

Formally, NP-completeness is defined in terms of "reduction" which is just a complicated way of saying one problem is easier than another.

We say that A is easier than B , and write $A < B$, if we can write down an algorithm for solving A that uses a small number of calls to a subroutine for B (with everything outside the subroutine calls being fast, polynomial time). There are several minor variations of this definition depending on the detailed meaning of "small" -- it may be a polynomial number of calls, a fixed constant number, or just one call.

Then if $A < B$, and B is in P , so is A : we can write down a polynomial algorithm for A by expanding the subroutine calls to use the fast algorithm for B .

KAITHANG

So "easier" in this context means that if one problem can be solved in polynomial time, so can the other. It is possible for the algorithms for A to be slower than those for B, even though $A < B$.

As an example, consider the Hamiltonian cycle problem. Does a given graph have a cycle visiting each vertex exactly once? Here's a solution, using longest path as a subroutine:

```
for each edge (u,v) of G
if there is a simple path of length n-1 from u to v
    return yes // path + edge form a cycle
return no
```

This algorithm makes m calls to a longest path subroutine, and does $O(m)$ work outside those subroutine calls, so it shows that Hamiltonian cycle $<$ longest path. (It doesn't show that Hamiltonian cycle is in P, because we don't know how to solve the longest path subproblems quickly.)

As a second example, consider a polynomial time problem such as the minimum spanning tree. Then for every other problem B, $B <$ minimum spanning tree, since there is a fast algorithm for minimum spanning trees using a subroutine for B. (We don't actually have to call the subroutine, or we can call it and ignore its results.)

Cook's Theorem

We are now ready to formally define NP-completeness. We say that a problem A in NP is NP-complete when, for every other problem B in NP, $B < A$.

This seems like a very strong definition. After all, the notion of reduction we've defined above seems to imply that if $B < A$, then the two problems are very closely related; for instance Hamiltonian cycle and longest path are both about finding very similar structures in graphs. Why should there be a problem that closely related to all the different problems in NP?

Theorem: an NP-complete problem exists.

We prove this by example. One NP-complete problem can be found by modifying the halting problem (which without modification is undecidable).

Bounded halting. This problem takes as input a program X and a number K. The problem is to find data which, when given as input to X, causes it to stop in at most K steps.

To be precise, this needs some more careful definition: what language is X written in? What constitutes a single step? Also for technical reasons K should be specified in *unary* notation, so that the length of that part of the input is K itself rather than $O(\log K)$.

For reasonable ways of filling in the details, this is in NP: to test if data is a correct solution, just simulate the program for K steps. This takes time polynomial in K and in the length of program. (Here's one point at which we need to be careful: the program can not perform unreasonable operations such as arithmetic on very large integers, because then we wouldn't be able to simulate it quickly enough.)

To finish the proof that this is NP-complete, we need to show that it's harder than anything else in NP. Suppose we have a problem A in NP. This means that we can write a program PA that tests solutions to A, and halts within polynomial time p(n) with a yes or no answer depending on whether the given solution is really a solution to the given problem. We can then easily form a modified program PA' to enter an infinite loop whenever it would halt with a no answer. If we could solve bounded halting, we could solve A by passing PA' and p(n) as arguments to a

KAITHANG

subroutine for bounded halting. So $A <$ bounded halting. But this argument works for every problem in NP, so bounded halting is NP-complete.

How to prove NP-completeness in practice

The proof above of NP-completeness for bounded halting is great for the theory of NP-completeness, but doesn't help us understand other more abstract problems such as the Hamiltonian cycle problem.

Most proofs of NP-completeness don't look like the one above; it would be too difficult to prove anything else that way. Instead, they are based on the observation that if $A < B$ and $B < C$, then $A < C$. (Recall that these relations are defined in terms of the existence of an algorithm that calls subroutines. Given an algorithm that solves A with a subroutine for B, and an algorithm that solves B with a subroutine for C, we can just use the second algorithm to expand the subroutine calls of the first algorithm, and get an algorithm that solves A with a subroutine for C.)

As a consequence of this observation, if A is NP-complete, B is in NP, and $A < B$, B is NP-complete. In practice that's how we prove NP-completeness: We start with one specific problem that we prove NP-complete, and we then prove that it's easier than lots of others which must therefore also be NP-complete.

So e.g. since Hamiltonian cycle is known to be NP-complete, and Hamiltonian cycle $<$ longest path, we can deduce that longest path is also NP-complete.

Starting from the bounded halting problem we can show that it's reducible to a problem of simulating circuits (we know that computers can be built out of circuits, so any problem involving simulating computers can be translated to one about simulating circuits). So various circuit simulation problems are NP-complete, in particular Satisfiability, which asks whether there is an input to a Boolean circuit that causes its output to be one.

Circuits look a lot like graphs, so from there it's another easy step to proving that many graph problems are NP-complete. Most of these proofs rely on constructing *gadgets*, small subgraphs that act (in the context of the graph problem under consideration) like Boolean gates and other components of circuits.

Reg. No._____

Name:_____

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**SIXTH SEMESTER B.TECH MODEL EXAMINATION, APRIL 2018****Course Code: CS304****Course Name: COMPILER DESIGN****Max. Marks: 100****Duration: 3 Hours****PART A***Answer all questions, each carries 3 marks.*

1. Briefly explain the role of a lexical analyzer.
2. What are regular expressions? Give examples.
3. Explain any two error recovery strategies in a parser.
4. What is an ambiguous grammar? Give an example.

PART B*Answer any two full questions, each carries 9 marks*

5. Explain the different phases of a compiler. (9)
6. a) What is the role of transition diagrams in the construction of a lexical analyzer?(4.5)
b) Show that the following grammar is not LL(1) . (4.5)

$$\begin{aligned} S &\rightarrow iEtSS' \mid a \\ S' &\rightarrow eS \mid \epsilon \end{aligned}$$

$$E \rightarrow b$$

7. a) Explain the working of Nonrecursive predictive parser. (5)
b) Find FIRST and FOLLOW for the given grammar (4)

$$D \rightarrow \text{type list ;}$$

$$\text{list} \rightarrow \text{id tlist ;}$$

$$\text{tlist} \rightarrow , \text{id tlist} \mid \epsilon$$

$$\text{type} \rightarrow \text{int} \mid \text{float}$$

PART C*Answer all questions, each carries 3 marks.*

8. What do you mean by configuration of an LR Parser?
9. What are the different types of conflicts that may occur in a shift reduce parser?

10. What is an L-attributed definition?
11. Draw a DAG for the expression $a+a^*(b-c)+(b-c)^*d$

PART D

Answer any two full questions, each carries 9 marks.

12. Explain operator precedence parsing with example (9)

13. Construct the SLR parsing table for the given grammar (9)

$$\begin{aligned} S &\rightarrow AA \\ A &\rightarrow aA \mid b \end{aligned}$$

14. a) What is the difference between synthesized attributes and inherited attributes? (5)
b) Explain bottom up evaluation of S- attributed definition. (4)

PART E

Answer any four full questions, each carries 10 marks.

15. Explain the different intermediate representations. (10)
16. Explain the different storage allocation strategies. (10)
17. Explain the different methods of translating Boolean expressions. (10)
18. Explain the code generation algorithm. (10)
19. Explain the principal sources of optimization. (10)
20. Explain the issues in the design of a code generator (10)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

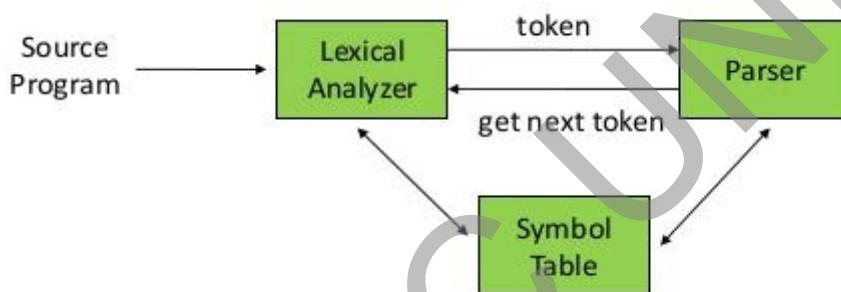
SIXTH SEMESTER B.TECH MODEL EXAMINATION, APRIL 2018

Course Code: CS304
Course Name: COMPILER DESIGN

ANSWER KEY

1)

- Task is to read the input characters and produce a sequence of tokens
- Gives output to parser for syntax analysis



- Performs other secondary tasks also
 - Removal of blank spaces
 - Correlating error messages
 - Can be divided into two phases
 - Scanning- performing simple jobs
 - Lexical analysis- performing complex jobs
- (Any 3 points- 3 marks)

2) A Regular Expression can be recursively defined as follows –

- ϵ is a Regular Expression indicates the language containing an empty string. ($L(\epsilon) = \{\epsilon\}$)
- ϕ is a Regular Expression denoting an empty language. ($L(\phi) = \{\}$)
- x is a Regular Expression where $L = \{x\}$
- If X is a Regular Expression denoting the language $L(X)$ and Y is a Regular Expression denoting the language $L(Y)$, then
 - $X + Y$ is a Regular Expression corresponding to the language $L(X) \cup L(Y)$ where $L(X+Y) = L(X) \cup L(Y)$.
 - $X . Y$ is a Regular Expression corresponding to the language $L(X) . L(Y)$ where $L(X.Y) = L(X) . L(Y)$

- R^* is a Regular Expression corresponding to the language $L(R^*)$ where $L(R^*) = (L(R))^*$

Eg: $(a+b)^*$: Set of strings of a's and b's of any length including the null string. So $L = \{ \epsilon, a, b, aa, ab, bb, ba, aaa, \dots \}$

3) Panic mode

When a parser encounters an error anywhere in the statement, it ignores the rest of the statement by not processing input from erroneous input to delimiter, such as semi-colon. This is the easiest way of error-recovery and also, it prevents the parser from developing infinite loops.

Statement mode

When a parser encounters an error, it tries to take corrective measures so that the rest of inputs of statement allow the parser to parse ahead. For example, inserting a missing semicolon, replacing comma with a semicolon etc. Parser designers have to be careful here because one wrong correction may lead to an infinite loop.

Error productions

Some common errors are known to the compiler designers that may occur in the code. In addition, the designers can create augmented grammar to be used, as productions that generate erroneous constructs when these errors are encountered.

Global correction

The parser considers the program in hand as a whole and tries to figure out what the program is intended to do and tries to find out a closest match for it, which is error-free. When an erroneous input (statement) X is fed, it creates a parse tree for some closest error-free statement Y. This may allow the parser to make minimal changes in the source code, but due to the complexity (time and space) of this strategy, it has not been implemented in practice yet.

(Any 2 strategies- 3marks)

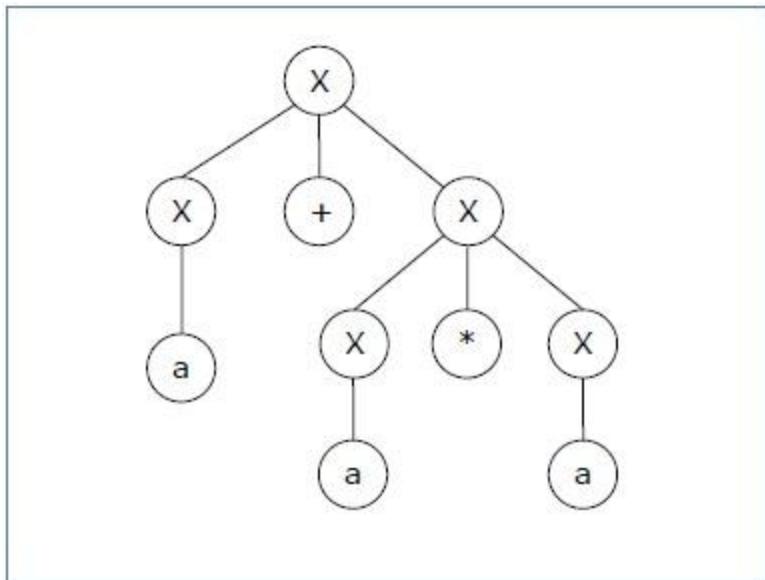
- 4) An **ambiguous grammar** is a context-free grammar for which there exists a string that can have more than one leftmost derivation or parse tree, while an **unambiguous grammar** is a context-free grammar for which every valid string has a unique leftmost derivation or parse tree.

Eg: $X \rightarrow X+X \mid X*X \mid X \mid a$

Let's find out the derivation tree for the string "a+a*a". It has two leftmost derivations.

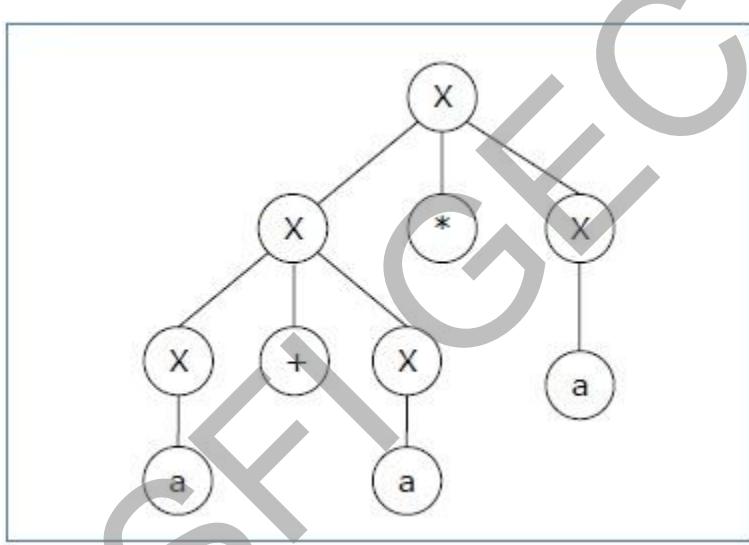
Derivation 1 – $X \rightarrow X+X \rightarrow a+X \rightarrow a+X*X \rightarrow a+a*X \rightarrow a+a*a$

Parse tree 1 –



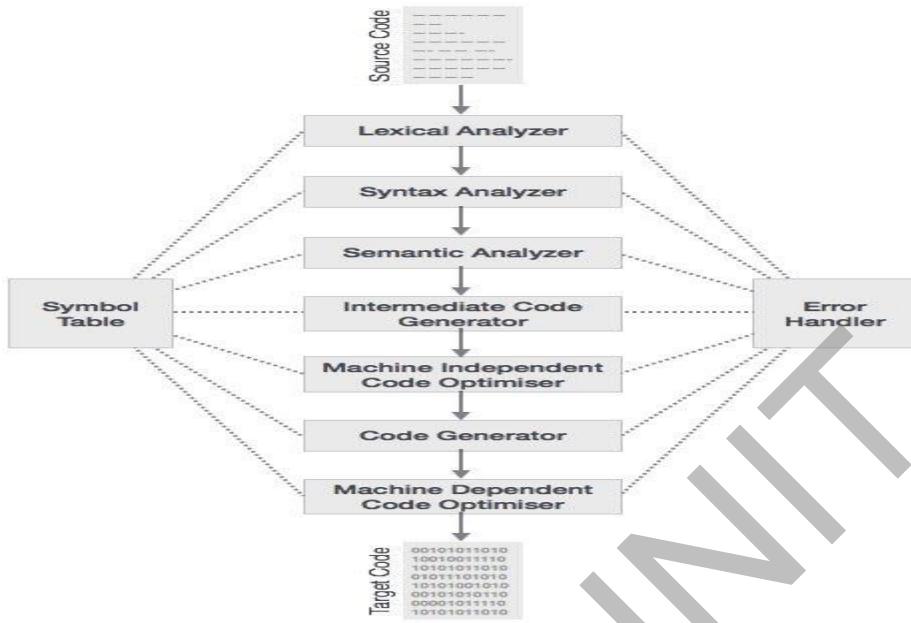
Derivation 2 – $X \rightarrow X*X \rightarrow X+X*X \rightarrow a+ X*X \rightarrow a+a*X \rightarrow a+a*a$

Parse tree 2 –



Since there are two parse trees for a single string "a+a*a", the grammar **G** is ambiguous.

- 5) The compilation process is a sequence of various phases. Each phase takes input from its previous stage, has its own representation of source program, and feeds its output to the next phase of the compiler.



Lexical Analysis

The first phase of scanner works as a text scanner. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. Lexical analyzer represents these lexemes in the form of tokens as:

<token-name, attribute-value>

Syntax Analysis

The next phase is called the syntax analysis or **parsing**. It takes the token produced by lexical analysis as input and generates a parse tree (or syntax tree). In this phase, token arrangements are checked against the source code grammar, i.e., the parser checks if the expression made by the tokens is syntactically correct.

Semantic Analysis

Semantic analysis checks whether the parse tree constructed follows the rules of language. For example, assignment of values is between compatible data types, and adding string to an integer. Also, the semantic analyzer keeps track of identifiers, their types and expressions; whether identifiers are declared before use or not, etc. The semantic analyzer produces an annotated syntax tree as an output.

Intermediate Code Generation

After semantic analysis, the compiler generates an intermediate code of the source code for the target machine. It represents a program for some abstract machine. It is in between the high-level language and the machine language. This intermediate code should be generated in such a way that it makes it easier to be translated into the target machine code.

Code Optimization

The next phase does code optimization of the intermediate code. Optimization can be assumed as something that removes unnecessary code lines, and arranges the sequence of statements in order to speed up the program execution without wasting resources (CPU, memory).

Code Generation

In this phase, the code generator takes the optimized representation of the intermediate code and maps it to the target machine language. The code generator translates the intermediate code into a sequence of

(generally) re-locatable machine code. Sequence of instructions of machine code performs the task as the intermediate code would do.

Symbol Table

It is a data-structure maintained throughout all the phases of a compiler. All the identifiers' names along with their types are stored here. The symbol table makes it easier for the compiler to quickly search the identifier record and retrieve it. The symbol table is also used for scope management.

6)

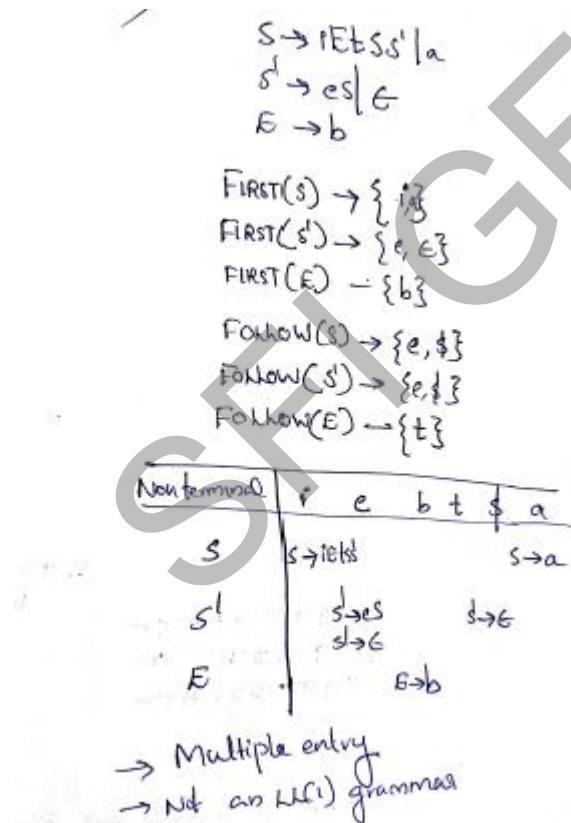
a) To specify a lexical analyzer we need a state machine, sometimes called Transition Diagram(TD), which is similar to a FSA. Transition Diagrams depict the actions that take place when the lexer is called by the parser to get the next token . FSA accepts or rejects a string. TD reads characters until finding a token, returns the read token and prepare the input buffer for the next call. In a TD, there is no out-transition from accepting states Transition labeled other (or not labeled) should be taken on any character except those labeling transitions out of a given state.

States can be marked with a *

: This indicates states on which a input retraction must take place.

To consider different kinds of lexeme, we usually build separate DFAs (or TD) corresponding to the regular expressions for each kind of lexeme then merge them into a single combined DFA (or TD).

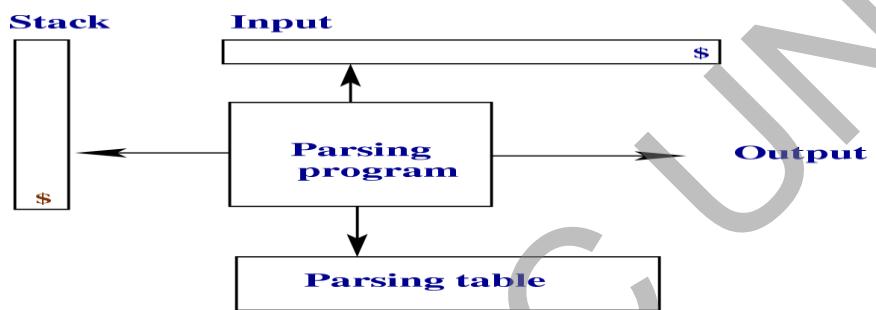
b)



7)

a) Predictive parsing can be performed using a pushdown stack, avoiding recursive calls.

- Initially the stack holds just the start symbol of the grammar.
- At each step a symbol X is popped from the stack:
 - if X is a terminal symbol then it is matched with *lookahead* and *lookahead* is advanced,
 - if X is a nonterminal, then using *lookahead* and a *parsing table* (implementing the FIRST sets) a production is chosen and its right hand side is pushed onto the stack.
- This process goes on until the stack and the input string become empty. It is useful to have an *end_of_stack* and an *end_of_input* symbols. We denote them both by $\$$.



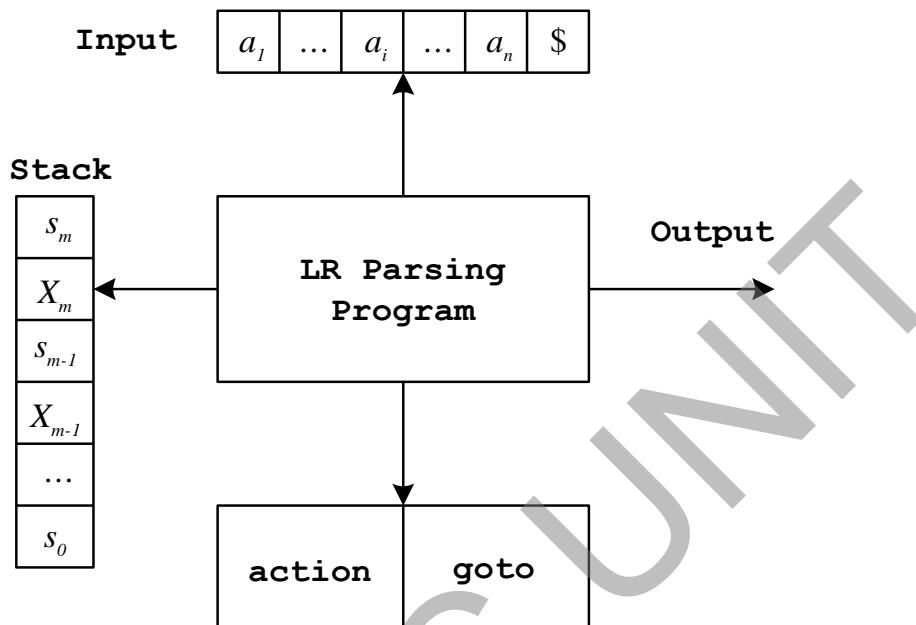
b)

$D \rightarrow \text{type. list}$
 $\text{list} \rightarrow \text{id. tlist};$
 $\text{tlist} \rightarrow , \text{id. tlist} | \epsilon$
 $\text{type} \rightarrow \text{int. float}$

$\text{FIRST}(D) - \{\text{int, float}\}$	$\text{Follow}(D) - \{\$\}$
$\text{FIRST}(\text{list}) - \{\text{id}\}$	$\text{Follow}(\text{list}) - \{\$\}$
$\text{FIRST}(\text{tlist}) - \{\, \text{id}\}$	$\text{Follow}(\text{tlist}) - \{\$\}$
$\text{FIRST}(\text{type}) - \{\text{int, float}\}$	$\text{Follow}(\text{type}) - \{\text{id}\}$

8) An *LR Parser* consists of an input, output, a stack, a driver program and a parsing table that contains actions and gotos. While the driver program is the same for all LR parsers, each parser uses a different parsing table derived from the particular grammar. The *parsing program* reads characters from

the input buffer and, and uses a *stack* to keep strings of *symbols* and *system states*. Each state summarizes the information contained in the stack below it, so that the combination of the state on the stack top and the current input symbol and used to index the parsing table to determine the shift-reduce parsing ecision.



A *configuration* of an LR parser is a pair whose first component is the stack contents and whose second component is the unexpected input:

($s_0 \ X_1 s_1 \ X_2 s_2 \dots X_m s_m, a_i \ a_{i+1} \dots a_n \ $)$

The configuration represents the right-sentential form:

$X_1 \ X_2 \dots X_m \ a_i \ a_{i+1} \dots a_n$

The possible configurations resulting after each type of move are:

1) If $action[s_m, a_i] = \text{shift } s$, the parser executes a shift move, entering the configuration:

($s_0 \ X_1 s_1 \ X_2 s_2 \dots X_m s_m \ a_i \ s, a_{i+1} \dots a_n \ $)$

2) If $action[s_m, a_i] = \text{reduce } A \rightarrow \beta$, then the prser executes a reduce move, entering the configuration:

$s_0 \ X_1 s_1 \ X_2 s_2 \dots X_{m-r} s_{m-r} A \ s, a_i \ a_{i+1} \dots a_n \ $)$

where $s = goto[s_{m-r}, A]$ and r is the length of β .

3) If $action[s_m, a_i] = \text{accept}$, parsing is completed

4) If $\text{action}[s_m, a_i] = \text{error}$, the parser has discovered an error and calls an error recovery routine.

9) When a grammar is not carefully thought out, the parser generated from the grammar may face two kinds of dilemmas:

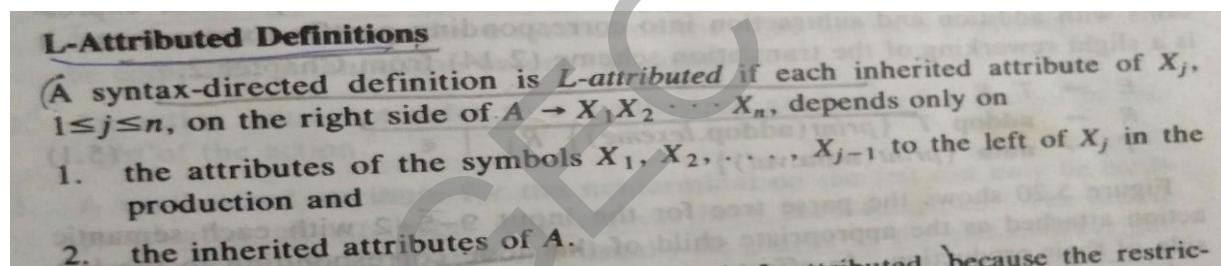
1. Shift/Reduce Conflict:

Enough terms have been read and a grammar rule can be recognized according to the accumulated terms. In this situation, the parser can make a reduction. If, however, there is also another grammar rule which calls for more terms to be accumulated and the look ahead token is just what the second grammar rule expected. In this situation, the parser may also make a shift operation. This dilemma faced by the parser is called the Shift/Reduce Conflict.

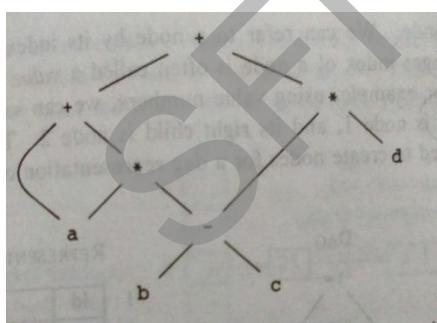
2. Reduce/Reduce Conflict:

Enough terms have been read and two grammar rules are recognized based on the accumulated terms. If the parser then decides to make a reduction, should it reduce the accumulated terms according to the first or second grammar rules? This type of difficulty faced by the parser is called the Reduce/Reduce Conflict.

10) L-attributed grammars are a special type of attribute grammars. They allow the attributes to be evaluated in one left-to-right traversal of the abstract syntax tree. As a result, attribute evaluation in L-attributed grammars can be incorporated conveniently in top-down parsing.



11)



12) Done on small class of grammars

- Operator grammar
- No production on the right side have ϵ
- no two adjacent non terminals

$$\begin{array}{l} E \rightarrow EAE \mid (E) \mid -E \mid \text{id} \\ A \rightarrow + \mid - \mid * \mid / \mid \uparrow \end{array}$$

- Not an operator grammar
 - By substituting A we get
- $$E \rightarrow E+E \mid E-E \mid E*E \mid E/E \mid E \uparrow E \mid (E) \mid -E \mid \text{id}$$
- Three disjoint precedence relations are defined between the pair of terminals

\prec , \doteq , and \succ

RELATION	MEANING
$a \prec b$	a "yields precedence to" b
$a \doteq b$	a "has the same precedence as" b
$a \succ b$	a "takes precedence over" b

(definition 5 marks+ example 4 marks)

(13)

Augmented Grammar

- ① $S' \rightarrow S$
- ② $S \rightarrow AA$
- ③ $A \rightarrow aA \mid b$

$I_0 : S' \rightarrow S$
 $S \rightarrow A \cdot A$
 $A \rightarrow aA$
 $A \rightarrow b$

$I_1 : \text{goto}(I_0, S)$
 $S' \rightarrow S \cdot$

$I_2 : \text{goto}(I_0, A)$
 $S \rightarrow A \cdot A$
 $A \rightarrow aA$
 $A \rightarrow b$

$I_3 : \text{goto}(I_0, a)$
 $A \rightarrow a \cdot A$
 $A \rightarrow \cdot aA$
 $A \rightarrow \cdot b$

$I_4 : \text{goto}(I_0, b)$
 $A \rightarrow b \cdot$

$I_5 : \text{goto}(I_2, A)$
 $S \rightarrow AA \cdot$

~~$I_6 : \text{goto}(I_2, a) \rightarrow I_3$~~
 $A \rightarrow a \cdot A$
 $A \rightarrow \cdot aA$
 $A \rightarrow \cdot b$

$\text{goto}(I_2, b) \rightarrow I_4$
 $I_6 = \text{goto}(I_3, A)$
 $A \rightarrow aA \cdot$

$\text{goto}(I_3, a) \rightarrow I_3$
 $\text{goto}(I_3, b) \rightarrow I_4$

Transition Diagram

Action

States	Action		Goto		
	a	b	\$	S	A
0	s_3	s_4		1	2
1			accept		
2	s_3	s_4		5	
3	s_3	s_4		6	
4	r_4	r_4	r_4		
5				r_2	
6	r_3	r_3	r_3		

$\text{Follow}(S) = \{\$\}$
 $\text{Follow}(A) = \{a, b, \$\}$

14)a) Synthesized attributes

- Synthesized attributes get values from the attribute values of their child nodes. To illustrate, assume the following production:
- $S \rightarrow ABC$
- If S is taking values from its child nodes (A, B, C), then it is said to be a synthesized attribute, as the values of ABC are synthesized to S .
- Synthesized attributes never take values from their parent nodes or any sibling nodes.

Inherited attributes

- In contrast to synthesized attributes, inherited attributes can take values from parent and/or siblings. As in the following production,
- $S \rightarrow ABC$
- A can get values from S, B and C. B can take values from S, A, and C. Likewise, C can take values from S, A, and B.

b)

- Synthesized Attributes can be evaluated by a bottom-up parser as the input is being analyzed avoiding the construction of a dependency graph.
- The parser keeps the values of the synthesized attributes in its stack.
- Whenever a reduction $A \rightarrow \alpha$ is made, the attribute for A is computed from the attributes of α which appear on the stack.
- Thus, a translator for an S-Attributed Definition can be simply implemented by extending the stack of an LR-Parser.
- Extra fields are added to the stack to hold the values of synthesized attributes.
- In the simple case of just one attribute per grammar symbol the stack has two fields: *state* and *val*.

<i>state</i>	<i>val</i>
Z	$Z.x$
Y	$Y.x$
X	$X.x$
...	...

- The current top of the stack is indicated by the pointer variable *top*.
- Synthesized attributes are computed just before each reduction:
 - Before the reduction $A \rightarrow XYZ$ is made, the attribute for A is computed
 - $A.a := f(val[top]; val[top - 1]; val[top - 2])$.
- 15) After syntax and semantic analysis, some compilers generate an explicit intermediate representation of the source program.
- This IR should have two important properties:
 - It should be easy to produce
 - it should be easy to translate into target program.
- IR is an intermediate stage of the mapping from source level abstractions to target machine abstractions.
- Kinds of Intermediate representations
 - Syntax trees
 - Postfix notation
 - Three address code
- Three-address code is a sequence of statements of the general form $x := y op z$
- Where x, y and z are names, constants, or compiler generated temporaries.

- op stands for any operator, such as fixed- or floating-point arithmetic operator, or a logical operator on Boolean-valued data.
- Thus a source language expression like $x + y * z$ might be translated into a sequence
- $t1 := y * z$
- $t2 := x + t1$
- where $t1$ and $t2$ are compiler-generated temporary names.
- Here is a list of the common three-address instruction forms:
 - Assignment instructions of the form $x = y \text{ op } z$, where op is a binary arithmetic or logical operation, and x, y, and z are addresses.
 - Assignments of the form $x = \text{op } y$, where op is a unary operation. Essential unary operations include unary minus, logical negation, shift operators etc.
 - Copy instructions of the form $x = y$, where x is assigned the value of y.
 - An unconditional jump goto L. The three-address instruction with label L is the next to be executed.
- 1. Conditional jumps such as if $x \text{ relop } y \text{ goto } L$, which apply a relational operator $<$, $=$, $>$, etc. to x and y, and execute the instruction with label L next if x stands in relation relop to y. If not, the three-address instruction following if $x \text{ relop } y \text{ goto } L$ is executed next, in sequence.
- 2. Procedure calls and returns are implemented using the following instructions: param x for parameters; call p, n and $y = \text{call } p, n$ for procedure and function calls, respectively; and return y, where y, representing a returned value, is optional. Their typical use is as the sequence of three
- address instructions
- param X1
- param X2
- param xn
- call p, n

generated as part of a call of the procedure $p(X1, X2, \dots, xn)$.

Indexed copy instructions of the form $x = y[i]$ and $x[i] = y$. The instruction $x = y[i]$ sets x to the value in the location i memory units beyond location y. The instruction $x[i] = y$ sets the contents of the location I units beyond x to the value of y.

Address and pointer assignments of the form $x = \&y$, $x = *y$, and $*x = y$

16) The different storage allocation strategies are :

1. Static allocation - lays out storage for all data objects at compile time
2. Stack allocation - manages the run-time storage as a stack.

3. Heap allocation - allocates and deallocates storage as needed at run time from a data area known as heap. (explain each briefly)

17)

Methods of Translating Boolean Expressions:

There are two principal methods of representing the value of a boolean expression. They are :

- * To encode true and false numerically and to evaluate a boolean expression analogously to an arithmetic expression. Often, 1 is used to denote true and 0 to denote false.
- * To implement boolean expressions by flow of control, that is, representing the value of a boolean expression by a position reached in a program. This method is particularly convenient in implementing the boolean expressions in flow-of-control statements, such as the if-then and while-do statements.

Numerical Representation

Here, 1 denotes true and 0 denotes false. Expressions will be evaluated completely from left to right, in a manner similar to arithmetic expressions.

For example :

- * The translation for a or b and not c is the three-address sequence
- * t1 := not c
- t2 := b and t1
- t3 := a or t2
- * A relational expression such as a < b is equivalent to the conditional statement
- * if a < b then 1 else 0

which can be translated into the three-address code sequence (aga statement numbers at 100) :

100 : if a < b goto 103 101 : t := 0

102 : goto 104

103 : t := 1

104 :

Short-Circuit Code:

We can also translate a boolean expression into three-address code without generating code for any of the boolean operators and without having the code necessarily evaluate the entire expression. This style of evaluation is sometimes called “short-circuit” or “jumping” code. It is possible to evaluate boolean expressions without generating code for the boolean operators and, or, and not if we represent the value of an expression by a position in the code sequence.

Translation of $a < b$ or $c < d$ and $e < f$

100 : if $a < b$ goto

103 101 : $t1 := 0$

102 : goto 104 103 : $t1 := 1$

104 : if $c < d$ goto

107 105 : $t2 := 0$

106 : goto 108

107 : $t2 := 1$

108 : if $e < f$ goto 111 109 : $t3 := 0$

110 : goto 112

111 : $t3 := 1$

112 : $t4 := t2 \text{ and } t3$

113 : $t5 := t1 \text{ or } t4$

18)

- Consider an instruction of the form “ $x := y \text{ op } z$ ”
- Invoke **getreg** to determine the location L where the result of “ $y \text{ op } z$ ” will be placed
- Determine a current location y' of y from the address descriptor (register location preferred). If y' is not L , generate “**MOV** y', L ”
- Generate “**op** z', L ”, where z' is a current location of z from the address descriptor.
- Update the address and register descriptors for x , y , z , and L

- Consider an instruction of the form “ $x := y$ ”
- If y is in a register, change the register and address descriptors
- If y is in memory,
 - if x has next use in the block, invoke **getreg** to find a register r , generate “**MOV** y, r ”, and make r the location of x
 - otherwise, generate “**MOV** y, x ”
- Once all statements in the basic block are processed, we store those names that are *live on exit* and *not in their memory locations*

19) A transformation of a program is called local if it can be performed by looking only at the statements in a basic block; otherwise, it is called global. Many transformations can be performed at both the local and global levels. Local transformations are usually performed first.

Function-Preserving Transformations

There are a number of ways in which a compiler can improve a program without changing the

function it computes.

Function preserving transformations examples:

Common sub expression elimination

Copy propagation,

(explain briefly)

Dead-code elimination

Constant folding

The other transformations come up primarily when global optimizations are performed.

Frequently, a program will include several calculations of the offset in an array. Some of the duplicate calculations cannot be avoided by the programmer because they lie below the level of detail accessible within the source language.

20) Issues arise during the code generation phase:

1. Input to code generator
2. Target program
3. Memory management
4. Instruction selection
5. Register allocation
6. Evaluation order

1. Input to code generator:

- The input to the code generation consists of the intermediate representation of the source program produced by front end , together with information in the symbol table to determine run-time addresses of the data objects denoted by the names in the intermediate representation.
- Intermediate representation can be :
 - a. Linear representation such as postfix notation
 - b. Three address representation such as quadruples
 - c. Virtual machine representation such as stack machine code
 - d. Graphical representations such as syntax trees and dags.
 - e. • Prior to code generation, the front end must be scanned, parsed and translated into intermediate representation along with necessary type checking. Therefore, input to code generation is assumed to

be error-free.

f. 2. Target program:

- The output of the code generator is the target program. The output may be :
 - a. Absolute machine language
 - It can be placed in a fixed memory location and can be executed immediately.
 - b. Relocatable machine language
 - It allows subprograms to be compiled separately.
- c. Assembly language
 - Code generation is made easier.

3. Memory management:

- Names in the source program are mapped to addresses of data objects in run-time memory by the front end and code generator.
- It makes use of symbol table, that is, a name in a three-address statement refers to a symbol-table entry for the name.
- Labels in three-address statements have to be converted to addresses of instructions. For example,

j:goto generates jump instruction as follows:

- * if $i < j$, a backward jump instruction with target address equal to location of code for quadruple i is generated.
- * if $i > j$, the jump is forward. We must store on a list for quadruple i the location of the first machine instruction generated for quadruple j. When i is processed, the machine locations for all instructions that forward jumps to i are filled.

4. Instruction selection:

- The instructions of target machine should be complete and uniform.
- Instruction speeds and machine idioms are important factors when efficiency of target program is considered.
- The quality of the generated code is determined by its speed and size.
- The former statement can be translated into the latter statement as shown below:

$$a := b + c$$

```
d:=a+e (a)
MOV b,R0
ADD c,R0
MOV R0,a (b)
MOV a,R0
ADD e,R0
MOV R0,d
```

5. Register allocation

- Instructions involving register operands are shorter and faster than those involving operands in memory.

The use of registers is subdivided into two subproblems :

1. Register allocation - the set of variables that will reside in registers at a point in the program is selected.
2. Register assignment - the specific register that a value picked.
3. Certain machine requires even-odd register pairs for some operands and results. For example , consider the division instruction of the form :D x, y where, x - dividend even register in even/odd register pair y-divisor

even register holds the remainder

odd register holds the quotient

6. Evaluation order

- The order in which the computations are performed can affect the efficiency of the target code.

Some computation orders require fewer registers to hold intermediate results than others.

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
SIXTH SEMESTER B.TECH DEGREE MODEL EXAMINATION, MARCH 2018
Department: Computer Science and Engineering

Subject: - CS304: COMPILER DESIGN

Time: 3 hours

Max. Marks: 100

PART A
Answer all questions

1. What are the tools used for compiler construction? (3)
2. What is a symbol table? (3)
3. Differentiate regular expression and regular definition with an example. (3)
4. Differentiate tokens, patterns, lexeme (3)

Total: (12)

PART B
Answer any two full questions

5. With a neat diagram, explain the different phases of a compiler. Mention the input and output of each phase. (9)
6. Explain Input buffering with example. (9)
7. Explain the construction of Predictive parsing table for the grammar.

E->TE'

E'->+TE'|e

T->FT'

T'->*FT' | e

F->(E) | id

(9)

Total: (18)

PART C
Answer all questions

8. What is an operator precedence parser? (3)
9. Define viable prefix, kernel & non-kernel items. (3)
10. Give the syntax-directed definition for if-else statement (3)
11. Explain synthesized and inherited attributes. (3)

Total: (12)

PART D

Answer any *two* full questions

12. Construct LR(0) items of following grammar

$$S \rightarrow L=R$$

$$S \rightarrow R$$

$$L \rightarrow *R$$

$$L \rightarrow ID$$

$$R \rightarrow L$$

13. Construct CLR parsing table for the following grammar

$$S \rightarrow CC$$

$$C \rightarrow cC \mid d$$

14. Explain bottom up evaluation of S attributed definitions.

(9)
(9)

(9)

Total: (18)

PART E

Answer any four full questions

15. Explain Storage- allocation strategies.

(10)

16. a. Explain implementations of three address statements.

(5)

- b. Draw syntax tree and DAG for statement $a:=b^*-c+b^*-c$

(5)

17. Explain Principal sources of code optimization

(10)

18. Explain the issues in the design of code generator.

(10)

Total: (40)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
SIXTH SEMESTER B.TECH DEGREE MODEL EXAMINATION, MARCH 2018
Department: Computer Science and Engineering
Subject: - CS304: COMPILER DESIGN
ANSWER KEY

Time: 3 hours

Max. Marks: 100

PART A
Answer *all* questions

1. a)Parser Generators

Input: Grammatical description of a programming language

Output: Syntax analyzers.

Parser generator takes the grammatical description of a programming language and produces a syntax analyzer.

b)Scanner Generators

Input: Regular expression description of the tokens of a language

Output: Lexical analyzers.

Scanner generator generates lexical analyzers from a regular expression description of the tokens of a language.

c)Syntax-directed Translation Engines

Input: Parse tree.

Output: Intermediate code.

Syntax-directed translation engines produce collections of routines that walk a parse tree and generates intermediate code.

d)Automatic Code Generators

Input: Intermediate language.

Output: Machine language.

Code-generator takes a collection of rules that define the translation of each operation of the intermediate language into the machine language for a target machine.

e)Data-flow Analysis Engines

Data-flow analysis engine gathers the **information**, that is, the values transmitted from one part of a program to each of the other parts. Data-flow analysis is a key part of code optimization. (3)

- 2.** A symbol table is a data structure containing a record for each identifier, with fields for the attributes of the identifier. The data structure allows us to find the record for each identifier quickly and to store or retrieve data from that record quickly. Whenever an identifier is detected by a lexical analyzer, it is entered into the symbol table. The attributes of an identifier cannot be determined by the lexical analyzer.

3. Regular expression: It is *representator* of regular language. Regular expression is mathematically represent by some expression called regular expression. Regular expression is character sequence that define a search pattern.

Eg: Regular expression for identifier is letter(letter/digit)*

Regular definition: is the name given to regular expression

eg:Regular definition is id---->letter(letter/digit)*

(3)

4. Tokens- Sequence of characters that have a collective meaning.

Patterns- There is a set of strings in the input for which the same token is produced as output. This set of strings is described by a rule called a pattern associated with the token

Lexeme- A sequence of characters in the source program that is matched by the pattern for a token.

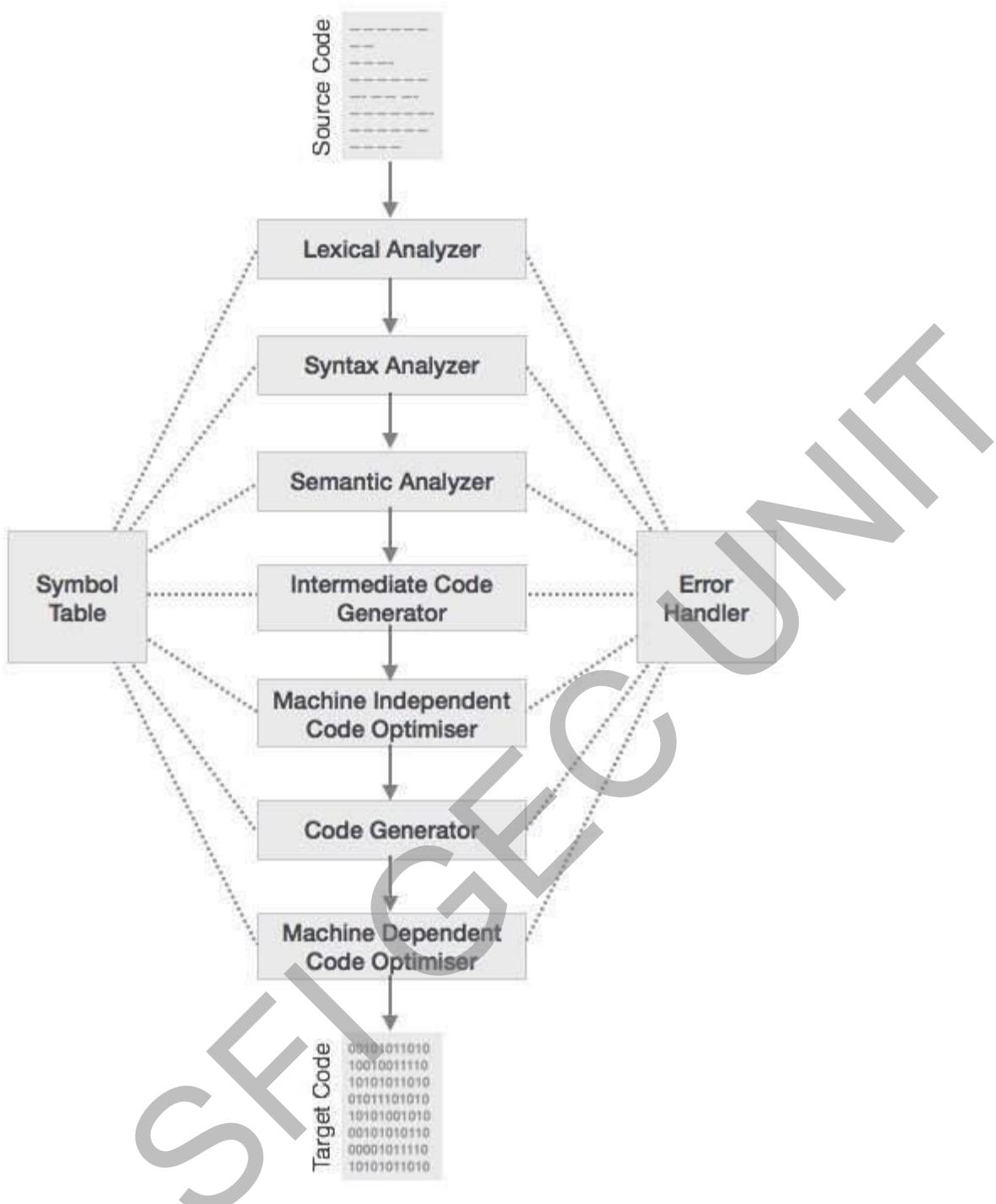
(3)

Total: (12)

PART B
Answer any two full questions

5.

The compilation process is a sequence of various phases. Each phase takes input from its previous stage, has its own representation of source program, and feeds its output to the next phase of the compiler.



Lexical Analysis

The first phase of scanner works as a text scanner. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. Lexical analyzer represents these lexemes in the form of tokens as:

<token-name, attribute-value>

Syntax Analysis

The next phase is called the syntax analysis or parsing. It takes the token produced by lexical analysis as input and generates a parse tree (or syntax tree). In this phase, token arrangements are checked against the source code grammar, i.e. the parser checks if the expression made by the tokens is syntactically correct.

Semantic Analysis

Semantic analysis checks whether the parse tree constructed follows the rules of language. For example, assignment of values is between compatible data types, and adding string to an integer. Also, the semantic analyzer keeps track of identifiers, their types and expressions; whether identifiers are declared before use or not etc. The semantic analyzer produces an annotated syntax tree as an output.

Intermediate Code Generation

After semantic analysis the compiler generates an intermediate code of the source code for the target machine. It represents a program for some abstract machine. It is in between the high-level language and the machine language. This intermediate code should be generated in such a way that it makes it easier to be translated into the target machine code.

Code Optimization

The next phase does code optimization of the intermediate code. Optimization can be assumed as something that removes unnecessary code lines, and arranges the sequence of statements in order to speed up the program execution without wasting resources (CPU, memory).

Code Generation

In this phase, the code generator takes the optimized representation of the intermediate code and maps it to the target machine language. The code generator translates the intermediate code into a sequence of (generally) re-locatable machine code. Sequence of instructions of machine code performs the task as the intermediate code would do.

Symbol Table

It is a data-structure maintained throughout all the phases of a compiler. All the identifier's names along with their types are stored here. The symbol table makes it easier for the compiler to quickly search the identifier record and retrieve it. The symbol table is also used for scope management.

(9)

6. Explain Input buffering with example.

To ensure that a right lexeme is found, one or more characters have to be looked up beyond the next lexeme.

- Hence a two-buffer scheme is introduced to handle large lookaheads safely.
- Techniques for speeding up the process of lexical analyzer such as the use of sentinels to mark the buffer end have been adopted.

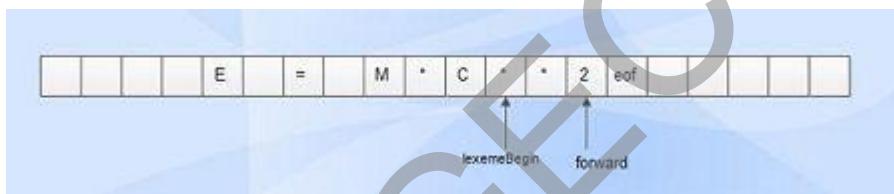
There are three general approaches for the implementation of a lexical analyzer:

- (i) By using a lexical-analyzer generator, such as lex compiler to produce the lexical analyzer from a regular expression based specification. In this, the generator provides routines for reading and buffering the input.
- (ii) By writing the lexical analyzer in a conventional systems-programming language, using I/O facilities of that language to read the input.
- (iii) By writing the lexical analyzer in assembly language and explicitly managing the reading of input.

Buffer Pairs

Because of large amount of time consumption in moving characters, specialized buffering techniques have been developed to reduce the amount of overhead required to process an input character.

Fig shows the buffer pairs which are used to hold the input data.



Scheme

- Consists of two buffers, each consists of N-character size which are reloaded alternatively.
- N-Number of characters on one disk block, e.g., 4096.
- N characters are read from the input file to the buffer using one system read command.
- *eof* is inserted at the end if the number of characters is less than N.

Pointers

Two pointers *lexemeBegin* and *forward* are maintained.

lexeme Begin points to the beginning of the current lexeme which is yet to be found.

forward scans ahead until a match for a pattern is found.

- Once a lexeme is found, *lexemebegin* is set to the character immediately after the lexeme which is just found and *forward* is set to the character at its right end.
- Current lexeme is the set of characters between two pointers.

Disadvantages of this scheme

- This scheme works well most of the time, but the amount of lookahead is limited.

- This limited lookahead may make it impossible to recognize tokens in situations where the distance that the forward pointer must travel is more than the length of the buffer.

(eg.) **DECLARE (ARG1, ARG2, . . . , ARGn)** in PL/1 program;

- It cannot determine whether the **DECLARE** is a keyword or an array name until the character that follows the right parenthesis.

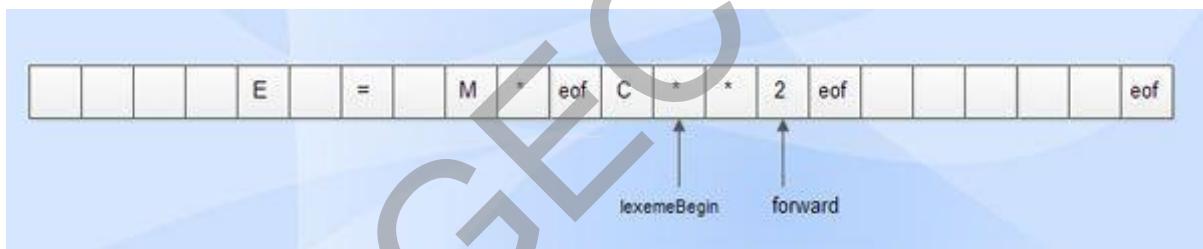
Sentinels

- In the previous scheme, each time when the forward pointer is moved, a check is done to ensure that one half of the buffer has not moved off. If it is done, then the other half must be reloaded.
- Therefore the ends of the buffer halves require two tests for each advance of the forward pointer.

Test 1: For end of buffer.

Test 2: To determine what character is read.

- The usage of sentinel reduces the two tests to one by extending each buffer half to hold a sentinel character at the end.
- The sentinel is a special character that cannot be part of the source program. (*eof* character is used as sentinel).



Advantages

- Most of the time, It performs only one test to see whether forward pointer points to an *eof*.
- Only when it reaches the end of the buffer half or *eof*, it performs more tests.
- Since N input characters are encountered between *eof*s, the average number of tests per input character is very close to 1.

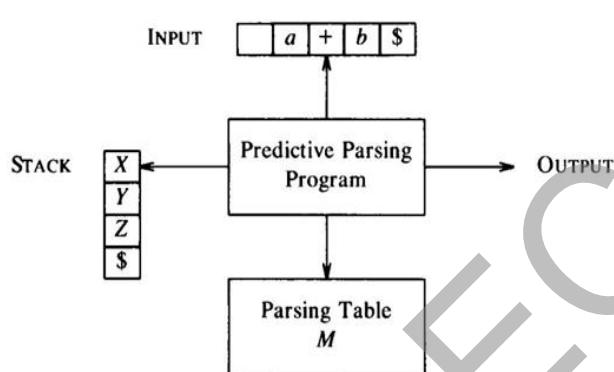
(9)

7.

Predictive parser can be implemented by recursive-descent parsing (may need to manipulate the grammar, e.g eliminating left recursion and left factoring).

Requirement: by looking at the first terminal symbol that a nonterminal symbol can derive, we should be able to choose the right production to expand the nonterminal symbol.

If the requirement is met, the parser easily be implemented using a non-recursive scheme by building a parsing table.



(9)

Fig. 4.13. Model of a nonrecursive predictive parser.

- **First(a)** - the set of tokens that can appear as the first symbols of one or more strings generated from a. If a is empty string or can generate empty string, then empty string is also in First(a).
- Predictive parsing won't work on some type of grammars:
 - Left recursion: $A \rightarrow A\omega$
 - Have common left factor: $A \rightarrow aB \mid aC$

Predictive parsers can be constructed for a class of grammars called LL(1).

L->Left

L->Leftmost derivation

1->One input symbol at each step

No left recursive or ambiguous grammar can be LL(1)

First(a): Here, a is a string of symbols. The set of terminals that begin strings derived from a. If a is empty string or generates empty string, then empty string is in First(a).

Follow(A): Here, A is a nonterminal symbol. Follow(A) is the set of terminals that can immediately follow A in a sentential form

E->TE'

$$\text{First}(E) = \{(, \text{id}\}, \text{Follow}(E) = \{\}, \$\}$$

$E' \rightarrow +TE' e$	First(E') = {+, e}, Follow(E') = {}, \$}
$T \rightarrow FT'$	First(T) = {(, id}, Follow(T) = {+,), \$}
$T' \rightarrow *FT' e$	First(T') = {*}, Follow(T') = {+,), \$}
$F \rightarrow (E) id$	First(F) = {(, id}, Follow(F) = {*}, {+,), \$}

Algorithm for construction of parsing table

INPUT :- Grammar G

OUTPUT:- Parsing table M

For each production $A \rightarrow \alpha$, do the following :For each terminal 'a' in FIRST(A), add $A \rightarrow \alpha$ to $M[A,a]$.If ϵ is in FIRST(α) then for each terminal b in FOLLOW(A) add $A \rightarrow \alpha$ to $M[A,b]$. If b is \$ then also add $A \rightarrow \alpha$ to $M[A,\$]$.If there is no production in $M[A,a]$, then set $M[A,a]$ to error.

Non Terminal	INPUT SYMBOLS						
	id	+	*	()		\$
E	$E \rightarrow TE'$					$E \rightarrow TE'$	
E'		$E \rightarrow +TE'$				$E' \rightarrow \epsilon$	$E' \rightarrow$
T	$T \rightarrow FT'$					$T \rightarrow FT'$	
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$			$T' \rightarrow \epsilon$	$T' \rightarrow$
F	$F \rightarrow id$					$F \rightarrow (E)$	

Step	Stack	Input	Next Action
1	\$E	id*id\$	$E \rightarrow TE'$
2	\$E'T	id*id\$	$T \rightarrow FT'$
3	\$E'T'F	id*id\$	$F \rightarrow id$
4	\$E'T'id	id*id\$	match id

5	\$E'T'	*id\$	T'→*FT'
6	\$T'F*	*id\$	match *
7	\$T'F	id\$	F→id
8	\$T'id	id\$	match id
9	\$T'	\$	T'→ε
10	\$	\$	accept

(9)

Non-terminal symbols

8. A grammar is said to be operator precedence if it possess the following properties:
1. No production on the right side is ϵ .
 2. There should not be any production rule possessing two adjacent non terminals at the right hand side.
 9. The set of prefixes of right sentential forms that can appear on the stack of a shift-reduce parser are called viable prefixes.

(3)

Kernel items, whish include the initial item, $S' \rightarrow .S$, and all items whose dots are not at the left end.

Non-kernel items, which have their dots at the left end.

(3)

10.

1. $S \rightarrow \text{if } E \text{ then } S_1$

E.true := new_label()

E.false := S.next

S1.next := S.next

S.code := E.code || gen_code(E.true ‘:’) || S1.code

2. $S \rightarrow \text{if } E \text{ then } S_1 \text{ else } S_2$

E.true := new_label()

E.false := new_label()

S1.next := S.next

S2.next := S.next

S.code := E.code || gen_code(E.true ':') || S1.code | gen_code('go to', S.next) | gen_code(E.false ':') ||

S2.code

(3)

11.

Synthesized attributes

These attributes get values from the attribute values of their child nodes. To illustrate, assume the following production:

$S \rightarrow ABC$

If S is taking values from its child nodes (A,B,C), then it is said to be a synthesized attribute, as the values of ABC are synthesized to S.

As in our previous example ($E \rightarrow E + T$), the parent node E gets its value from its child node. Synthesized attributes never take values from their parent nodes or any sibling nodes.

Inherited attributes

In contrast to synthesized attributes, inherited attributes can take values from parent and/or siblings. As in the following production,

$S \rightarrow ABC$

A can get values from S, B and C. B can take values from S, A, and C. Likewise, C can take values from S, A, and B.

(3)

PART D Answer any two full questions

12. LR Parsers

- + Can be constructed to recognize virtually all programming languages constructed from context-free grammars
- + Most general non-backtracking shift-reduce parsing method
- + Very fast at detecting syntactic errors - Too much work to construct LR parsing by hand

- Item: production rule with information about current parsing state

$A \rightarrow XYZ$ yields the four items

$$\begin{aligned} A &\rightarrow \cdot XYZ \\ A &\rightarrow X \cdot YZ \\ A &\rightarrow XY \cdot Z \\ A &\rightarrow XYZ \cdot \end{aligned}$$

$$\begin{aligned} I_0: \quad S' &\rightarrow \cdot S \\ S &\rightarrow \cdot L = R \\ S &\rightarrow \cdot R \\ L &\rightarrow \cdot * R \\ L &\rightarrow \cdot \text{id} \\ R &\rightarrow \cdot L \end{aligned}$$

$$I_1: \quad S' \rightarrow S \cdot$$

$$\begin{aligned} I_2: \quad S &\rightarrow L \cdot = R \\ R &\rightarrow L \cdot \end{aligned}$$

$$I_3: \quad S \rightarrow R \cdot$$

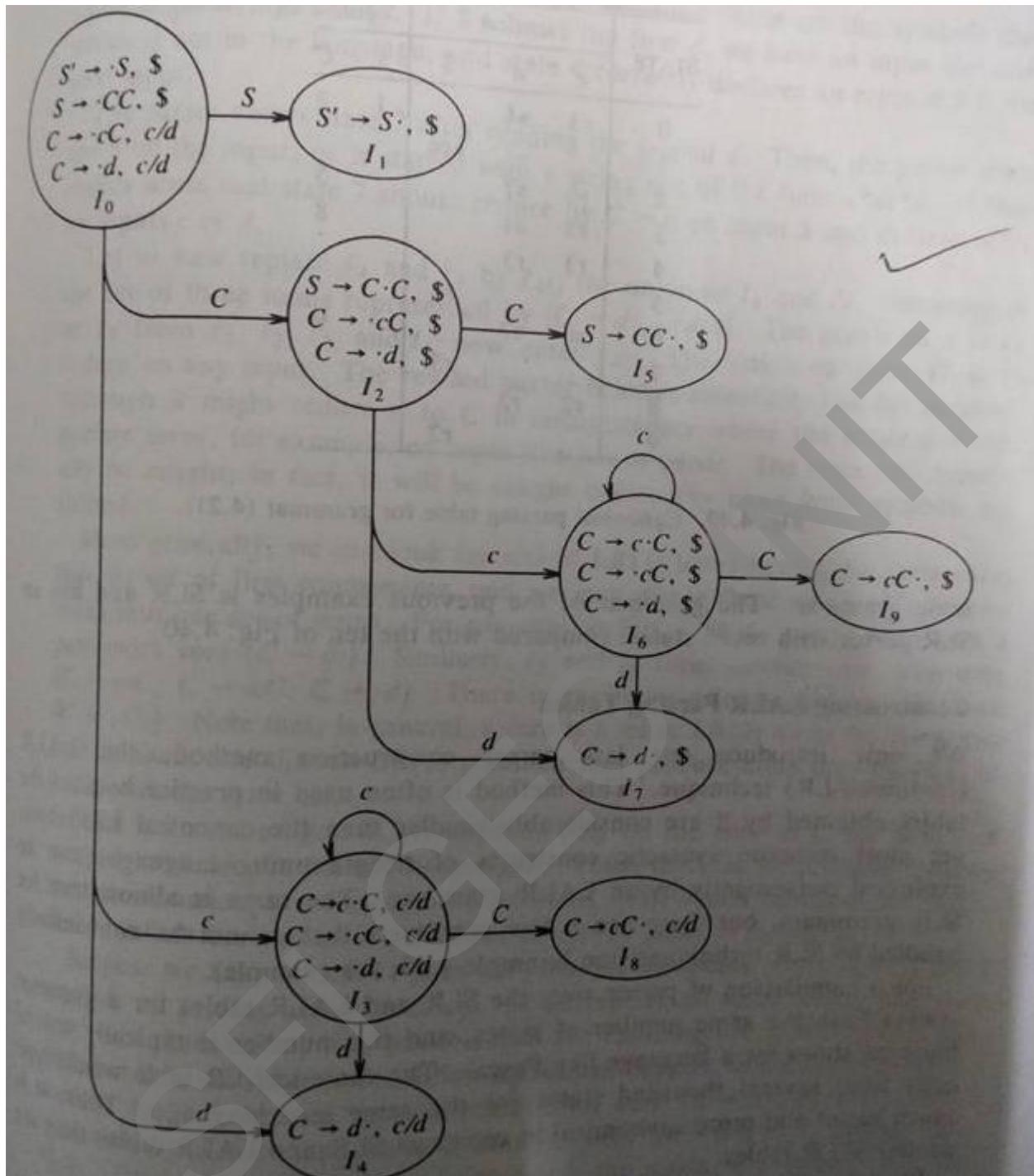
$$\begin{aligned} I_4: \quad L &\rightarrow \cdot * R \\ R &\rightarrow \cdot L \\ L &\rightarrow \cdot * R \\ L &\rightarrow \cdot \text{id} \end{aligned}$$

$$\begin{aligned} I_5: \quad L &\rightarrow \text{id} \cdot \\ I_6: \quad S &\rightarrow L = \cdot R \\ R &\rightarrow \cdot L \\ L &\rightarrow \cdot * R \\ L &\rightarrow \cdot \text{id} \end{aligned}$$

$$I_7: \quad L \rightarrow \cdot * R \cdot$$

$$I_8: \quad R \rightarrow L \cdot$$

$$I_9: \quad S \rightarrow L = R \cdot$$

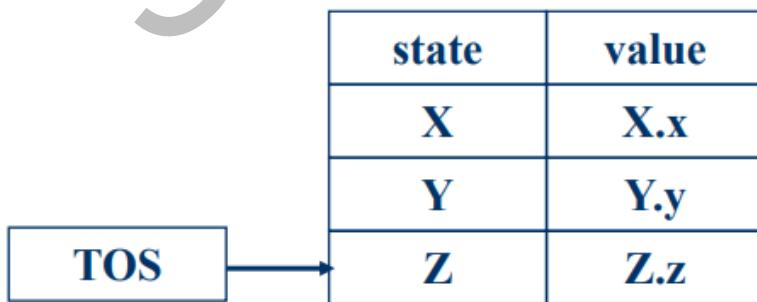


STATE	action			goto	
	c	d	\$	S	C
0	s3	s4		1	2
1			acc		5
2	s6	s7			8
3	s3	s4			
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

(9)

14.

- Syntax-directed definitions with only **synthesised attributes**
- can be evaluated by a bottom up parser (BUP) as input is parsed
- values associated with the attributes can be kept on the stack as extra fields
- implementation using an LR parser (e.g. **YACC**)
- e.g. **A => XYZ** and **A.a := f (X.x, Y.y, Z.z)**



S-attributed Definitions: Parser Example

production	semantic rules
L => E \n	print(val[TOS])
E => E ₁ + T	val[NTOS] := val[TOS-2] + val[TOS]
E => T	
T => T ₁ * F	val[NTOS] := val[TOS-2] * val[TOS]
T => F	
F => (E)	val[NTOS] := val[TOS-1]
F => id	

- NTOS = TOS - r + 1 (r is # symbols reduced on RHS)

example of $3 * 5 + 4$

S-attributed Definitions: Parser Example

input	state	value	production used
$3 * 5 + 4 \backslash n$	-	-	
$* 5 + 4 \backslash n$	3	3	
$* 5 + 4 \backslash n$	F	3	
$* 5 + 4 \backslash n$	T	3	
$5 + 4 \backslash n$	T *	3 :	$F \Rightarrow \text{digit}$
$+ 4 \backslash n$	T * 5	3 : 5	$T \Rightarrow F$
$+ 4 \backslash n$	T * F	3 : 5	
$+ 4 \backslash n$	T	15	
$+ 4 \backslash n$	E	15	
$4 \backslash n$	E +	15 :	$F \Rightarrow \text{digit}$
$\backslash n$	E + 4	15 : 4	$T \Rightarrow T * F$
$\backslash n$	E + F	15 : 4	$E \Rightarrow T$
$\backslash n$	E + T	15 : 4	
$\backslash n$	E	19	
$\backslash n$	E \backslash n	19 :	$F \Rightarrow \text{digit}$
	L	19	$T \Rightarrow F$
			$E \Rightarrow E + T$
			$L \Rightarrow E \backslash n$

(9)

PART E Answer any four full questions

15. The different storage allocation strategies are;
1. Static allocation lays out storage for all data objects at compile time.
 2. Stack allocation manages the run-time storage as a stack.
 3. Heap allocation: allocates and de-allocates storage as needed at runtime from a data known as the heap.

Static allocation

- In static allocation, names bound to storage as the program is compiled, so there is no need for a run-time support package.time
- Since the bindings do not change at runtime, every time a procedure activated, its run-time, names bounded to the same storage location.
- Therefore values of local names retained across activations of a procedure. That is when control returns to a procedure the value of the local are the same as they were when control left the last time.

Stack allocation : Storage allocation strategies

- All compilers for languages that use procedures, functions or methods as units of user functions define actions manage at least part of their runtime memory as a stack run-time stack.
- Each time a procedure called, space for its local variables is pushed onto a stack, and when the procedure terminates, space popped off the stack.

Calling Sequences: Storage allocation strategies

- Procedures called implemented in what is called as calling sequence, which consists of code that allocates an activation record on the stack and enters information into its fields.
- A return sequence is similar to code to restore the state of a machine so the calling procedure can continue its execution after the call.
- The code in calling sequence is often divided between the calling procedure (caller) and a procedure it calls (callee).
- When designing calling sequences and the layout of activation record, the following principles are helpful:
 1. Value communicated between caller and callee generally placed at the beginning of the callee's activation record, so they are as close as possible to the caller's activation record.
 2. Fixed length items generally placed in the middle. Such items typically include the control link, the access link, and the machine status field.,
 3. Items whose size may not be known early enough placed at the end of the activation record.
 4. We must locate the top of the stack pointer judiciously. A common approach is to have it point to the end of fixed length fields in the activation record to fix the end of fixed length fields in the activation record. Fixed length data can then be accessed by fixed offsets, known to the intermediate code generator, relative to the top of the stack pointer.

Storage allocation strategies

- The calling sequence and its division between caller and callee are as follows:
 1. The caller evaluates the actual parameters.
 2. The caller stores a return address and the old value of top_sp into the callee's activation record. The caller then increments the top_sp to the respective positions.
 3. The callee saves the register values and other status information.
 4. The callee initializes its local data and begins execution.

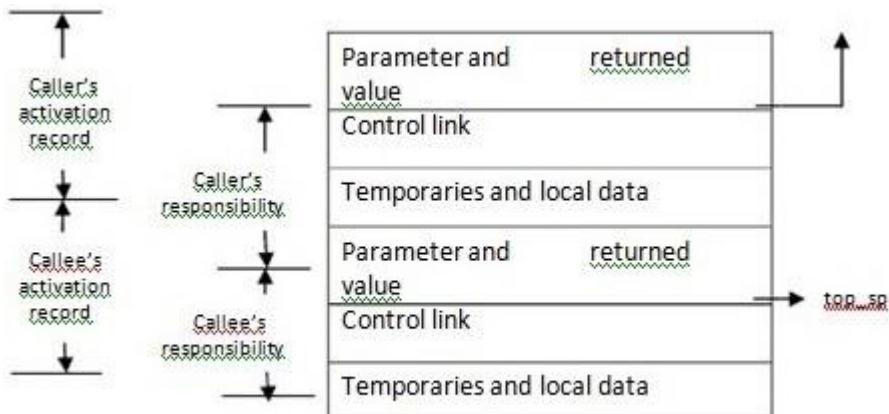


Fig. Division of task between caller and callee6.4

A suitable, corresponding return sequence is:

1. The callee places the return value next to the parameters.
2. Using the information in the machine status field, the callee restores `top_sp` and other registers, and then branches to the return address that the caller placed in the status field.
3. Although `top_sp` has been decremented, the caller knows where the return value is, relative to the current value of `top_sp`; the caller, therefore, may use that value.

Variable length data on stack: Storage allocation strategies

- The runtime memory management system must deal frequently with the allocation of management objects, the sizes of which are not known at the compile time, but which are local to a procedure and thus may be allocated on the stack.
- The same scheme works for objects of any type if they are local to the procedure called local have a size that depends on the parameter of the call.

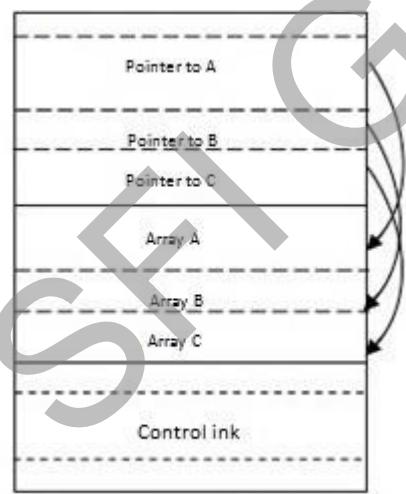


Fig. Access to dynamically allocated arrays Dangling Reference (Storage allocation strategies)

- Whenever storage allocated, the problem of dangling reference arises. The dangling reference occurs when there is a reference to storage that has been allocated.
- It is a logical error to u dangling reference, since, the value of de use de-allocated storage is undefined according to the semantics of most languages.
 - Whenever storage allocated, the problem of dangling reference arises. The dangling reference occurs when there is a reference to storage that has been allocated. (10)

16. a.

Three-Address Code

Intermediate code generator receives input from its predecessor phase, semantic analyzer, in the form of an annotated syntax tree. That syntax tree then can be converted into a linear representation, e.g., postfix notation. Intermediate code tends to be machine independent code. Therefore, code generator assumes to have unlimited number of memory storage (register) to generate code.

For example:

```
a = b + c * d;
```

The intermediate code generator will try to divide this expression into sub-expressions and then generate the corresponding code.

```
r1 = c * d;
```

```
r2 = b + r1;
```

```
a = r2
```

r being used as registers in the target program.

A three-address code has at most three address locations to calculate the expression. A three-address code can be represented in two forms : quadruples, triples, Indirect Triples
Quadruples

Each instruction in quadruples presentation is divided into four fields: operator, arg1, arg2, and result. The above example is represented below in quadruples format:

Op	arg ₁	arg ₂	result
*	c	d	r1
+	b	r1	r2
+	r2	r1	r3

=	r3		a
---	----	--	---

Triples

Each instruction in triples presentation has three fields : op, arg1, and arg2. The results of respective sub-expressions are denoted by the position of expression. Triples represent similarity with DAG and syntax tree. They are equivalent to DAG while representing expressions.

Op	arg ₁	arg ₂
*	c	d
+	b	(0)
+	(1)	(0)
=	(2)	

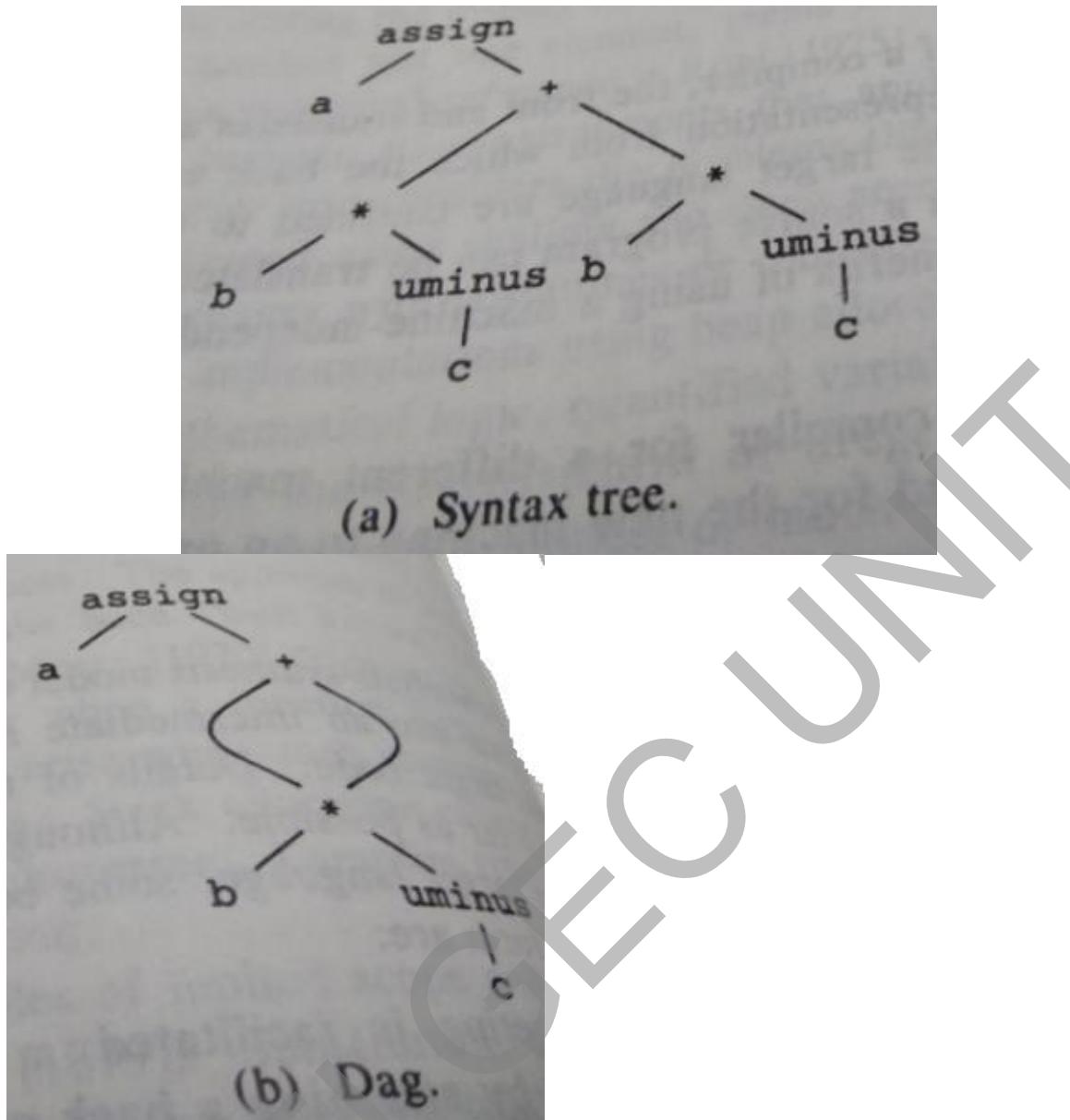
Triples face the problem of code immovability while optimization, as the results are positional and changing the order or position of an expression may cause problems.

Indirect Triples

This representation is an enhancement over triples representation. It uses pointers instead of position to store results. This enables the optimizers to freely re-position the sub-expression to produce an optimized code.

b.

(5)



(5)

17. A transformation of a program is called local if it can be performed by looking only at the statements in a basic block; otherwise, it is called global. Many transformations can be performed at both the local and global levels. Local transformations are usually performed first.

Function-Preserving Transformations

There are a number of ways in which a compiler can improve a program without changing the function it computes.

Function preserving transformations examples:

Common sub expression elimination

Copy propagation,

Dead-code elimination

Constant folding

The other transformations come up primarily when global optimizations are performed.

Frequently, a program will include several calculations of the offset in an array. Some of the duplicate calculations cannot be avoided by the programmer because they lie below the level of detail accessible within the source language.

Common Sub expressions elimination:

- An occurrence of an expression E is called a common sub-expression if E was previously computed, and the values of variables in E have not changed since the previous computation. We can avoid recomputing the expression if we can use the previously computed value.
- For example

```
t1: = 4*i  
t2: = a [t1]  
t3: = 4*j  
t4: = 4*i  
t5: = n  
t6: = b [t4] +t5
```

The above code can be optimized using the common sub-expression elimination as

```
t1: = 4*i  
t2: = a [t1]  
t3: = 4*j  
t5: = n  
t6: = b [t1] +t5
```

The common sub expression t4: =4*i is eliminated as its computation is already in t1 and the value of i is not been changed from definition to use.

Copy Propagation:

Assignments of the form $f := g$ called copy statements, or copies for short. The idea behind the copy-propagation transformation is to use g for f, whenever possible after the copy statement $f := g$. Copy propagation means use of one variable instead of another. This may not appear to be an improvement, but as we shall see it gives us an opportunity to eliminate x.

- For example:

```
x=Pi;
```

A=x*r*r;

The optimization using copy propagation can be done as follows: A=Pi*r*r;

Here the variable x is eliminated

Dead-Code Eliminations:

A variable is live at a point in a program if its value can be used subsequently; otherwise, it is dead at that point. A related idea is dead or useless code, statements that compute values that never get used. While the programmer is unlikely to introduce any dead code intentionally, it may appear as the result of previous transformations.

Example:

```
i=0;  
if(i=1)  
{  
a=b+5;  
}
```

Here, 'if' statement is dead code because this condition will never get satisfied.

Constant folding:

Deducing at compile time that the value of an expression is a constant and using the constant instead is known as constant folding. One advantage of copy propagation is that it often turns the copy statement into dead code.

For example,

a=3.14157/2 can be replaced by
a=1.570 thereby eliminating a division operation.

Loop Optimizations:

In loops, especially in the inner loops, programs tend to spend the bulk of their time. The running time of a program may be improved if the number of instructions in an inner loop is decreased, even if we increase the amount of code outside that loop.

Three techniques are important for loop optimization:

- Ø Code motion, which moves code outside a loop;
- Ø Induction-variable elimination, which we apply to replace variables from inner loop.

- Ø Reduction in strength, which replaces an expensive operation by a cheaper one, such as a multiplication by an addition.

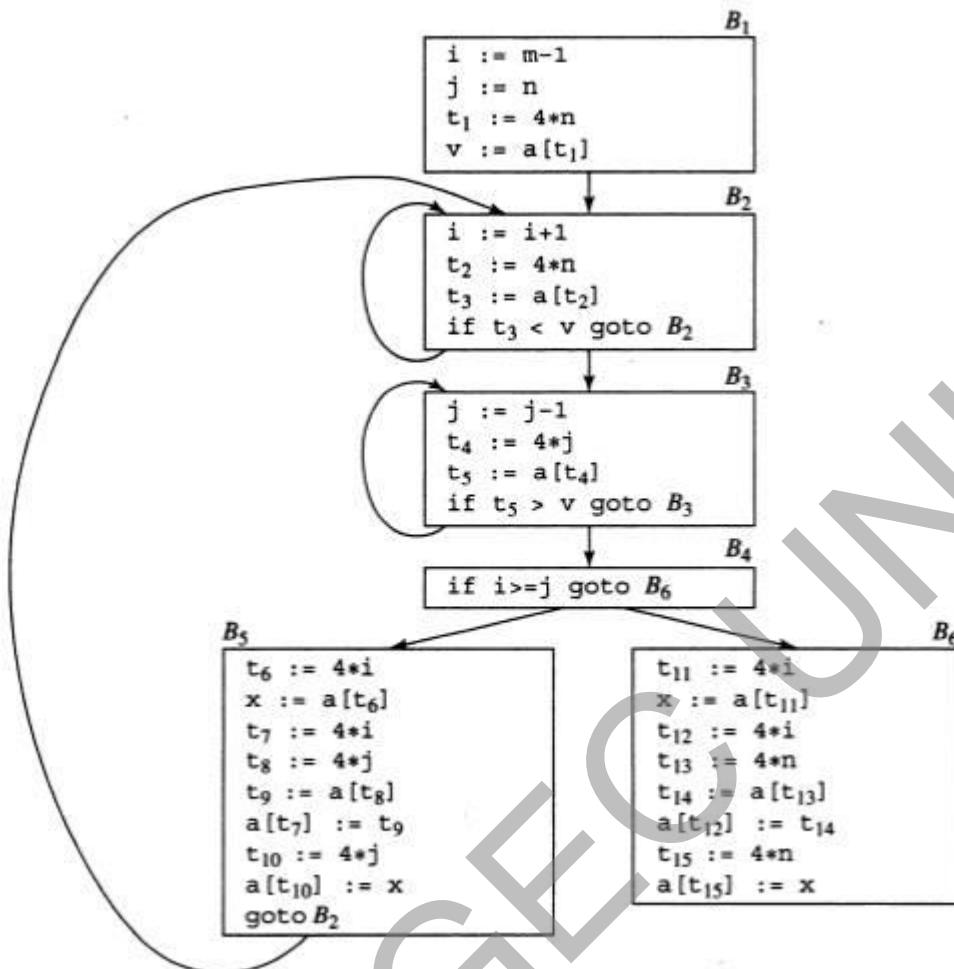


Fig. 5.2 Flow graph

Code Motion:

An important modification that decreases the amount of code in a loop is code motion. This transformation takes an expression that yields the same result independent of the number of times a loop is executed (a loop-invariant computation) and places the expression before the loop. Note that the notion “before the loop” assumes the existence of an entry for the loop. For example, evaluation of limit-2 is a loop-invariant computation in the following while-statement:

```
while (i <= limit-2) /* statement does not change limit */
```

Code motion will result in the equivalent of

```
t= limit-2;
while (i<=t) /* statement does not change limit or t */
```

Induction Variables :

Loops are usually processed inside out. For example consider the loop around B3. Note that the values of j and t4 remain in lock-step; every time the value of j decreases by 1, that of t4 decreases by 4 because $4*j$ is assigned to t4. Such identifiers are called induction variables.

When there are two or more induction variables in a loop, it may be possible to get rid of all but one, by the process of induction-variable elimination. For the inner loop around B3 in Fig.5.3 we cannot get rid of either j or t4 completely; t4 is used in B3 and j in B4.

However, we can illustrate reduction in strength and illustrate a part of the process of induction-variable elimination. Eventually j will be eliminated when the outer loop of B2- B5 is considered.

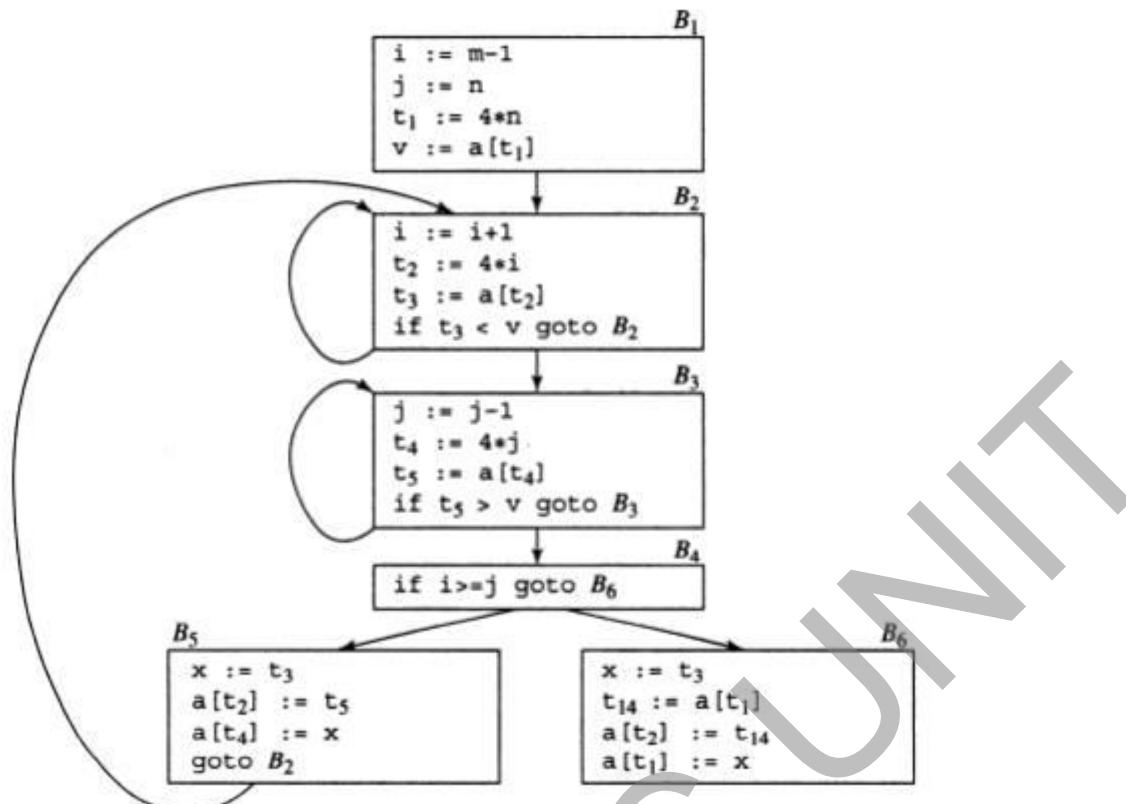
Example:

As the relationship $t4:=4*j$ surely holds after such an assignment to t4 in Fig. and t4 is not changed elsewhere in the inner loop around B3, it follows that just after the statement $j:=j-1$ the relationship $t4:= 4*j-4$ must hold. We may therefore replace the assignment $t4:= 4*j$ by $t4:= t4-4$. The only problem is that t4 does not have a value when we enter block B3 for the first time. Since we must maintain the relationship $t4=4*j$ on entry to the block B3, we place an initializations of t4 at the end of the block where j itself is initialized, shown by the dashed addition to block B1 in Fig.5.3.

The replacement of a multiplication by a subtraction will speed up the object code if multiplication takes more time than addition or subtraction, as is the case on many machines.

Reduction In Strength:

Reduction in strength replaces expensive operations by equivalent cheaper ones on the target machine. Certain machine instructions are considerably cheaper than others and can often be used as special cases of more expensive operators. For example, x^2 is invariably cheaper to implement as $x*x$ than as a call to an exponentiation routine. Fixed-point multiplication or division by a power of two is cheaper to implement as a shift. Floating-point division by a constant can be implemented as multiplication by a constant, which may be cheaper.

**Fig. 5.3 B5 and B6 after common subexpression elimination****Fig. 5.3 B5 and B6 after common subexpression elimination**

(10)

18. Code generator phase generates the target code taking input as intermediate code.

- The output of intermediate code generator may be given directly to code generation or may pass through code optimization before generating code.
- Code generator main tasks:
 - Instruction selection
 - Register allocation and assignment
 - Instruction ordering



Issues in Design of Code generation:

- Target code mainly depends on available instruction set and efficient usage of registers. The main issues in design of code generation are
 - Intermediate representation

- Linear representation like postfix and three address code or quadruples and graphical representation like Syntax tree or DAG.
- Assume type checking is done and input in free of errors.

2. Target Code

- The code generator has to be aware of the nature of the target language for which the code is to be transformed.
- That language may facilitate some machine-specific instructions to help the compiler generate the code in a more convenient way.
- The target machine can have either CISC or RISC processor architecture.
- The target code may be absolute code, re-locatable machine code or assembly language code.
- Absolute code can be executed immediately as the addresses are fixed.
- But in case of re-locatable it requires linker and loader to place the code in appropriate location and map (link) the required library functions.
- If it generates assembly level code then assemblers are needed to convert it into machine level code before execution.
- Re-locatable code provides great deal of flexibilities as the functions can be compiled separately before generation of object code.

1.1 Absolute machine language:

- Producing an absolute machine language program as output has the advantage that it can be placed in a fixed location in memory and immediately executed.

1.2 Relocatable machine language:

- Producing a relocatable machine language program as output allows subprograms to be compiled separately. A set of relocatable object modules can be linked together and loaded for execution by a linking loader.
- If the target machine does not handle relocation automatically, the compiler must provide explicit relocation information to the loader, to link the separately compiled program segments.

3. Assembly language:

- Producing an assembly language program as output makes the process of code generation somewhat easier.

Example

MOV R0, R1

ADD R1, R2

Target Machine supports for the following addressing modes

a. Absolute addressing mode

Example: MOV R0, M where M is the address of memory location of one of the operands.
MOV R0, M moves the contents of register R0 to memory location M.

b. Register addressing mode where both the operands are in register.

Example: ADD R0, R1

c. Immediate addressing mode – The operand value appears in the instruction.

Example: ADD # 1, R0

d. Index addressing mode- this is of the form C(R) where the address of operand is at the location C+Contents(R)

Example: MOV 4(R0), M the operand is located at address = contents (4+contents (R0)) Cost of instruction is defined as cost of execution plus the number of memory access.

3. Address mapping

- Address mapping defines the mapping between intermediate representations to address in the target code.
- These addresses are based on the runtime environment used like static, stack or heap.
- The identifiers are stored in symbol table during declaration of variables or functions, along with type.
- Each identifier can be accessed in symbol table based on width of each identifier and offset. The address of the specific instruction (in three address code) can be generated using back patching

4. Instruction Set

- The instruction set should be complete in the sense that all operations can be implemented.
- Sometimes a single operation may be implemented using many instruction (many set of instructions).
- The code generator should choose the most appropriate instruction.
- The instruction should be chosen in such a way that speed of execution is minimum or other machine related resource utilization should be minimum.
- The code generator takes Intermediate Representation as input and converts (maps) it into target machine's instruction set.
- One representation can have many ways (instructions) to convert it, so it becomes the responsibility of the code generator to choose the appropriate instructions wisely.

Example

The factors to be considered during instruction selection are:

- The uniformity and completeness of the instruction set.
- Instruction speed and machine idioms.
- Size of the instruction set.

Eg., for the following address code is:

$a := b + c$

$d := a + e$

inefficient assembly code is:

MOV b, R0	R0 ← b
ADD c, R0	R0 ← c + R0
MOV R0, a	a ← R0
MOV a, R0	R0 ← a
ADD e, R0	R0 ← e + R0
MOV R0 , d	d ← R0

Here the fourth statement is redundant, and so is the third statement if 'a' is not subsequently used.

5. Register Allocation

- A program has a number of values to be maintained during the execution.
- The target machine's architecture may not allow all of the values to be kept in the CPU memory or registers.
- Code generator decides what values to keep in the registers.
- Also, it decides the registers to be used to keep these values.
- Instructions involving register operands are usually shorter and faster than those involving operands in memory.
- Therefore efficient utilization of registers is particularly important in generating good code.
- During register allocation we select the set of variables that will reside in registers at each point in the program.
- During a subsequent register assignment phase, we pick the specific register that a variable will reside in.

6. Memory Management

- Mapping names in the source program to addresses of data objects in run-time memory is done cooperatively by the front end and the code generator.
- A name in a three- address statement refers to a symbol-table entry for the name.
- From the symbol-table information, a relative address can be determined for the name in a data area for the procedure.

7. Evaluation of order

- At last, the code generator decides the order in which the instruction will be executed.
- It creates schedules for instructions to execute them.
- The order in which computations are performed can affect the efficiency of the target code.
- Some computation orders require fewer registers to hold intermediate results than others.

(10)
Total: (40)

Reg. No: _____

Name: _____

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
SIXTH SEMESTER B.TECH DEGREE MODEL EXAMINATION, MARCH 2018
CS304 COMPILER DESIGN

Max. Marks: 100**Duration: 3 hours**

PART A
(Answer all questions. Each carries 3 marks.)

- 1.What advantages are there to a language processing system in which the compiler produces assembly language rather than machine language. (3)
 - 2.Describe the languages denoted by the following regular expressions. (3)
- 1.a(a|b)*a 2.((ε|a)b*)* 3.(a|b)*a(a|b)(a|b)
- 3.Consider the context-free grammar: $S \rightarrow S S + | S S^* | a$ and the string aa + a*. (3)
- 1.Give a leftmost derivation for the string.
 - 2.Give a rightmost derivation for the string.
 3. Give a parse tree for the string.
- 4.Design grammars for the following languages: (3)
1. The set of all strings of 0s and 1s that are palindromes; that is, the string reads the same backward as forward.
 - 2.The set of all strings of 0s and 1s with an equal number of 0s and 1s.

PART B
(Answer any two questions. Each carries 9 marks.)

- 5.a) Explain the phases of a compiler. (6)
 - b)Calculate the number of tokens in the following C statement
 $\text{printf("i = \%d, \&i = \%x", i, \&i);}$ (2)
- 6.Construct NFA for the regular expression a(a|b)*a . And convert it to DFA. (9)
 - 7.a) Consider the grammar, remove left recursion. (3)
- $S \rightarrow A a | b$
 $A \rightarrow A c | S d | \epsilon$
- b) Compute FIRST and FOLLOW
 $S \rightarrow ACB|Cbb|Ba$ $A \rightarrow da|BC$ $B \rightarrow g|\epsilon$ $C \rightarrow h|\epsilon$ (4)
- c) $S \rightarrow a | ab | abc | abcd | e | f$ Perform left factoring. (2)

PART C
(Answer all questions. Each carries 3 marks.)

- 8.Explain Handle pruning. (3)
- 9.What is operator precedence grammar?.Give examples. (3)
- 10.Explain the applications of syntax directed translation (3)
- 11.Define synthesized and inherited translation. (3)

PART D
(Answer any two questions. Each carries 9 marks.)

12. Show that the following is LR(1) but not LALR(1) (9)

$$S \rightarrow A\ a \mid b\ A\ c \mid B\ c \mid b\ B\ a$$

$$A \rightarrow d$$

$$B \rightarrow d$$

13.a) Write the syntax directed definition of a simple desk calculator. (6)

b) Describe all the viable prefixes for the grammar $S \rightarrow 0S1|01$ (3)

14. a) Describe SDT for infix-to-prefix translation (3)

b) Describe the conflicts in shift reduce parsing (3)

c) Compute leading and trailing for the following grammar (3)

$$S \rightarrow a \mid ^\wedge \mid (T)$$

$$T \rightarrow T, S \mid S$$

PART E
(Answer any four questions. Each carries 10 marks.)

15.a) Explain with example how three address code is partitioned to basic blocks (4)

b) Explain peephole optimization (6)

16. What are the issues in the design of a code generator. (10)

17.a) Construct the DAG for the expression. (4)

$$((x+y)-((x+y)*(x-y)))+((x+y)*(x-y))$$

b) Translate the arithmetic expression $a+-(b+c)$ into (6)

- a) Syntax tree b) Quadruples c) Triples d) Indirect triples

18. Write a note on the translation of boolean expression. (10)

19. Explain optimization of basic blocks. (10)

20. Discuss the different storage allocation strategies. (10)

Reg. No: _____

Name: _____

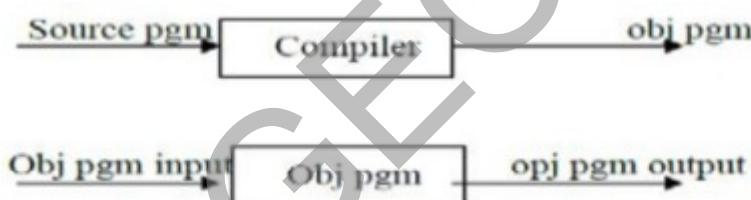
**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
SIXTH SEMESTER B.TECH DEGREE MODEL EXAMINATION, MARCH 2018
CS304 COMPILER DESIGN**

Max. Marks: 100**Duration: 3 hours****Answer key**

PART A
(Answer all questions. Each carries 3 marks.)

1.What advantages are there to a language processing system in which the compiler produces assembly language rather than machine language. (3)

Compiler is a translator program that translates a program written in (HLL) the source program and translate it into an equivalent program in (MLL) the target program. As an important part of a compiler is error showing to the programmer. Executing a program written in HLL programming language is basically of two parts. The source program must first be compiled translated into a object program. Then the results object program is loaded into a memory executed.



The compiler may produce an assembly-language program as its output, because assembly language is easier to produce as output and is easier to debug. Programmers found it difficult to write or read programs in machine language. They began to use a mnemonic (symbols) for each machine instruction, which they would subsequently translate into machine language. Such a mnemonic machine language is now called an assembly language. Programs known as assembler were written to automate the translation of assembly language into machine language. The input to an assembler program is called source program, the output is a machine language translation (object program).

2. Describe the languages denoted by the following regular expressions. (3)

$$1. a(a|b)^*a \quad 2. ((\epsilon|a)b^*)^* \quad 3. (a|b)^*a(a|b)(a|b)$$

- 1.String of a's and b's that start and end with a.
- 2.String of a's and b's.

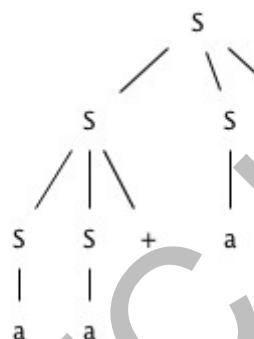
- 3.String of a's and b's that the character third from the last is a.
 3.Consider the context-free grammar: $S \rightarrow S\ S^+ \mid S\ S^* \mid a$ and the string aa + a*. (3)

- 1.Give a leftmost derivation for the string.
- 2.Give a rightmost derivation for the string.
3. Give a parse tree for the string.

1.S =lm=> SS* => SS+S* => aS+S* => aa+S* => aa+a*

2.S =rm=> SS* => Sa* => SS+a* => Sa+a* => aa+a*

3.



- 4.Design grammars for the following languages: (3)

1. The set of all strings of 0s and 1s that are palindromes; that is, the string reads the same backward as forward.

2.The set of all strings of 0s and 1s with an equal number of 0s and 1s.

1.S -> 0S0 | 1S1 | 0 | 1 | ε

2.S -> 0S1S | 1S0S | ε

PART B

(Answer any two questions. Each carries 9 marks.)

- 5.a) Explain the phases of a compiler. (6)

Phases of a compiler: A compiler operates in phases. A phase is a logically interrelated operation that takes source program in one representation and produces output in another representation. The phases of a compiler are shown in below

There are two phases of compilation.

- a. Analysis (Machine Independent/Language Dependent)
- b. Synthesis(Machine Dependent/Language independent)

Compilation process is partitioned into no-of-sub processes called ‘phases’.

Lexical Analysis:-

LA or Scanners reads the source program one character at a time, carving the source program into a sequence of atomic units called tokens.

Syntax Analysis:-

The second stage of translation is called Syntax analysis or parsing. In this phase expressions, statements, declarations etc... are identified by using the results of lexical analysis. Syntax analysis is aided by using techniques based on formal grammar of the programming language.

Intermediate Code Generations:-

An intermediate representation of the final machine language code is produced. This phase bridges the analysis and synthesis phases of translation.

Code Optimization :-

This is optional phase described to improve the intermediate code so that the output runs faster and takes less space.

Code Generation:-

The last phase of translation is code generation. A number of optimizations to reduce the length of machine language program are carried out during this phase. The output of the code generator is the machine language program of the specified computer.

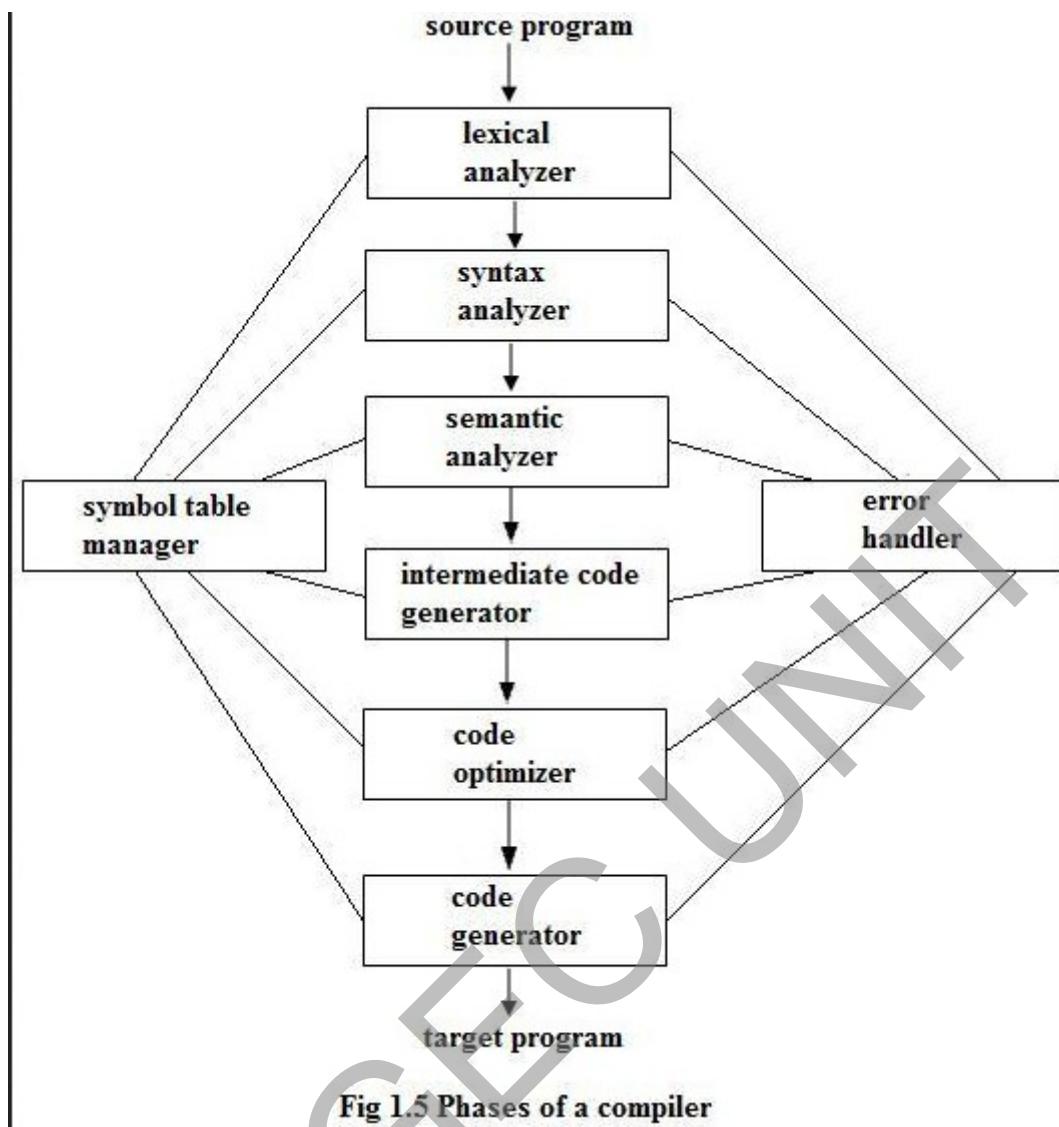
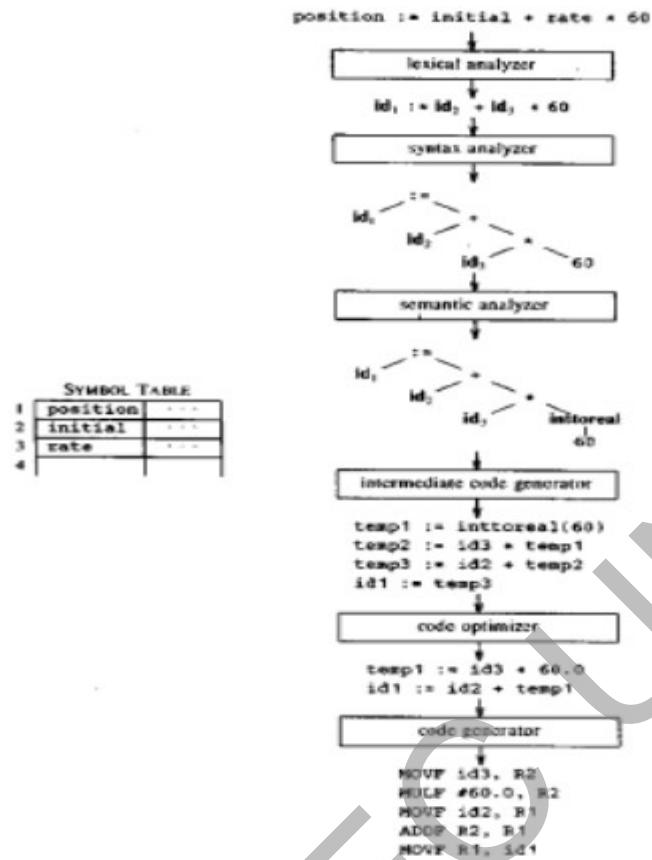


Fig 1.5 Phases of a compiler



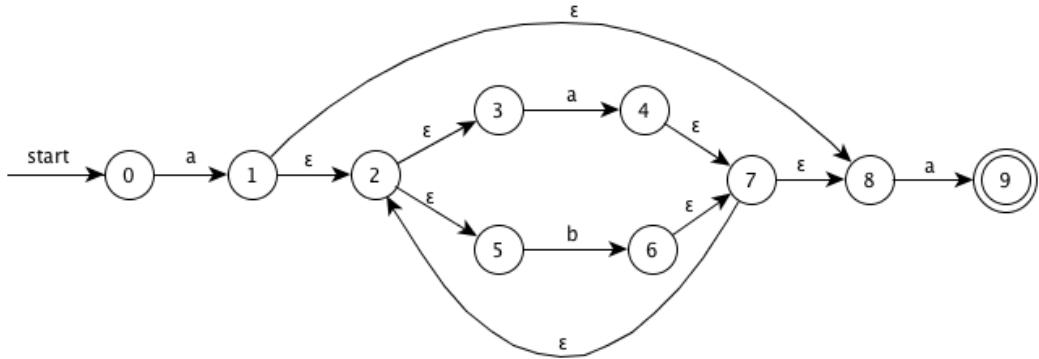
5 b) Calculate the number of tokens in the following C statement

printf("i = %d, &i = %x", i, &i); (2)

No of Tokens-10

- 1.printf
- 2.(
- 3.“i=%d,&i=%x”
- 4.,
- 5.i
- 6.,
- 7.&
- 8.i
- 9.)
- 10.;

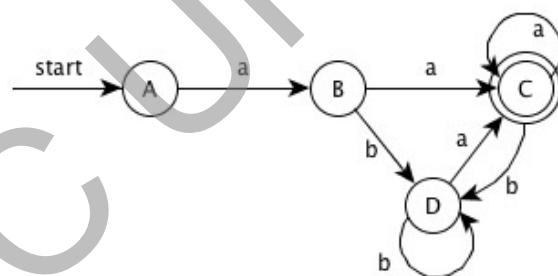
6. Construct NFA for the regular expression $a(a|b)^*a$. And convert it to DFA. (9)



I

DFA:

NFA	DFA	a	b
{0}	A	B	
{1,2,3,5,8}	B	C D	
{2,3,4,5,7,8,9}	C	C D	
{2,3,5,6,7,8}	D	C D	



7.a) Consider the grammar, remove left recursion. (3)

$$\begin{aligned} S &\rightarrow Aa \mid b \\ A &\rightarrow Ac \mid Sd \mid \epsilon \end{aligned}$$

$$\begin{aligned} S &\rightarrow Aa \mid b \\ A &\rightarrow bdA' \mid A' \\ A' &\rightarrow cA' \mid adA' \mid \epsilon \end{aligned}$$

b) Compute FIRST and FOLLOW

$$S \rightarrow ACB \mid Cbb \mid Ba \quad A \rightarrow da \mid BC \quad B \rightarrow g \mid \epsilon \quad C \rightarrow h \mid \epsilon \quad (4)$$

$$\text{FIRST}(S) = \text{FIRST}(A) \cup \text{FIRST}(B) \cup \text{FIRST}(C) = \{ d, g, h, \epsilon, b, a \}$$

$$\text{FIRST}(A) = \{ d \} \cup \text{FIRST}(B) = \{ d, g, h, \epsilon \}$$

$$\text{FIRST}(B) = \{ g, \epsilon \}$$

$$\text{FIRST}(C) = \{ h, \epsilon \}$$

FOLLOW Set

$$\text{FOLLOW}(S) = \{ \$ \}$$

$$\text{FOLLOW}(A) = \{ h, g, \$ \}$$

$$\text{FOLLOW}(B) = \{ a, \$, h, g \}$$

$$\text{FOLLOW}(C) = \{ b, g, \$, h \}$$

c) $S \rightarrow a \mid ab \mid abc \mid abcd \mid e \mid f$ Perform left factoring. (2)

$$S \rightarrow a S' \mid e \mid f$$

$$S' \rightarrow b S'' \mid \epsilon \quad - \text{For single } a$$

$$S'' \rightarrow c S''' \mid \epsilon \quad - \text{For } ab$$

$$S''' \rightarrow d \mid \epsilon \quad - \text{For } abc$$

PART C

(Answer all questions. Each carries 3 marks.)

8.Explain Handle pruning. (3)

Bottom-up parsing during a left-to-right scan of the input constructs a rightmost derivation in reverse. A handle is a substring that matches the body of a production and whose reduction represents one step along the reverse of a rightmost derivation. A rightmost derivation in reverse can be obtained by handle pruning. That is, we start with a string of terminals w to be parsed. If w is a sentence of the grammar at hand, then finally reach to the start symbol.

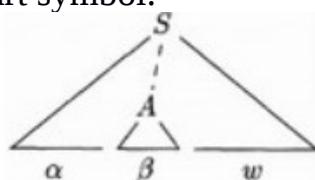


Figure 4.27: A handle $A \rightarrow \beta$ in the parse tree for $\alpha\beta w$

$$S = \gamma_0 \xrightarrow{\text{rlr}} \gamma_1 \xrightarrow{\text{rlr}} \gamma_2 \xrightarrow{\text{rlr}} \dots \xrightarrow{\text{rlr}} \gamma_{n-1} \xrightarrow{\text{rlr}} \gamma_n = w.$$

9.What is operator precedence grammar?.Give examples. (3)

An operator precedence grammar is a context-free grammar that has the property (among others) that no production has either an empty right-hand side or two adjacent nonterminals in its right-hand side. These properties allow precedence relations to be defined between the terminals of the grammar.

The grammar

$$\begin{aligned} E &\rightarrow EAE \mid (E) \mid -E \mid id \\ A &\rightarrow + \mid - \mid * \mid / \mid \uparrow \end{aligned}$$

is not an operator grammar, because the right side EAE has two (in fact three) consecutive nonterminals;

However, if we substitute for A each of its alternatives, we obtain the following operator grammar:

$$E \rightarrow E+E \mid E-E \mid E^*E \mid E/E \mid E\uparrow E \mid (E) \mid -E \mid id$$

10.Explain the applications of syntax directed translation (3)

Syntax-directed translation (SDT) refers to a method of compiler implementation where the source language translation is completely driven by the parser, i.e., based on the syntax of the language. The parsing process and parse trees are used to direct semantic analysis and the translation of the source program. Almost all modern compilers are syntax-directed. SDT can be a separate phase of a compiler or we can augment our conventional grammar with information to control the semantic analysis and translation. Such grammars are called attribute grammars.

Applications of Syntax-Directed Translation

1 Construction of Syntax Trees

2 The Structure of a Type

11. Define synthesized and inherited translation. (3)

Synthesized Attributes

A SDD defines zero or more *attributes* for each nonterminal and terminal. A *synthesized attribute* has its value defined in terms of attributes at its children.

Example:

A production $A \rightarrow BC$ and a rule like

$$A.att := f(B.att1, B.att2, C.att3)$$

makes $A.att$ a synthesized attribute.

- It is conventional to call the attributes of terminals, which are generally lexical values returned by the lexical analyzer, “synthesized.”
 - A SDD with only synthesized attributes is called an *S-attributed definition*. All the examples seen so far are S-attributed.
-

Implementing S-attributed Definitions

It is easy to implement an S-attributed definition on an LR grammar by a postfix SDT.

- Values of attributes for symbol X are stored along with any occurrence of X on the parsing stack.
 - When a reduction occurs, the values of attributes for the nonterminal on the left are computed from the attributes for the symbols on the right (which are all at the top of the stack), before the stack is popped and the left side pushed onto the stack, along with its attributes.
-

Infix to postfix conversion

$$E.post := E_1.post \mid T.post \mid '+'$$

which can be turned into SDT

$$E \rightarrow E + T \{ \text{print } '+' \}$$

In principle, if we wanted to translate to prefix, we could blithely write the SDD with rules like:

$$E.pre = '+' \mid E_1.pre \mid T.pre$$

The corresponding SDT, with rules like

$$E \rightarrow \{ \text{print } '+' \} E + T$$

is legal, but *cannot be executed as written*, because the grammar will not let the parser know when to execute the print actions. Rather, it must be implemented as discussed previously: build the parse tree and then traverse it in preorder to execute the actions.

Inherited Attributes

Any attribute that is not synthesized is called *inherited*.

- The typical inherited attribute is computed at a child node as a function of attributes of its parent.
 - It is also possible that attributes at sibling nodes (including the node itself) will be used.
-

Example:

Consider the grammar with nonterminals

1. $D = \text{type definition}$.
2. $T = \text{type (integer or real)}$.
3. $L = \text{list of identifiers}$.

and SDD

$$\begin{aligned}
 D &\rightarrow T L \\
 &\quad L.type := T.type \\
 T &\rightarrow \text{int} \\
 &\quad T.type := \text{INT} \\
 T &\rightarrow \text{real} \\
 &\quad T.type := \text{REAL} \\
 L &\rightarrow L_1, id \\
 &\quad L_1.type := L.type; \\
 &\quad \text{addtype}(id.entry, L.type) \\
 L &\rightarrow id \\
 &\quad \text{addtype}(id.entry, L.type)
 \end{aligned}$$

Notes

- The call to *addtype* is a side effect of the SDD. The timing of the call is not clear, but it must take place at least once for every occurrence of the last two productions in the parse tree.
- We shall discuss “L-attributed definitions,” where there is a natural order of evaluation, making ambiguities like this one disappear.
- We assume that terminal *id* has an attribute *entry*, which is a pointer to the symbol table entry for that identifier. The effect of *addtype* is to enter the declared type for that identifier.

PART D

(Answer any two questions. Each carries 9 marks.)

12. Show that the following is LR(1) but not LALR(1) (9)

$$S \rightarrow A a \mid b A c \mid B c \mid b B a$$

$$A \rightarrow d$$

$$B \rightarrow d$$

$$S \rightarrow A a \mid b A c \mid B c \mid b B a$$

$$A \rightarrow d$$

$B \rightarrow d$

LR(1) Parser

State 0: [$S \rightarrow *Aa , \$$]

[$S \rightarrow *bAc , \$$]

[$S \rightarrow *Bc , \$$]

[$S \rightarrow *bBa , \$$]

[$A \rightarrow *d , a$]

[$B \rightarrow *d , c$]

State 1: Goto(State 0, d) = [$A \rightarrow d^* , a$]

[$B \rightarrow d^* , c$]

State 2: Goto(State 0, b) = [$S \rightarrow b^*Ac , \$$]

[$S \rightarrow b^*Ba , \$$]

[$A \rightarrow *d , c$]

[$B \rightarrow *d , a$]

State 3: Goto(State 2, d) = [$A \rightarrow d^* , c$]

[$B \rightarrow d^* , a$]

LALR(1) Parser

Merge lookaheads for State 1 and State 3 in

LR(1) parser to

create new state:

State = Merge (State 1, State 3) = [$A \rightarrow d^* , a$]

[$A \rightarrow d^* , c$]

[$B \rightarrow d^* , c$]

[$B \rightarrow d^* , a$]

Reduce/reduce conflict for lookahead "a" and "c"

(i.e., can't decide whether to reduce "d" to A or B)

13.a) Write the syntax directed definition of a simple desk calculator. (6)

A Syntax Directed Definition(SDD) is a context-free grammar together with attributes and rules. Attributes are associated with grammar symbols and rules are associated with productions. If X is a symbol and a is one of its attributes, then we write X.a to denote the value of a at a particular parse-tree node labelled X. If we implement the nodes of the parse tree by records or objects, then the attributes of X can be implemented by data fields in the records that represent the nodes for X. The attributes are evaluated by the semantic rules attached to the productions.

Example:

PRODUCTION SEMANTIC RULE

$E \rightarrow E1 + T$ $E.code = E1.code \parallel T.code \parallel '+'$

SDDs are highly readable and give high-level specifications for translations. But they
hide many implementation details.

Production	Semantic Rules
$L \rightarrow E \text{ n}$	$\text{print}(E.\text{val})$
$E \rightarrow E_1 + T$	$E.\text{val} := E_1.\text{val} + T.\text{val}$
$E \rightarrow T$	$E.\text{val} := T.\text{val}$
$T \rightarrow T_1 * F$	$T.\text{val} := T_1.\text{val} * F.\text{val}$
$T \rightarrow F$	$T.\text{val} := F.\text{val}$
$F \rightarrow (E)$	$F.\text{val} := E.\text{val}$
$F \rightarrow \text{digit}$	$F.\text{val} := \text{digit}.\text{lexval}$

Fig. 5.2. Syntax-directed definition of a simple desk calculator

- b) Describe all the viable prefixes for the grammar $S \rightarrow 0S1|01$ (3)
 The prefixes of right sentential forms that can appear on the stack of a shift - reduce parser are called viable prefixes. By definition, a viable prefix is a prefix of a right sentential form that does not continue past the right end of the rightmost handle of that sentential form.

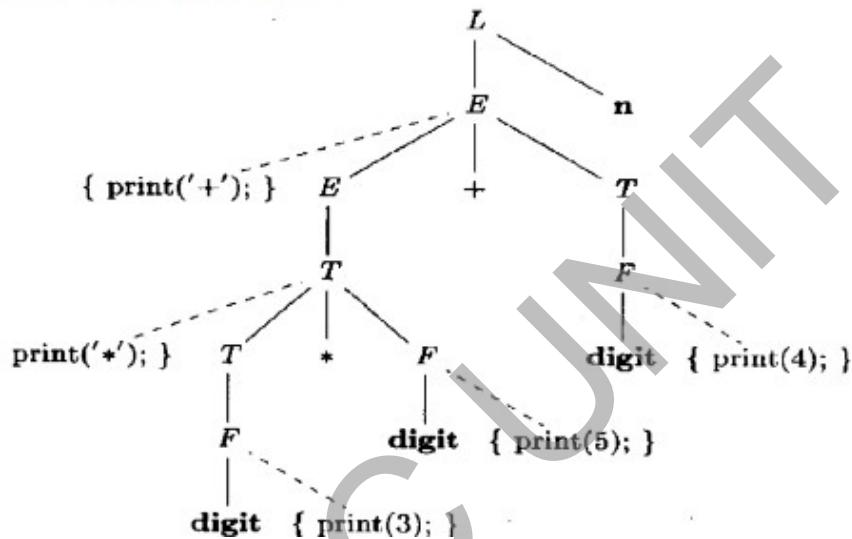
Stack	Input	Action
\$	000111\$	Shift
\$0	00111\$	Shift
\$00	0111\$	Shift
\$000	111\$	Shift
\$0001	11\$	Reduce $S \rightarrow 01$
\$00S	1\$	Shift
\$00S1	\$	Reduce $S \rightarrow 0S1$
\$0S		Shift
\$0S1		Reduce $S \rightarrow 0S1$
\$S		Accept

Stack always contains viable prefixes.

14. a) Describe SDT for infix-to-prefix translation (3)

- 1) $L \rightarrow E \text{ n}$
- 2) $E \rightarrow \{\text{print('+';}\} E_1 + T$
- 3) $E \rightarrow T$
- 4) $T \rightarrow T_1 * F \{\text{print('*');}\}$
- 5) $T \rightarrow F$
- 6) $F \rightarrow (E)$
- 7) $F \rightarrow \text{digit } \{\text{print(digit.lexval);}\}$

Parse Tree with Actions Embedded



b) Describe the conflicts in shift reduce parsing (3)

There are context-free grammars for which shift-reduce parsing cannot be used. Every shift-reduce parser for such a grammar can reach configuration in which the parser, knowing the entire stack and also the next k input symbols, cannot decide whether to shift or to reduce (a shift/reduce conflict), or cannot decide which of several reductions to make (a reduce/reduce conflict).

c) Compute leading and trailing for the following grammar (3)

$$\begin{aligned} S &\rightarrow a \mid \wedge \mid (T) \\ T &\rightarrow T, S \mid S \end{aligned}$$

The algorithm for finding LEADING(A) where A is a non-terminal is given below
LEADING(A)

{

1. 'a' is in Leading(A) if $A \geq \gamma\alpha\delta$ where γ is ϵ or any Non-Terminal
2. If 'a' is in Leading(B) and $A \geq B\alpha$, then a in Leading(A)

}

Step 1 of algorithm indicates how to add the first terminal occurring in the RHS of every production directly. Step 2 of the algorithm indicates to add the first terminal, through another non-terminal B to be included indirectly to the LEADING() of every non-terminal. Similarly the algorithm to find TRAILING(A) is given below.

TRAILING(A)

{

1. a is in Trailing(A) if $A \succ^* ya\delta$ where δ is ϵ or any Non-Terminal
2. If a is in Trailing(B) and $A \succ^* \alpha B$, then a in Trailing(A)

}

Leading(S)={a, \wedge ,(, ,)}

Leading(T)={,}

Trialing(S)={a, \wedge ,),,,}

Trailing(T)={,}

PART E

(Answer any four questions. Each carries 10 marks.)

- 15.a) Explain with example how three address code is partitioned to basic blocks (4)

A program flow graph is also necessary for compilation. the nodes are the basic blocks. There is an arc from block B1 to block B2 if B2 can follow B1 in some execution sequence.

A basic block is a sequence of consecutive statements in which flow of control enters at the beginning and leaves at the end without halts or possibility of branching except at the end.

Basic blocks are important concepts from both code generation and optimization point of view.

```
w = 0;
x = x + y;
y = 0;
if( x > z)
{
    y = x;
    x++;
}
else
{
    y = z;
    z++;
}
w = x + z;
```

Source Code

```
w = 0;
x = x + y;
y = 0;
if( x > z)
```

```
y = x;
x++;
```

```
y = z;
z++;
```

```
w = x + z;
```

Basic Blocks

Basic blocks play an important role in identifying variables, which are being used more than once in a single basic block. If any variable is being used more than once, the register memory allocated to that variable need not be emptied unless the block finishes execution.

Control Flow Graph

Basic blocks in a program can be represented by means of control flow graphs. A control flow graph depicts how the program control is being passed among the blocks. It is a useful tool that helps in optimization by help locating any unwanted loops in the program.

B1

```
w = 0;
x = x + y;
y = 0;
if( x > z )
```

B2

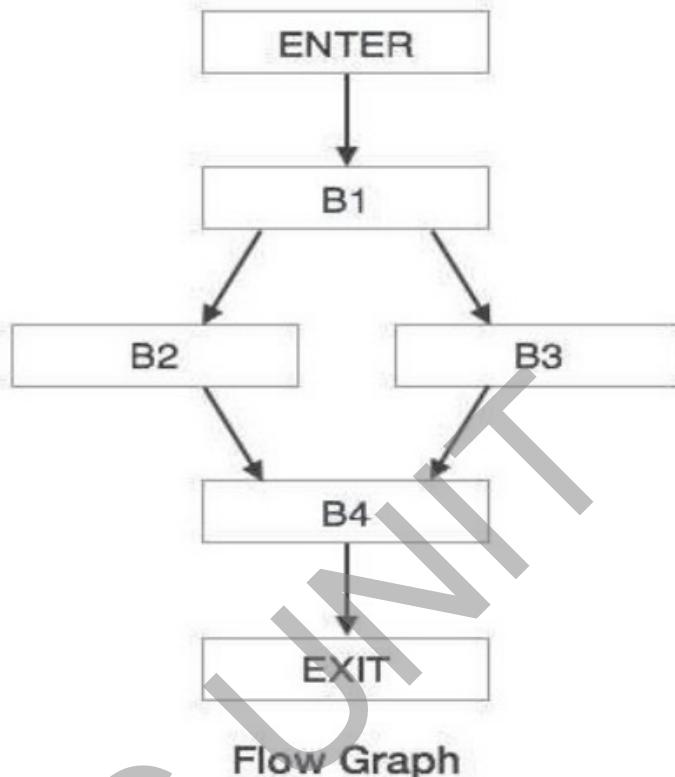
```
y = x;
x++;
```

B3

```
y = z;
z++;
```

B4

```
w = x + z;
```

Basic Blocks**Flow Graph**

b) Explain peephole optimization (6)

A statement-by-statement code-generations strategy often produce target code that contains redundant instructions and suboptimal constructs .The quality of such target code can be improved by applying “optimizing” transformations to the target program.

☞ A simple but effective technique for improving the target code is peephole optimization,a method for trying to improving the performance of the target program by examining a short sequence of target instructions (called the peephole) and replacing these instructions by a shorter or faster sequence, whenever possible.

☞ The peephole is a small, moving window on the target program. The code in the peephole need not contiguous, although some implementations do require this.it is characteristic of peephole optimization that each improvement may spawn opportunities for additional improvements.

☞ Characteristic of peephole optimizations:

Redundant-instructions elimination

Flow-of-control optimizations

Algebraic simplifications

Use of machine idioms

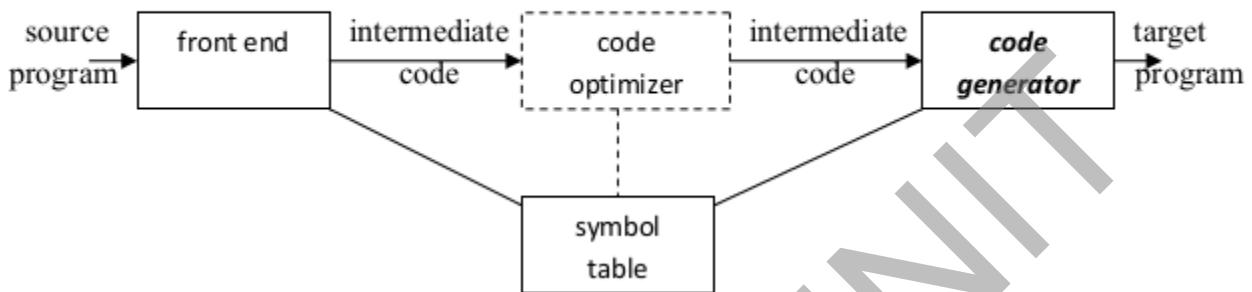
Unreachable Code.

16.What are the issues in the design of a code generator.

(10)

The final phase in compiler model is the code generator. It takes as input an intermediate representation of the source program and produces as output an equivalent target program. The code generation techniques presented below can be used whether or not an optimizing phase occurs before code generation.

Position of code generator



ISSUES IN THE DESIGN OF A CODE GENERATOR

The following issues arise during the code generation phase :

1. Input to code generator
2. Target program
3. Memory management
4. Instruction selection
5. Register allocation
6. Evaluation order

. Input to code generator:

☞ The input to the code generation consists of the intermediate representation of the source program produced by front end , together with information in the symbol table to determine run-time addresses of the data objects denoted by the names in the intermediate representation.

☞ Intermediate representation can be :

- a. Linear representation such as postfix notation
- b. Three address representation such as quadruples
- c. Virtual machine representation such as stack machine code
- d. Graphical representations such as syntax trees and dags.

☞ Prior to code generation, the front end must be scanned, parsed and translated into intermediate representation along with necessary type checking. Therefore, input to code

generation is assumed to be error-free.

2. Target program:

☞ The output of the code generator is the target program. The output may be :

a. Absolute machine language

- It can be placed in a fixed memory location and can be executed immediately.

b. Relocatable machine language

- It allows subprograms to be compiled separately.

c. Assembly language

- Code generation is made easier.

3. Memory management:

- ⊐ Names in the source program are mapped to addresses of data objects in run-time memory by the front end and code generator.

- ⊐ It makes use of symbol table, that is, a name in a three-address statement refers to a

symbol-table entry for the name.

- ⊐ Labels in three-address statements have to be converted to addresses of instructions.

For example,

j : goto i generates jump instruction as follows :

- ≡ if $i < j$, a backward jump instruction with target address equal to location of code for quadruple i is generated.

- ≡ if $i > j$, the jump is forward. We must store on a list for quadruple i the location of the first machine instruction generated for quadruple j. When i is processed, the machine locations for all instructions that forward jumps to i are filled.

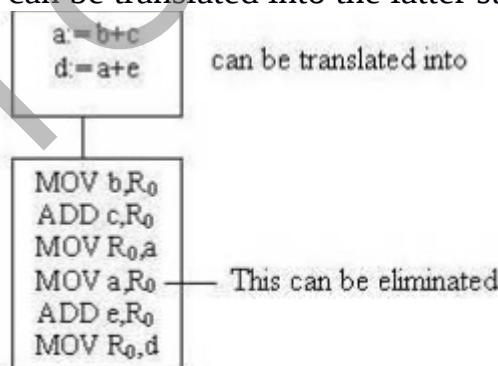
4. Instruction selection:

- ⊐ The instructions of target machine should be complete and uniform.

- ⊐ Instruction speeds and machine idioms are important factors when efficiency of target program is considered.

- ⊐ The quality of the generated code is determined by its speed and size.

- ⊐ The former statement can be translated into the latter statement as shown below:



5. Register allocation

- ⊐ Instructions involving register operands are shorter and faster than those involving operands in memory.

⊐

The use of registers is subdivided into two subproblems :

- ≡ Register allocation – the set of variables that will reside in registers at a point in the program is selected.

Register assignment – the specific register that a variable will reside in is picked. Certain machine requires even-odd register pairs for some operands and results. For example , consider the division instruction of the form :

D x, y where, x – dividend even register in even/odd register pair y – divisor even register holds the remainder odd register holds the quotient

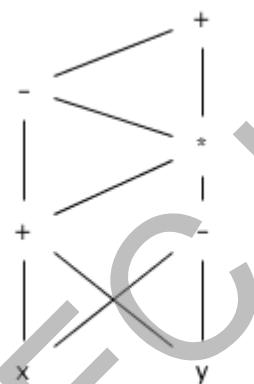
6. Evaluation order

☞ The order in which the computations are performed can affect the efficiency of the target code. Some computation orders require fewer registers to hold intermediate results than others.

17.a) Construct the DAG for the expression.

(4)

$$((x+y)-((x+y)*(x-y)))+((x+y)*(x-y))$$

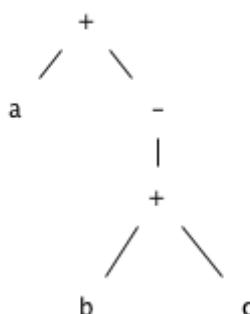


b) Translate the arithmetic expression $a+-(b+c)$ into

(6)

- a)Syntax tree b)Quadruples c)Triples d)Indirect triples

a)Syntax tree



b)Quadruples.

	op	arg1	arg2	result
0	+	b	c	t1
1	minus	t1		t2
2	+	a	t2	t3

c) Triples

	op	arg1	arg2
0	+	b	c
1	minus	(0)	
2	+	a	(1)

d) Indirect triples

	op	arg1	arg2
0	+	b	c
1	minus	(0)	
2	+	a	(1)

	instruction
0	(0)
1	(1)
2	(2)

18. Write a note on the translation of boolean expression.

(10)

Boolean Expressions

compute logical values

change the flow of control

boolean operators are:

and or not

$E \rightarrow E \text{ or } E$

| $E \text{ and } E$

| $\text{not } E$

| (E)

	id relop id
	true
	false

Methods of translation

Evaluate similar to arithmetic expressions

Normally use 1 for true and 0 for false

implement by flow of control

given expression E1 or E2

if E1 evaluates to true

then E1 or E2 evaluates to true

without evaluating E2

Numerical representation

a or b and not c

t1 = not c

t2 = b and t1

t3 = a or t2

relational expression $a < b$ is equivalent to

if $a < b$ then 1 else 0

1. if $a < b$ goto 4.

2. t = 0

3. goto 5

4. t = 1

Syntax directed translation of boolean expressions

$E \rightarrow E_1 \text{ or } E_2$

E.place := newtmp

emit(E.place ':=' E₁.place 'or' E₂.place) $E \rightarrow E_1 \text{ and } E_2$

E.place:= newtmp

emit(E.place ':=' E₁.place 'and' E₂.place) $E \rightarrow \text{not } E_1$

E.place := newtmp

emit(E.place ':=' 'not' E₁.place) $E \rightarrow (E_1)$ E.place = E₁.place $E \rightarrow \text{id1 relop id2}$

E.place := newtmp

emit(if id1.place relop id2.place goto nextstat+3)

emit(E.place = 0)

emit(goto nextstat+2)

emit(E.place = 1)

 $E \rightarrow \text{true}$

E.place := newtmp

emit(E.place = '1')

 $E \rightarrow \text{false}$

E.place := newtmp

emit(E.place = '0')

Control flow translation of boolean expression

$E \rightarrow E_1 \text{ or } E_2$ $E_1.\text{true} := E.\text{true}$
 $E_1.\text{false} := \text{newlabel}$
 $E_2.\text{true} := E.\text{true}$
 $E_2.\text{false} := E.\text{false}$
 $E.\text{code} := E_1.\text{code} \parallel \text{gen}(E_1.\text{false})$
 $\parallel E_2.\text{code}$

$E \rightarrow E_1 \text{ and } E_2$ $E_1.\text{true} := \text{new label}$
 $E_1.\text{false} := E.\text{false}$
 $E_2.\text{true} := E.\text{true}$
 $E_2.\text{false} := E.\text{false}$
 $E.\text{code} := E_1.\text{code} \parallel \text{gen}(E_1.\text{true})$
 $\parallel E_2.\text{code}$

Control flow translation of boolean expression ...

$E \rightarrow \text{not } E_1$ $E_1.\text{true} := E.\text{false}$

$E_1.\text{false} := E.\text{true}$

$E.\text{code} := E_1.\text{code}$

$E \rightarrow (E_1)$

$E_1.\text{true} := E.\text{true}$

$E_1.\text{false} := E.\text{false}$

$E.\text{code} := E_1.\text{code}$

19.Explain optimization of basic blocks. (10)

OPTIMIZATION OF BASIC BLOCKS

There are two types of basic block optimizations. They are :

- = Structure-Preserving Transformations
- = Algebraic Transformations

Structure-Preserving Transformations:

The primary Structure-Preserving Transformation on basic blocks are:

- Common sub-expression elimination
- Dead code elimination
- Renaming of temporary variables
- Interchange of two independent adjacent statements.

Common sub-expression elimination:

Common sub expressions need not be computed over and over again. Instead

they can be computed once and kept in store from where it's referenced when encountered again of course providing the variable values in the expression still remain constant.

Example:

a: =b+c
b: =a-d
c: =b+c
d: =a-d

The 2 nd and 4 th statements compute the same expression: b+c and a-d

Basic block can be transformed to

a: = b+c
b: = a-d
c: = a
d: = b

Dead code elimination:

It's possible that a large amount of dead (useless) code may exist in the program. This might be especially caused when introducing variables and procedures as part of construction or error-correction of a program – once declared and defined, one forgets to remove them in case they serve no purpose. Eliminating these will definitely optimize the code.

Renaming of temporary variables:

A statement $t := b+c$ where t is a temporary name can be changed to $u := b+c$ where u is another temporary name, and change all uses of t to u.

In this we can transform a basic block to its equivalent block called normal-form block.

Interchange of two independent adjacent statements:

Two statements

t 1 :=b+c
t 2 :=x+y

can be interchanged or reordered in its computation in the basic block when value of t 1

does not affect the value of t 2 .

Algebraic Transformations:

Algebraic identities represent another important class of optimizations on basic blocks. This includes simplifying expressions or replacing expensive operation by cheaper ones i.e. reduction in strength. Another class of related optimizations is constant folding. Here we evaluate constant expressions at compile time and replace the constant expressions by their values. Thus the expression $2*3.14$ would be replaced by 6.28.

The relational operators \leq , \geq , $<$, $>$, $+$ and $=$ sometimes generate unexpected common sub expressions.

Associative laws may also be applied to expose common sub expressions. For example, if

the source code has the assignments

a := b+c
e := c+d+b

the following intermediate code may be generated:

a := b+c
t := c+d
e := t+b
⋮

Example:

x:=x+0 can be removed

x:=y**2 can be replaced by a cheaper statement x:=y*y

The compiler writer should examine the language carefully to determine what rearrangements of computations are permitted, since computer arithmetic does not always obey the algebraic identities of mathematics. Thus, a compiler may evaluate $x*y - x*z$ as $x*(y-z)$ but it may not evaluate $a+(b-c)$ as $(a+b)-c$.

20. Discuss the different storage allocation strategies.

(10)

Static allocation

Compiler makes the decision regarding storage allocation by looking only at the program text

Dynamic allocation

Storage allocation decisions are made only while the program is running

Stack allocation

Names local to a procedure are allocated space on a stack

Heap allocation

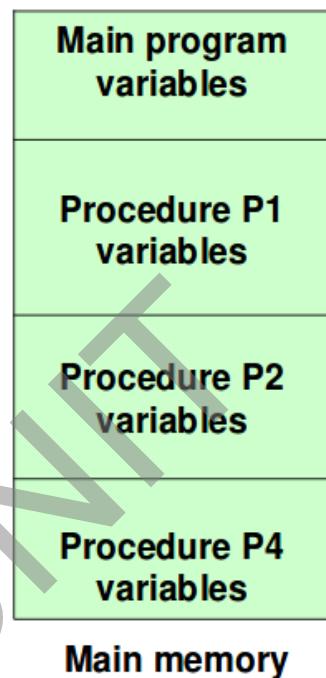
Used for data that may live even after a procedure call returns

Ex: dynamic data structures such as symbol tables

Requires memory manager with garbage collection

Static Data Storage Allocation

- Compiler allocates space for all variables (local and global) of all procedures at compile time
 - No stack/heap allocation; no overheads
 - Ex: Fortran IV and Fortran 77
 - Variable access is fast since addresses are known at compile time
 - No recursion

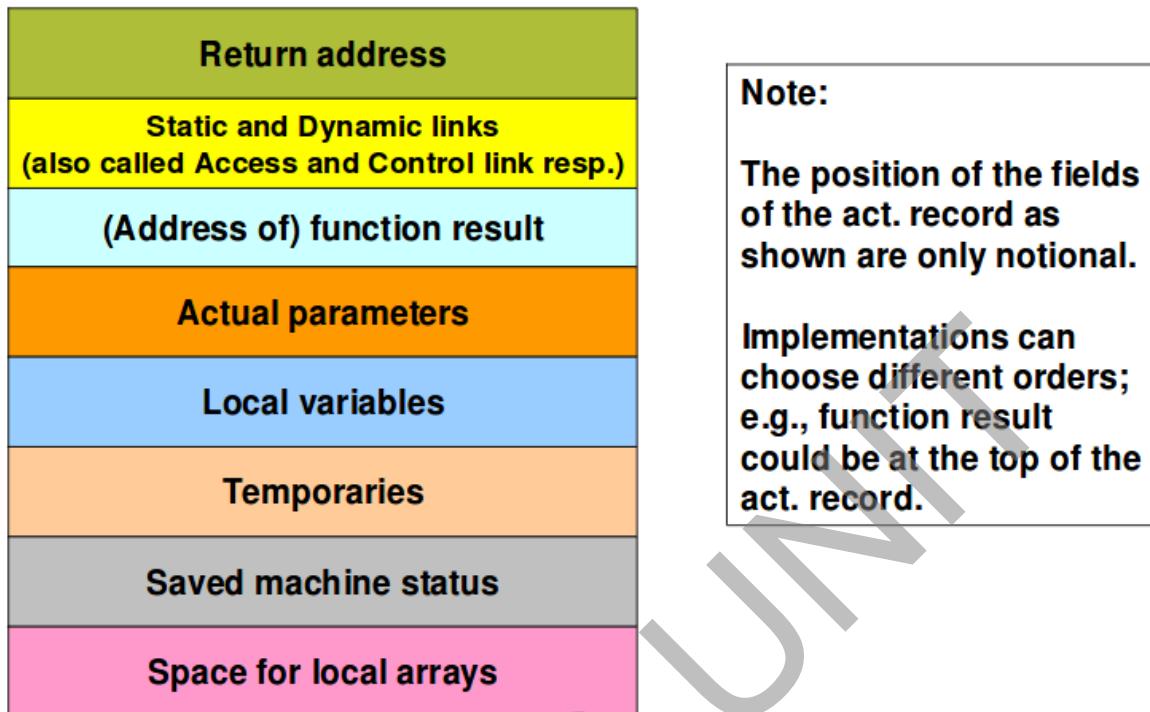


Dynamic Data Storage Allocation

- Compiler allocates space only for global variables at compile time
- Space for variables of procedures will be allocated at run-time
 - Stack/heap allocation
 - Ex: C, C++, Java, Fortran 8/9
 - Variable access is slow (compared to static allocation) since addresses are accessed through the stack/heap pointer
 - Recursion can be implemented



Activation Record Structure



Total Pages: 2

Reg No:.....

Name:.....

6TH SEMESTER B.TECH DEGREE MODEL EXAMINATION, MARCH 2018

COURSE CODE : CS306

COURSE NAME : COMPUTER NETWORKS

MAX MARKS : 100

DURATION : 3 Hours

PART A

Answer all questions. Each carries 3 marks.

1. What is the use of piggybacking
2. Differentiate b/w connection oriented and connectionless services
3. What are Service primitives?
4. Give the classification based on transmission technology.

PART B

Answer any two questions. Each carries 9 marks.

5. Write about the OSI model (reference model).
6. Briefly describe IEEE 802.4.
7. What is HDLC protocol?

PART C

Answer all questions. Each carries 3 marks.

8. What is subnet. Why is subnetting useful?
9. Write the steps in Link state routing process.
10. Difference between distance vector and link state routing.
11. Explain Dijikstra's algorithm.

PART D

Answer any two questions. Each carries 9 marks.

12. Which are the different OSPF packets.
13. Explain tunnelling and fragmentations.
14. Describe different open and closed congestion control and also explain Leaky bucket and Token bucket algorithm briefly

PART E

Answer any four questions. Each carries 10 marks.

15. Compare TCP and UDP protocols
16. Explain DNS addressing scheme
17. Give the 2 mail access protocols.
18. Describe the functions of ICMP.
19. Compare IP v4 and IP v6.
20. Explain the address resolution protocol.

SFI GEC UNIT

CO-OPERATIVE ACADEMY OF PROFESSIONAL EDUCATION

6TH SEMESTER B.TECH DEGREE MODEL EXAMINATION, MARCH 2018

COURSE CODE : CS306

COURSE NAME : COMPUTER NETWORKS

ANSWER KEY

PART A

Answer all questions. Each carries 3 marks.

1. **Piggybacking** is used to improve the efficiency of bidirectional transmission. When a frame is carrying data from A to B, it can also carry control information about frames from B; when a frame is carrying data from B to A, it can also carry control information about frames from A.

With an example.

2. Any 3.

BASIS OF COMPARISON	CONNECTION-ORIENTED SERVICE	CONNECTION-LESS SERVICE
Prior Connection Requirement	Necessary	Not required
Reliability	Ensures reliable transfer of data.	Not guaranteed.
Congestion	Unlikely	Occur likely.
Transferring mode	It can be implemented using circuit switching and virtual circuit.	It is implemented using packet switching.
Lost data retransmission	Feasible	Practically, not possible.
Suitability	Suitable for long and steady communication.	Suitable for bursty Transmission.
Signalling	Used for connection establishment.	There is no concept of signalling.

Packet forwarding	Packets sequentially travel to their destination node and follows the same route.	Packets reach the destination randomly without following the same route.
Delay	There is a delay in transfer of information, but once the connection is established faster delivery can be achieved.	Due to the absence of connection establishment phase, the transmission is faster.
Resource Allocation	Need to be allocated.	No prior allocation of the resource is required.

3. Service primitives

A service is formally specified by a set of primitives (operations) available to a user process to access the service. These primitives tell the service to perform some action or report on an action taken by a peer entity. If the protocol stack is located in the operating system, as it often is, the primitives are normally system calls. These calls cause a trap to kernel mode, which then turns control of the machine over to the operating system to send the necessary packets. The set of primitives available depends on the nature of the service being provided. The primitives for connection-oriented service are different from those of connection less service. As a minimal example of the service primitives that might be provided to implement a reliable byte stream in a client-server environment, consider the primitives listed in Fig

Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

4. Classification based on transmission technology

There are two types of transmission technology that are in widespread use. They are as follows: 1. Broadcast links. 2. Point-to-point links. Broadcast networks have a single communication channel that is shared by all the machines on the network. Short messages, called packets in certain contexts, sent by any machine are received by all the others. An address field within the packet specifies the intended recipient. Upon receiving a packet, a machine checks the address field. If the packet is intended for the receiving machine, that

machine processes the packet; if the packet is intended for some other machine, it is just ignored. As an analogy, consider someone standing at the end of a corridor with many rooms off it and shouting "Watson, come here. I want you." Although the packet, may actually be received (heard) by many people, only Watson responds. The others just ignore it. Another analogy is an airport announcement asking all flight 644 passengers to report to gate 12 for immediate boarding. Broadcast systems generally also allow the possibility of addressing a packet to all destinations by using a special code in the address field.. When a packet with this code is transmitted, it is received and processed by every machine on the network. This mode of operation is called broadcasting. Some broadcast systems also support transmission to a subset of the machines, something known as multicasting. One possible scheme is to reserve one bit to indicate multicasting. The remaining ($n - 1$) address bits can hold a group number. Each machine can "subscribe" to any or all of the groups. When a packet is sent to a certain group, it is delivered to all machines subscribing to that group.

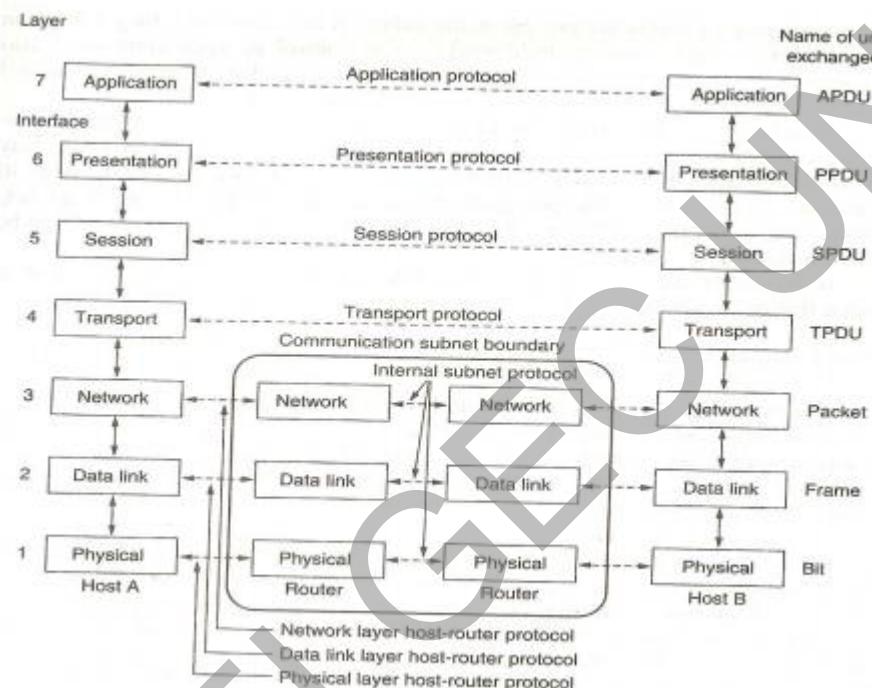
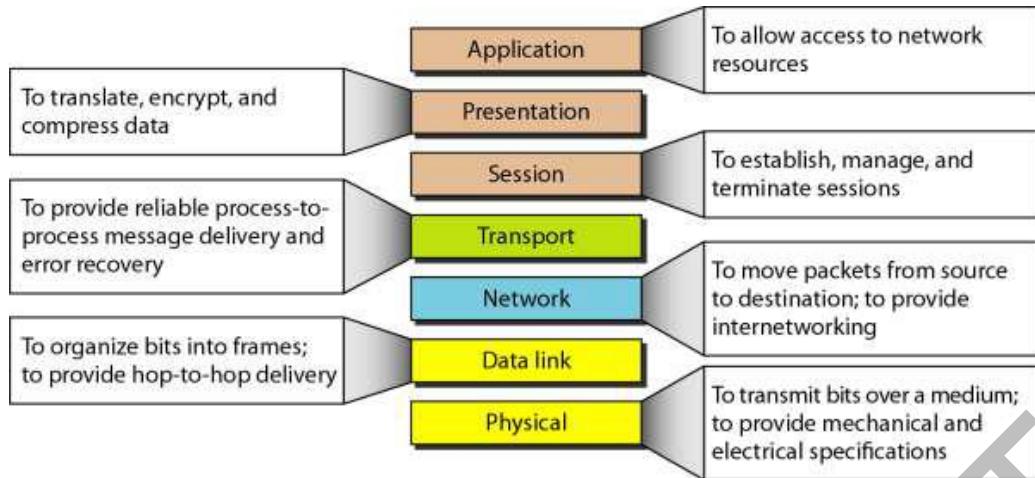
In contrast, point-to-point networks consist of many connections between individual pairs of machines. To go from the source to the destination, a packet on this type of network may have to, first visit one or more intermediate machines. Often multiple routes, of different lengths, are possible, so finding good ones is important in point-to-point networks. As a general rule (although there are many exceptions), smaller, geographically localized networks tend to use broadcasting, whereas larger networks usually are point to-point. Point-to-point transmission with one sender and one receiver is sometimes called uni-casting.

PART B

Answer any two questions. Each carries 9 marks.

5. OSI model (reference model).

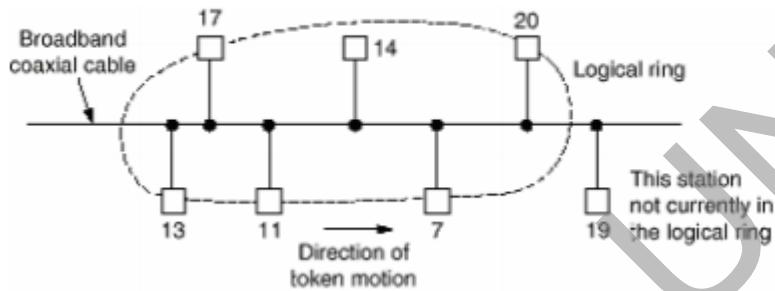
Established in 1947, the International Standards Organization (ISO) is a multinational body dedicated to worldwide agreement on international standards. An ISO standard that covers all aspects of network communications is the Open Systems Interconnection (OSI) model. It was first introduced in the late 1970s. In this section we briefly describe the functions of each layer in the OSI model



6. IEEE has produced 802.4 standards for Token Bus Local Area Networks. IEEE 802.4 Token bus uses a broadband coaxial cable with analog transmission as shown in figure 4.2. Stations physically on a bus, but logically in a ring, with left and right neighbors. The stations on the bus or tree form a logical ring. It determines the logical ring of the physical bus by the numeric value of the address. Each station knows the identity of the stations preceding and following it. The TOKEN regulates the access rights (transmissions), the station received TOKEN can transmission time slot is granted, after transmission, pass the token to the next DTE. When a token is passed to a successor, the sender waits for evidence that it has been transmitted properly (i.e. the successor transmits the next frame). Otherwise it may establish a new successor (the next address to the problematic station).

The access control byte contains a bit called the "token" bit, which is set to 0 if this frame is the token. When a station wants to send, it seizes the token and changes that bit to a 1. A station wishing to transmit waits until it detects a free token passing by. The operations are similar to token ring except more flexible for logical ring definition (can put in essentially

any order) can recover single-node errors better cover phase-coherent, frequency shift keying - uses two frequencies, no phase mismatch. It can avoid much noise by filtering everything but these two frequencies similar to token ring mechanism. It has a complex maintenance scheme, which have to required: Ring initialization, Add/ delete station algorithm, Fault management, Based on predecessor and successor scheme, Recovery and Priority. Token Bus has major functions in MAC are Interface, Access Control, Receive, Transmit, and RRM Regenerative Repeater machine. Under light loads there is some inefficiency. Under heavy loads, the ring functions in a round-robin fashion. Provides a regulated access. Must provide for token maintenance. Token Bus has a special frame called the "token". When you have the token, you can transmit; when you don't have it, you can't.



The access control byte contains a bit called the "token" bit, which is set to 0 if this frame is the token. When a station wants to send, it seizes the token and changes that bit to a 1. A station wishing to transmit waits until it detects a free token passing by. The operations are similar to token ring except more flexible for logical ring definition (can put in essentially any order) can recover single-node errors better cover phase-coherent, frequency shift keying - uses two frequencies, no phase mismatch. It can avoid much noise by filtering everything but these two frequencies similar to token ring mechanism. It has a complex maintenance scheme, which have to required: Ring initialization, Add/ delete station algorithm, Fault management, Based on predecessor and successor scheme, Recovery and Priority. Token Bus has major functions in MAC are Interface, Access Control, Receive, Transmit, and RRM Regenerative Repeater machine. Under light loads there is some inefficiency. Under heavy loads, the ring functions in a round-robin fashion. Provides a regulated access. Must provide for token maintenance.

1. Write difference between static and dynamic routing

7. The HDLC protocol is a general purpose protocol which operates at the data link layer of the OSI reference model. The protocol uses the services of a physical layer, and provides either a best effort or reliable communications path between the transmitter and receiver (i.e. with acknowledged data transfer). The type of service provided depends upon the HDLC mode which is used.

Each piece of data is encapsulated in an HDLC frame by adding a trailer and a header. The header contains an HDLC address and an HDLC control field. The trailer is found at the end of the frame, and contains a Cyclic Redundancy Check (CRC) which detects any errors which may occur during transmission. The frames are separated by HDLC flag sequences which are transmitted between each frame and whenever there is no data to be transmitted.

It is a transmission protocol used at the data link layer (layer 2) of the OSI seven layer model for data communications. The HDLC protocol embeds information in a data frame that allows devices to control data flow and correct errors. HDLC is an ISO standard developed from the Synchronous Data Link Control (SDLC) standard proposed by IBM in the 1970's.

For any HDLC communications session, one station is designated primary and the other secondary. A session can use one of the following connection modes, which determine how the primary and secondary stations interact.

- Normal unbalanced: The secondary station responds only to the primary station.
- Asynchronous: The secondary station can initiate a message.
- Asynchronous balanced: Both stations send and receive over its part of a duplex line. This mode is used for X.25 packet-switching networks.

There are three fundamental types of HDLC frames.

- Information frames, or **I-frames**, transport user data from the network layer. In addition they can also include flow and error control information piggybacked on data.
- Supervisory Frames, or **S-frames**, are used for flow and error control whenever piggybacking is impossible or inappropriate, such as when a station does not have data to send. S-frames **do not** have information fields.
- Unnumbered frames, or **U-frames**, are used for various miscellaneous purposes, including link management. Some U-frames contain an information field, depending on the type.

PART C

Answer all questions. Each carries 3 marks.

8.

Subnet mask is a [mask](#) used to determine what [subnet](#) an [IP address](#) belongs to. An IP address has two components, the network address and the [host](#) address. For example, consider the IP address **150.215.017.009**. Assuming this is part of a Class B network, the first two numbers (**150.215**) represent the Class B network address, and the second two numbers (**017.009**) identify a particular host on this network.

Subnetting enables the network administrator to further divide the host part of the address into two or more subnets. In this case, a part of the host address is reserved to identify the particular subnet. This is easier to see if we show the IP address in binary format.

The full address is:

10010110.11010111.00010001.00001001

The Class B network part is:

10010110.11010111

The host address is:

00010001.00001001

If this network is divided into 14 subnets, however, then the first 4 bits of the host address (0001) are reserved for identifying the subnet.

The subnet mask is the network address plus the bits reserved for identifying the subnetwork -- by convention, the bits for the network address are all set to 1, though it would also work if the bits were set exactly as in the network address. In this case, therefore, the subnet mask would be **11111111.11111111.11110000.00000000**. It's called a *mask* because it can be used to identify the subnet to which an IP address belongs by performing a [bitwise AND operation](#) on the mask and the IP address. The result is the subnetwork address:

Subnet Mask	255.255.240.000	11111111.11111111.11110000.00000000
IP Address	150.215.017.009	10010110.11010111.00010001.00001001
Subnet Address	150.215.016.000	10010110.11010111.00010000.00000000

The subnet address, therefore, is 150.215.016.000.

9. steps in Link-State Routing Process

1. Each router learns about its own links (directly connected networks)
2. Find directly connected neighbors
3. Builds a Link-State Packet (LSP) with the state of each directly connected link
4. Floods the LSP to all neighbors, who stores the received LSPs in a database
5. Each router uses the database to construct a complete map of the network topology
6. Computes the best path to each destination network

b)

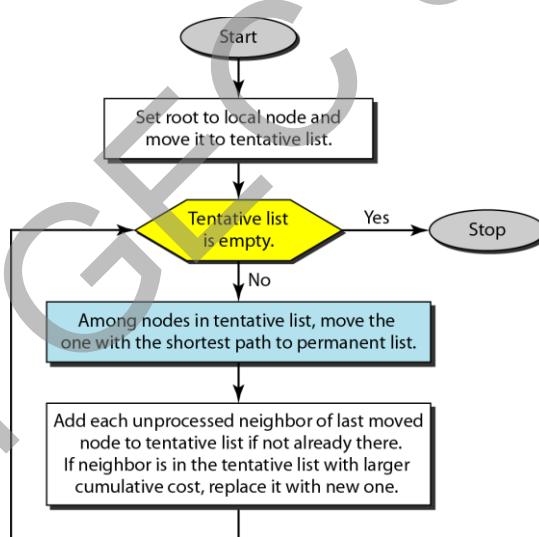
A routing table can be either static or dynamic. A **static routing** table contains information entered manually. A **dynamic routing table** is updated periodically by using one of the dynamic routing protocols such as RIP, OSPF, or BGP

Dijkstras algorithm

10. Difference between DVP and LSP

Distance Vector	Link State
<ul style="list-style-type: none"> View network topology from neighbor's perspective Adds distance vectors from router to router Frequent, periodic updates: Slow convergence Passes copies of routing tables to neighbor routers 	<ul style="list-style-type: none"> Gets common view of entire network topology Calculates the shortest path to other routers Event-triggered updates: Faster convergence Passes link-state routing updates to other routers

11. Dijkstras Algorithm:



PART D

Answer any two questions. Each carries 9 marks.

12. Give the different OSPF Packets

OSPF uses following five packet types to flow routing information between routers:

- 1: hello [every 10 sec] – Hello Builds adjacencies between neighbors

- 2: DBD [Database Descriptor Packet] – DBD for database synchronization between routers
- 3: LSR [Link State Request Packet] – Requests specific link-state records from router to router
- 4: LSU [Link State Update Packet] – Sends specifically requested link-state records
- 5: LSAck [Link State Ack Packet] – Acknowledges the above packet types

LSA types

There are 5 common LSA types:

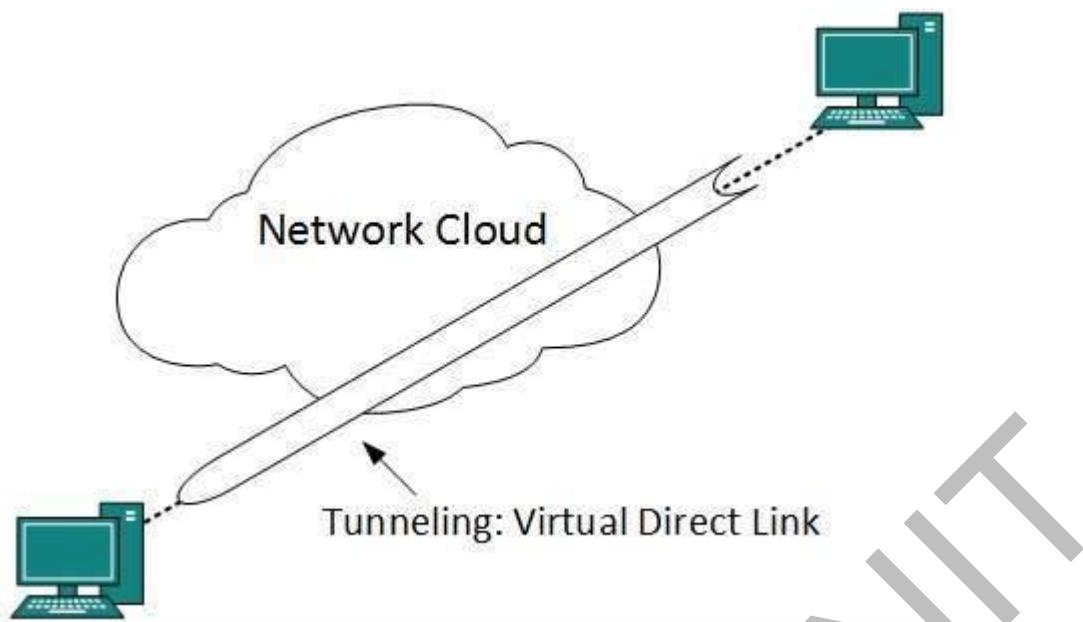
LSA Type	LSA Name	LSA Description
Type 1	Router-LSA	Describes the link status and link cost of a router, generated and advertised in the area the router belongs.
Type 2	Network-LSA	Describes the link status of all routers in the local network segment, generated by DR and advertised in the area to which the DR belongs.
Type 3	Network-Summary-LSA	Describes the routes in a network segment and advertises the routes to the related non-totally STUB or NSSA area.
Type 4	ASBR-Summary-LSA	Describes routes to an ASBR, generated by an ABR and advertised in the related areas, except the area to which the ASBR belongs.
Type 5	AS-External-LSA	Describes routes to a destination outside the AS. Generated by an ASBR and advertised in all areas, except stub areas and Not-So-Stubby Areas (NSSA).

13.

Tunneling

If they are two geographically separate networks, which want to communicate with each other, they may deploy a dedicated line between or they have to pass their data through intermediate networks.

Tunneling is a mechanism by which two or more same networks communicate with each other, by passing intermediate networking complexities. Tunneling is configured at both ends.



When the data enters from one end of Tunnel, it is tagged. This tagged data is then routed inside the intermediate or transit network to reach the other end of Tunnel. When data exists the Tunnel its tag is removed and delivered to the other part of the network.

Both ends seem as if they are directly connected and tagging makes data travel through transit network without any modifications.

Packet Fragmentation

Most Ethernet segments have their maximum transmission unit (MTU) fixed to 1500 bytes. A data packet can have more or less packet length depending upon the application. Devices in the transit path also have their hardware and software capabilities which tell what amount of data that device can handle and what size of packet it can process.

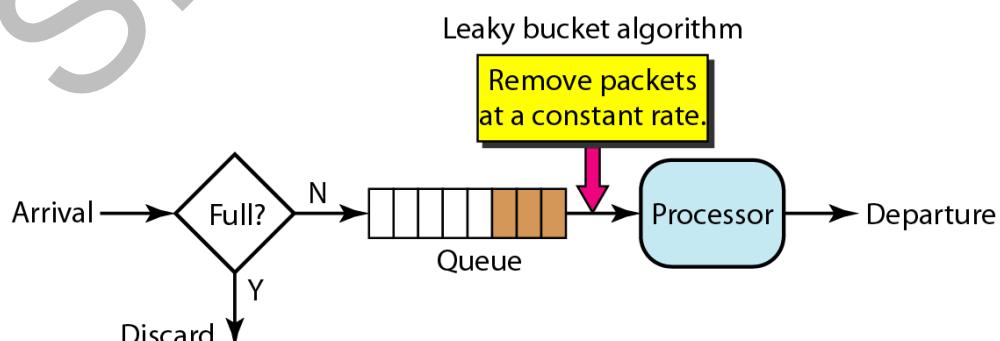
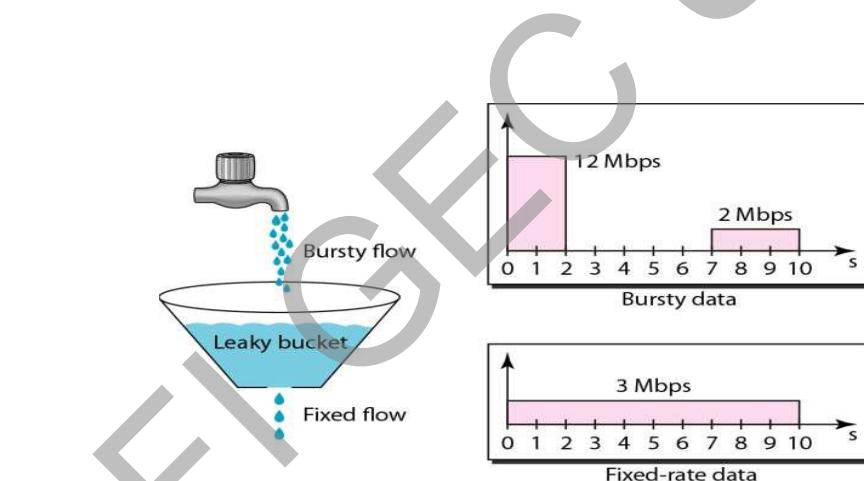
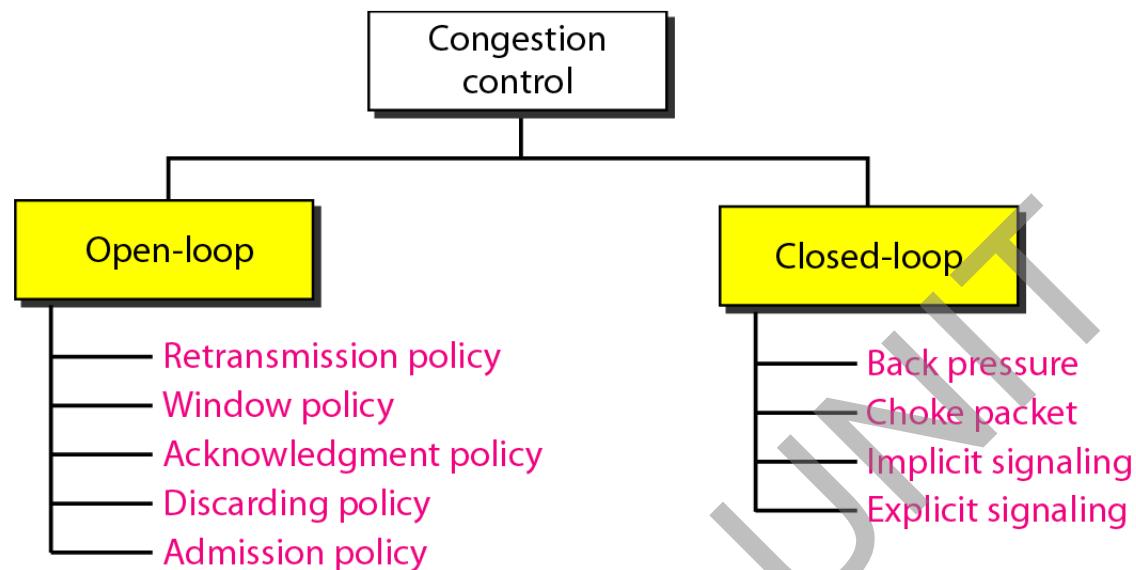
If the data packet size is less than or equal to the size of packet the transit network can handle, it is processed neutrally. If the packet is larger, it is broken into smaller pieces and then forwarded. This is called packet fragmentation. Each fragment contains the same destination and source address and routed through transit path easily. At the receiving end it is assembled again.

If a packet with DF (don't fragment) bit set to 1 comes to a router which can not handle the packet because of its length, the packet is dropped.

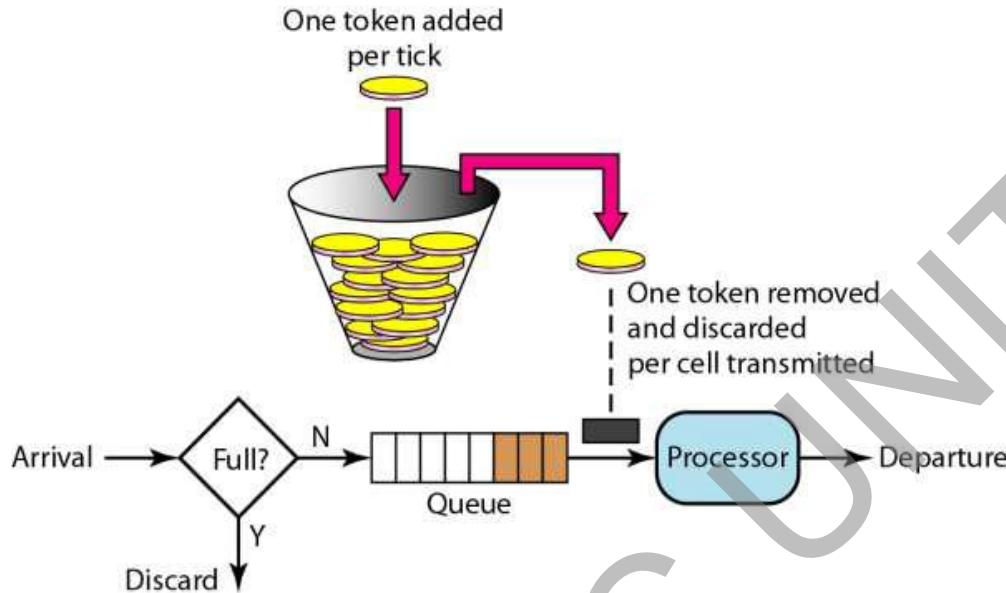
When a packet is received by a router has its MF (more fragments) bit set to 1, the router then knows that it is a fragmented packet and parts of the original packet is on the way.

If packet is fragmented too small, the overhead is increases. If the packet is fragmented too large, intermediate router may not be able to process it and it might get dropped.

14. Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. In general, we can divide congestion control mechanisms into two broad categories: open-loop congestion control (prevention) and closed-loop congestion control (removal).



A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.

Token bucket**PART E**

Answer any four questions. Each carries 10 marks.

15.

Compare TCP and UDP protocols

There are two types of Internet Protocol (IP) traffic. They are **TCP** or **Transmission Control Protocol** and **UDP** or **User Datagram Protocol**. TCP is connection oriented – once a connection is established, data can be sent bidirectional. UDP is a simpler, connectionless Internet protocol. Multiple messages are sent as packets in chunks using UDP.

Acronym for	Transmission Control Protocol	User Datagram Protocol or Universal Datagram Protocol
Connection	TCP is a connection-oriented protocol.	UDP is a connectionless protocol.
Function	As a message makes its way across the <u>internet</u> from one computer to another. This is connection based.	UDP is also a protocol used in message transport or transfer. This is not connection based which means that one program can send a load of packets

		to another and that would be the end of the relationship.
Usage	TCP is suited for applications that require high reliability, and transmission time is relatively less critical.	UDP is suitable for applications that need fast, efficient transmission, such as games. UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients.
Use by other protocols	HTTP, HTTPs, FTP, SMTP, Telnet	DNS, DHCP, TFTP, SNMP, RIP, VOIP.
Ordering of data packets	TCP rearranges <u>data</u> packets in the order specified.	UDP has no inherent order as all packets are independent of each other. If ordering is required, it has to be managed by the application layer.
Speed of transfer	The speed for TCP is slower than UDP.	UDP is faster because error recovery is not attempted. It is a "best effort" protocol.
Reliability	There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent.	There is no guarantee that the messages or packets sent would reach at all.
Header Size	TCP header size is 20 bytes	UDP Header size is 8 bytes.
Common Header Fields	Source port, Destination port, Check Sum	Source port, Destination port, Check Sum
Streaming of data	Data is read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries.	Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent.
Weight	TCP is heavy-weight. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and	UDP is lightweight. There is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP.

	congestion control.	
Data Flow Control	TCP does Flow Control. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.	UDP does not have an option for flow control
Error Checking	TCP does error checking and error recovery. Erroneous packets are retransmitted from the source to the destination.	UDP does error checking but simply discards erroneous packets. Error recovery is not attempted.
Fields	1. Sequence Number, 2. AcK number, 3. Data offset, 4. Reserved, 5. Control bit, 6. Window, 7. Urgent Pointer 8. Options, 9. Padding, 10. Check Sum, 11. Source port, 12. Destination port	1. Length, 2. Source port, 3. Destination port, 4. Check Sum
Acknowledgement	Acknowledgement segments	No Acknowledgment
Handshake	SYN, SYN-ACK, ACK	No handshake (connectionless protocol)

16. Explain DNS addressing scheme

Domain Name System helps to resolve the host name to an address. It uses a hierarchical naming scheme and distributed database of IP addresses and associated names

IP Address

IP address is a unique logical address assigned to a machine over the network. An IP address exhibits the following properties:

- IP address is the unique address assigned to each host present on Internet.
- IP address is 32 bits (4 bytes) long.
- IP address consists of two components: **network component** and **host component**.
- Each of the 4 bytes is represented by a number from 0 to 255, separated with dots.
For example 137.170.4.124

IP address is 32-bit number while on the other hand domain names are easy to remember names. For example, when we enter an email address we always enter a symbolic string such as webmaster@tutorialspoint.com.

Uniform Resource Locator (URL)

Uniform Resource Locator (URL) refers to a web address which uniquely identifies a document over the internet.

For example, www.tutorialspoint.com/internet_technology/index.html is an URL to the index.html which is stored on tutorialspoint web server under internet_technology directory.

URL Types

There are two forms of URL as listed below:

1. Absolute URL
2. Relative URL

ABSOLUTE URL

Absolute URL is a complete address of a resource on the web. This completed address comprises of protocol used, server name, path name and file name.

For example [http:// www.tutorialspoint.com / internet_technology /index.htm](http://www.tutorialspoint.com/internet_technology/index.htm). where:

- **http** is the protocol.
- **tutorialspoint.com** is the server name.
- **index.htm** is the file name.

The protocol part tells the web browser how to handle the file. Similarly we have some other protocols also that can be used to create URL are:

- FTP
- https
- Gopher
- mailto
- news

RELATIVE URL

Relative URL is a partial address of a webpage. Unlike absolute URL, the protocol and server part are omitted from relative URL.

Relative URLs are used for internal links i.e. to create links to file that are part of same website as the WebPages on which you are placing the link.

Domain Name System Architecture

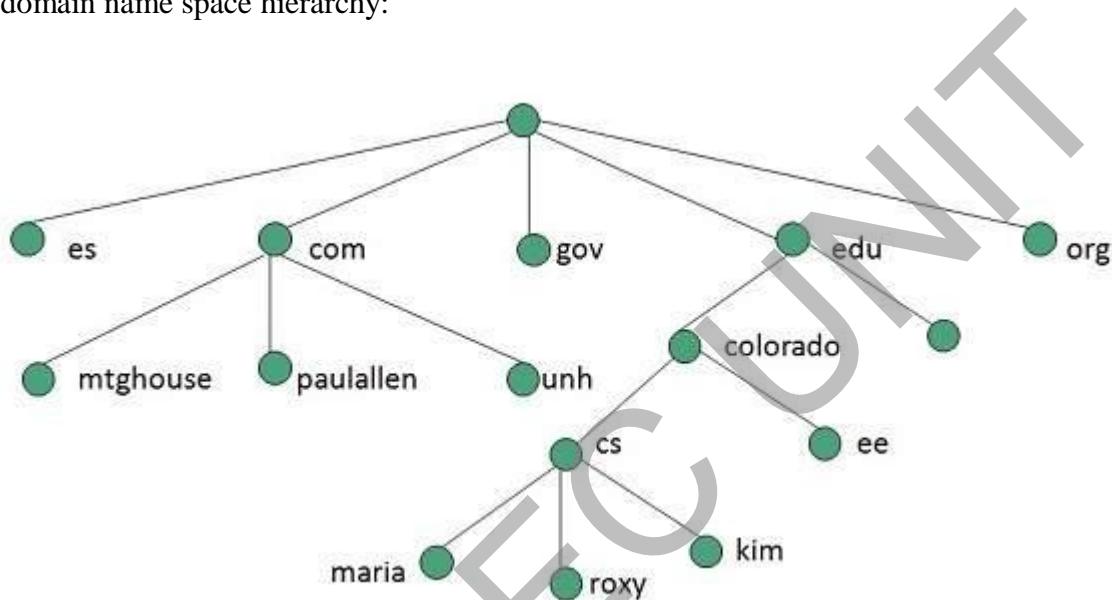
The Domain name system comprises of **Domain Names**, **Domain Name Space**, **Name Server** that have been described below:

Domain Names

Domain Name is a symbolic string associated with an IP address. There are several domain names available; some of them are generic such as **com**, **edu**, **gov**, **net** etc, while some country level domain names such as **au**, **in**, **za**, **usetc**.

Domain Name Space

The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127), with a root at the top. The following diagram shows the domain name space hierarchy:



In the above diagram each subtree represents a domain. Each domain can be partitioned into sub domains and these can be further partitioned and so on.

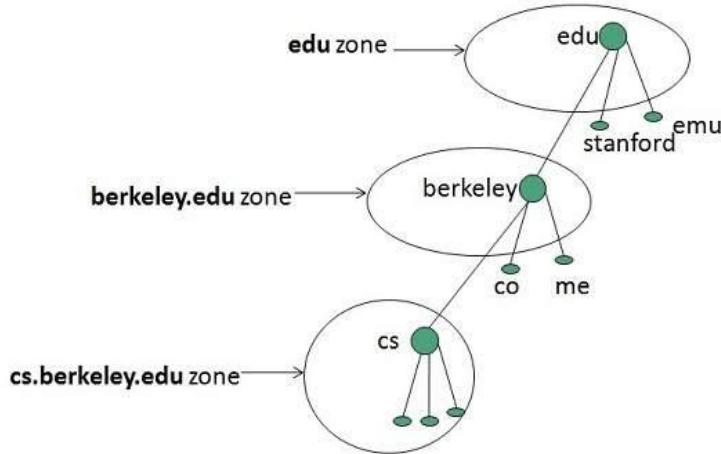
Name Server

Name server contains the DNS database. This database comprises of various names and their corresponding IP addresses. Since it is not possible for a single server to maintain entire DNS database, therefore, the information is distributed among many DNS servers.

- Hierarchy of server is same as hierarchy of names.
- The entire name space is divided into the zones

Zones

Zone is collection of nodes (sub domains) under the main domain. The server maintains a database called zone file for every zone.



If the domain is not further divided into sub domains then domain and zone refers to the same thing.

The information about the nodes in the sub domain is stored in the servers at the lower levels however; the original server keeps reference to these lower levels of servers.

TYPES OF NAME SERVERS

Following are the three categories of Name Servers that manages the entire Domain Name System:

1. Root Server
2. Primary Server
3. Secondary Server

ROOT SERVER

Root Server is the top level server which consists of the entire DNS tree. It does not contain the information about domains but delegates the authority to the other server

PRIMARY SERVERS

Primary Server stores a file about its zone. It has authority to create, maintain, and update the zone file.

SECONDARY SERVER

Secondary Server transfers complete information about a zone from another server which may be primary or secondary server. The secondary server does not have authority to create or update a zone file.

DNS Working

DNS translates the domain name into IP address automatically. Following steps will take you through the steps included in domain resolution process:

- When we type **www.tutorialspoint.com** into the browser, it asks the local DNS Server for its IP address.

Here the local DNS is at ISP end.

- When the local DNS does not find the IP address of requested domain name, it forwards the request to the root DNS server and again enquires about IP address of it.
- The root DNS server replies with delegation that **I do not know the IP address of www.tutorialspoint.com but know the IP address of DNS Server.**
- The local DNS server then asks the com DNS Server the same question.
- The **com** DNS Server replies the same that it does not know the IP address of www.tutorialspoint.com but knows the address of tutorialspoint.com.
- Then the local DNS asks the tutorialspoint.com DNS server the same question.
- Then tutorialspoint.com DNS server replies with IP address of www.tutorialspoint.com.
- Now, the local DNS sends the IP address of www.tutorialspoint.com to the computer that sends the request.

17. Two mail access protocols

E-mail Protocols are set of rules that help the client to properly transmit the information to or from the mail server. various protocols such as **SMTP, POP, and IMAP.**

Basically, a protocol is about a standard method used at each end of a communication channel, in order to properly transmit information. In order to deal with your email you must use a mail client to access a mail server. The mail client and mail server can exchange information with each other using a variety of protocols.

:: IMAP Protocol:

IMAP (Internet Message Access Protocol) – Is a standard protocol for accessing e-mail from your local server. IMAP is a client/server protocol in which e-mail is received and held for you by your Internet server. As this requires only a small data transfer this works well even over a slow connection such as a modem. Only if you request to read a specific email message will it be downloaded from the server. You can also create and manipulate folders or mailboxes on the server, delete messages etc.

:: POP3 Protocol:

The **POP (Post Office Protocol 3)** protocol provides a simple, standardized way for users to access mailboxes and download messages to their computers.

When using the POP protocol all your eMail messages will be downloaded from the mail server to your local computer. You can choose to leave copies of your eMails on the server as well. The advantage is that once your messages are downloaded you can cut the internet connection and read your eMail at your leisure without incurring further communication costs. On the other hand you might have transferred a lot of message (including spam or viruses) in which you are not at all interested at this point.

:: SMTP Protocol:

The **SMTP (Simple Mail Transfer Protocol)** protocol is used by the Mail Transfer Agent (MTA) to deliver your eMail to the recipient's mail server. The SMTP protocol can only be used to send emails, not to receive them. Depending on your network / ISP settings, you may only be able to use the SMTP protocol under certain conditions

:: HTTP Protocol:

The **HTTP protocol** is not a protocol dedicated for email communications, but it can be used for accessing your mailbox. Also called web based email, this protocol can be used to compose or retrieve emails from an your account. Hotmail is a good example of using HTTP as an email protocol.

SMTP

SMTP stands for **Simple Mail Transfer Protocol**. It was first proposed in 1982. It is a standard protocol used for sending e-mail efficiently and reliably over the internet.

Key Points:

- SMTP is application level protocol.
- SMTP is connection oriented protocol.
- SMTP is text based protocol.
- It handles exchange of messages between e-mail servers over TCP/IP network.
- Apart from transferring e-mail, SMPT also provides notification regarding incoming mail.
- When you send e-mail, your e-mail client sends it to your e-mail server which further contacts the recipient mail server using SMTP client.
- These SMTP commands specify the sender's and receiver's e-mail address, along with the message to be send.

- The exchange of commands between servers is carried out without intervention of any user.
- In case, message cannot be delivered, an error report is sent to the sender which makes SMTP a reliable protocol.

SMTP Commands

The following table describes some of the SMTP commands:

S.N.	Command Description
1	HELLO This command initiates the SMTP conversation.
2	EHELLO This is an alternative command to initiate the conversation. ESMTP indicates that the sender server wants to use extended SMTP protocol.
3	MAIL FROM This indicates the sender's address.
4	RCPT TO It identifies the recipient of the mail. In order to deliver similar message to multiple users this command can be repeated multiple times.
5	SIZE This command let the server know the size of attached message in bytes.
6	DATA The DATA command signifies that a stream of data will follow. Here stream of data refers to the body of the message.
7	QUIT This command is used to terminate the SMTP connection.
8	VERFY This command is used by the receiving server in order to verify whether the given username is valid or not.
9	EXPN It is same as VRFY, except it will list all the users name when it used with a

	distribution list.
--	--------------------

IMAP

IMAP stands for **Internet Mail Access Protocol**. It was first proposed in 1986. There exist five versions of IMAP as follows:

1. Original IMAP
2. IMAP2
3. IMAP3
4. IMAP2bis
5. IMAP4

Key Points:

- IMAP allows the client program to manipulate the e-mail message on the server without downloading them on the local computer.
- The e-mail is held and maintained by the remote server.
- It enables us to take any action such as downloading, delete the mail without reading the mail. It enables us to create, manipulate and delete remote message folders called mail boxes.
- IMAP enables the users to search the e-mails.
- It allows concurrent access to multiple mailboxes on multiple mail servers.

IMAP Commands

The following table describes some of the IMAP commands:

S.N.	Command Description
1	IMAP_LOGIN This command opens the connection.
2	CAPABILITY This command requests for listing the capabilities that the server supports.
3	NOOP This command is used as a periodic poll for new messages or message status updates during a period of inactivity.

4	SELECT This command helps to select a mailbox to access the messages.
5	EXAMINE It is same as SELECT command except no change to the mailbox is permitted.
6	CREATE It is used to create mailbox with a specified name.
7	DELETE It is used to permanently delete a mailbox with a given name.
8	RENAME It is used to change the name of a mailbox.
9	LOGOUT This command informs the server that client is done with the session. The server must send BYE untagged response before the OK response and then close the network connection.

POP

POP stands for Post Office Protocol. It is generally used to support a single client. There are several versions of POP but the POP 3 is the current standard.

Key Points

- POP is an application layer internet standard protocol.
- Since POP supports offline access to the messages, thus requires less internet usage time.
- POP does not allow search facility.
- In order to access the messages, it is necessary to download them.
- It allows only one mailbox to be created on server.
- It is not suitable for accessing non mail data.
- POP commands are generally abbreviated into codes of three or four letters. Eg. STAT.

POP Commands

The following table describes some of the POP commands:

S.N.	Command Description
1	LOGIN This command opens the connection.
2	STAT It is used to display number of messages currently in the mailbox.
3	LIST It is used to get the summary of messages where each message summary is shown.
4	RETR This command helps to select a mailbox to access the messages.
5	DELE It is used to delete a message.
6	RSET It is used to reset the session to its initial state.
7	QUIT It is used to log off the session.

18. Describe the functions of ICMP

The Internet Control Message Protocol (ICMP) is one of the main protocols of the internet protocol suite. It is used by network devices, like routers, to send error messages indicating, for example, that a requested service is not available or that a host or router could not be reached. ICMP can also be used to relay query messages.

- i. The Internet Control Message Protocol (ICMP) is a mechanism used by hosts and gateways to send notification of datagram problems back to the sender. ICMP sends query and error reporting messages.
- ii. The Internet Control Message Protocol (ICMP) has been designed to compensate for the two deficiencies: lack of error control and lack of assistance mechanisms. It is a companion to the IP protocol.

Types of Messages: ICMP messages are divided into two broad categories: error-reporting messages and query messages. The error-reporting messages report problems that a router or a host (destination) may encounter when it processes an IP packet. The query messages, which occur in pairs, help a host or a network manager get specific information from a router or another host.

I) Error Reporting:

- One of the main responsibilities of ICMP is to report errors. Although technology has produced increasingly reliable transmission media, errors still exist and must be handled.
- IP is an unreliable protocol. This means that error checking and error control are not a concern of IP.
- ICMP was designed, in part, to compensate for this shortcoming. However, ICMP does not correct errors-it simply reports them. Error correction is left to the higher-level protocols.
- Error messages are always sent to the original source because the only information available in the datagram about the route is the source and destination IP addresses.
- ICMP uses the source IP address to send the error message to the source (originator) of the datagram.
- Five types of errors are handled: destination unreachable, source quench, time exceeded, parameter problems, and redirection (see figure1).



Fig1: Error-reporting messages

- **Destination Unreachable:** When a router cannot route a datagram or a host cannot deliver a datagram, the datagram is discarded and the router or the host sends a destination-unreachable message back to the source host that initiated the datagram.
- **Source Quench:** The source-quench message in ICMP was designed to add a kind of flow control to the IP. When a router or host discards a datagram due to congestion, it sends a source-quench message to the sender of the datagram. This message has two purposes. First, it informs the source that the datagram has been discarded. Second, it warns the source that there is congestion somewhere in the path and that the source should slow down (quench) the sending process.
- **Time Exceeded:** When the time-to-live value reaches 0, after decrementing, the router discards the datagram. However, when the datagram is discarded, a time-exceeded message must be sent by the router to the original source. Second, a time-exceeded message is also generated when not all fragments that make up a message arrive at the destination host within a certain time limit.
- **Parameter Problem:** Any ambiguity in the header part of a datagram can create serious problems as the datagram travels through the Internet. If a router or the destination host discovers an ambiguous or missing value in any field of the datagram, it discards the datagram and sends a parameter-problem message back to the source.
- **Redirection:** This concept of redirection is shown in Figure 2. Host A wants to send a datagram to host B. Router R2 is obviously the most efficient routing choice, but host A did not choose router R2. The datagram goes to R1 instead. Router R1, after consulting its table, finds that the packet should have gone to R2. It sends the packet to R2 and, at the same time, sends a redirection message to host A. Host A's routing table can now be updated.

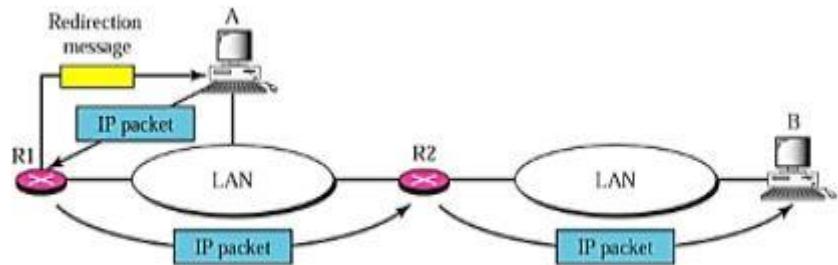


Fig2: Redirection concept

II) Query:

In addition to error reporting, ICMP can diagnose some network problems. This is accomplished through the query messages, a group of four different pairs of messages, as shown in Figure3

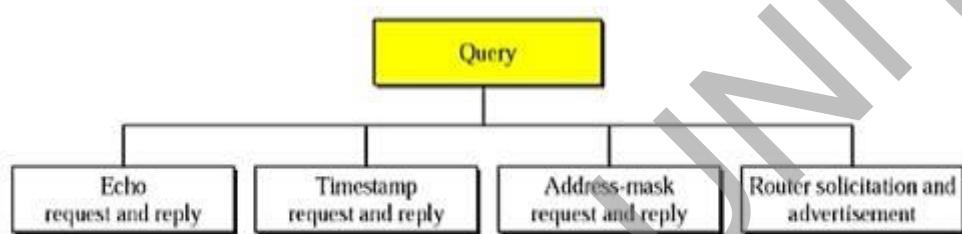


Fig3: Query messages

- Echo Request and Reply:** The echo-request and echo-reply messages are designed for diagnostic purposes. Network managers and users utilize this pair of messages to identify network problems.
- Timestamp Request and Reply:** Two machines (hosts or routers) can use the timestamp request and timestamp reply messages to determine the round-trip time needed for an IP datagram to travel between them. It can also be used to synchronize the clocks in two machines.
- Address-Mask Request and Reply:** If the host knows the address of the router, it sends the request directly to the router. If it does not know, it broadcasts the message. The router receiving the address-mask-request message responds with an address-mask-reply message, providing the necessary mask for the host.
- Router Solicitation and Advertisement:** A host that wants to send data to a host on another network needs to know the address of routers connected to its own network. Also, the host must know if the routers are alive and functioning. The router-solicitation and router-advertisement messages can help in this situation.
- Checksum:** In ICMP the checksum is calculated over the entire message (header and data).

19. compare IP v4 and IP v6

The following table lists the important differences between IPv4 and IPv6.

IPv4	IPv6
------	------

<u>IPv4 addresses</u> are 32 bit length.	<u>IPv6 addresses</u> are 128 bit length.
<u>IPv4 addresses</u> are <u>binary numbers</u> represented in decimals.	<u>IPv6 addresses</u> are <u>binary numbers</u> represented in <u>hexadecimals</u> .
<u>IPSec</u> support is only optional.	Inbuilt <u>IPSec</u> support.
<u>Fragmentation</u> is done by sender and forwarding routers.	<u>Fragmentation</u> is done only by sender.
No packet flow identification.	Packet flow identification is available within the <u>IPv6 header</u> using the <u>Flow Label</u> field.
<u>Checksum field</u> is available in <u>IPv4 header</u>	No checksum field in <u>IPv6 header</u> .
<u>Options fields</u> are available in <u>IPv4 header</u> .	No option fields, but <u>IPv6 Extension headers</u> are available.
<u>Address Resolution Protocol (ARP)</u> is available to map <u>IPv4 addresses</u> to <u>MAC addresses</u> .	<u>Address Resolution Protocol (ARP)</u> is replaced with a function of <u>Neighbor Discovery Protocol (NDP)</u> .
Internet Group Management Protocol (IGMP) is used to manage multicast group membership.	IGMP is replaced with Multicast Listener Discovery (MLD) messages.
<u>Broadcast messages</u> are available.	<u>Broadcast messages</u> are not available. Instead a link-local scope "All nodes" <u>multicast IPv6 address</u> (FF02::1) is used for broadcast similar functionality.
Manual configuration (Static) of <u>IPv4 addresses</u> or DHCP (Dynamic configuration) is required to configure <u>IPv4 addresses</u> .	Auto-configuration of addresses is available.

20. What Is ARP

The address resolution protocol (arp) is a protocol used by the Internet Protocol (IP), specifically IPv4, to map IP network addresses to the hardware addresses used by a data link

protocol. The protocol operates below the network layer as a part of the interface between the OSI network and OSI link layer. It is used when IPv4 is used over Ethernet.

The term address resolution refers to the process of finding an address of a computer in a network. The address is "resolved" using a protocol in which a piece of information is sent by a client process executing on the local computer to a server process executing on a remote computer. The information received by the server allows the server to uniquely identify the network system for which the address was required and therefore to provide the required address. The address resolution procedure is completed when the client receives a response from the server containing the required address.

There are four types of arp messages that may be sent by the arp protocol. These are identified by four values in the "operation" field of an arp message. The types of message are:

1. ARP request
2. ARP reply
3. RARP request
4. RARP reply

The format of an arp message is shown below:

0	8	15_16	31
Hardware Type		Protocol Type	
HLEN	PLEN	Operation	
Sender HA (octets 0-3)			
Sender HA (octets 4-5)		Sender IP (octets 0-1)	
Sender IP (octets 2-3)		Target HA (octets 0-1)	
Target HA (octets 2-5)			
Target IP (octets 0-3)			

Format of an arp message used to resolve the remote [MAC Hardware Address](#) (HA)

ARP working

When an incoming packet destined for a host machine on a particular local area network arrives at a gateway, the gateway asks the ARP program to find a physical host or MAC address that matches the IP address. The ARP program looks in the ARP cache and, if it finds the address, provides it so that the packet can be converted to the right packet length and format and sent to the machine. If no entry is found for the IP address, ARP broadcasts a request packet in a special format to all the machines on the LAN to see if one machine knows that it has that IP address associated with it. A machine that recognizes the IP address as its own returns a reply so indicating. ARP updates the ARP cache for future reference and then sends the packet to the MAC address that replied.

MODEL QUESTION PAPER
COLLEGE OF ENGINEERING ARANMULA
SIXTH SEMESTER B.TECH DEGREE EXAMINATION, MARCH 2018
Course Code: CS306
Course Name: Computer Networks

Max. Marks: 100

Duration: 3 Hours

PART A

Answer all questions, each carries 3 marks.

- | | | Marks |
|---|---|-------|
| 1 | Enumerate any four reasons for using layered protocols? | (3) |
| 2 | Explain about WAN? | (3) |
| 3 | With neat sketch explain the basic concept involved in Elementary Protocol:stop and wait? | (3) |
| 4 | Define Gigabit Ethernet ? | (3) |

PART B

Answer any two full questions, each carries 9 marks.

- | | | |
|---|--|------------|
| 5 | a) Explain TCP/IP reference model in detail? | (9) |
| 6 | a) Discuss about the configuration and control fields of HDLC?
b) Describe the services provided by PPP protocol. Also, list some services which does PPP does not provide. ? | (5)
(4) |
| 7 | a) What is layered architecture? Explain design issues for the layers?
b) Discuss briefly about the MAC layers in the 802.11 standard? | (5)
(4) |

PART C

Answer all questions, each carries 3 marks.

- | | | |
|----|---|-----|
| 8 | Explain the concept involved in Flooding algorithm? | (3) |
| 9 | Discuss about routing for mobile hosts? | (3) |
| 10 | Define Congestion. What is TCP/IP congestion avoidance? | (3) |
| 11 | Compare between IPV4 and IPV6? | (3) |

PART D

Answer any two full questions, each carries 9 marks.

- | | | |
|----|--|------------|
| 12 | Explain Distance Vector routing algorithm with an example? | (9) |
| 13 | Mention different steps of link state routing protocol? | (9) |
| 14 | a) Explain leaky bucket and token bucket algorithms?
b) What is the format of IPv4 header? Describe the significance of each field? | (5)
(4) |

PART E

Answer any four full questions, each carries 10 marks.

- | | | |
|----|---|------|
| 15 | a) Differentiate between
i)ARP and RARP
ii)TCP and UDP | (5) |
| b) | Write short on IPV6? | (5) |
| 16 | How DNS service maps domain names to IP addresses? | (10) |
| 17 | Write about electronic mail in detail? | (10) |
| 18 | Explain in detail different types of ICMP messages? | (10) |
| 19 | Discuss the role of BGP in Exterior routing ? | (10) |
| 20 | Explain in detail about the Client and Server in World Wide Web.? | (10) |

Answer Key

PART A

1. The layered protocol is defined as the protocol that has been separated into layered pattern to make the tasks as simple

- It simplifies the design process as the functions of each layers and their interactions are well defined.
- The layered architecture provides flexibility to modify and develop network services.
- The number of layers, name of layers and the tasks assigned to them may change from network to network. But for all the networks, always the lower layer offers certain services to its upper layer
- The concept of layered architecture redefines the way of convincing networks.
- This leads to a considerable cost savings and managerial benefits.
- Addition of new services and management of network infrastructure become easy.
- Due to segmentation, it is possible to break complex problems into smaller and more manageable pieces.
- Logical segmentation helps development taking place by different terms.

2. A wide area network, or WAN, spans a large geographical area, often a country or continent. It contains a collection of machines intended for running user (i.e., application) programs. We will follow traditional usage and call these machines hosts. The hosts are connected by a communication subnet, or just subnet for short. The hosts are owned by the customers (e.g., people's personal computers), whereas the communication subnet is typically owned and operated by a telephone company or Internet service provider. The job of the subnet is to carry messages from host to host, just as the telephone system carries words from speaker to listener. Separation of the pure communication aspects of the network (the subnet) from the application aspects (the hosts), greatly simplifies the complete network design.

In most wide area networks, the subnet consists of two distinct components: transmission lines and switching elements. Transmission lines move bits between machines. They can be made of copper wire, optical fiber, or even radio links. Switching elements are specialized computers that connect three or more transmission lines. When data arrive on an incoming line, the switching element must choose an outgoing line on which to forward them.

A short comment about the term "subnet" is in order here. Originally, its only meaning was the collection of routers and communication lines that moved packets from the source host to the destination host. However, some years later, it also acquired a second meaning in conjunction with network addressing . Unfortunately, no widely-used alternative exists for its initial meaning, so with some hesitation we will use it in both senses. From the context, it will always be clear which is meant.

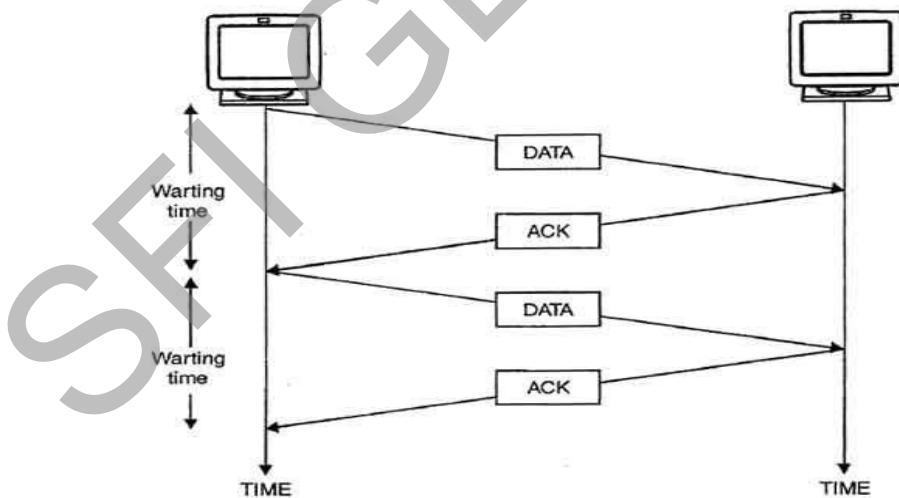
In most WANs, the network contains numerous transmission lines, each one connecting a pair of routers. If two routers that do not share a transmission line wish to communicate, they must do this indirectly, via other routers. When a packet is sent from one router to another via one or more intermediate routers, the packet is received at each intermediate router in its entirety, stored there until the required output line is free, and then forwarded. A subnet organized according to this principle is called a store-and-forward or

packet-switched subnet. Nearly all wide area networks (except those using satellites) have store-and-forward subnets. When the packets are small and all the same size, they are often called cells.

The principle of a packet-switched WAN is so important that it is worth devoting a few more words to it. Generally, when a process on some host has a message to be sent to a process on some other host, the sending host first cuts the message into packets, each one bearing its number in the sequence. These packets are then injected into the network one at a time in quick succession. The packets are transported individually over the network and deposited at the receiving host, where they are reassembled into the original message and delivered to the receiving process.

Not all WANs are packet switched. A second possibility for a WAN is a satellite system. Each router has an antenna through which it can send and receive. All routers can hear the output from the satellite, and in some cases they can also hear the upward transmissions of their fellow routers to the satellite as well. Sometimes the routers are connected to a substantial point-to-point subnet, with only some of them having a satellite antenna. Satellite networks are inherently broadcast and are most useful when the broadcast property is important.

3. In this method of flow control, the sender sends a single frame to receiver & waits for an acknowledgment. The next frame is sent by sender only when acknowledgment of previous frame is received. This process of sending a frame & waiting for an acknowledgment continues as long as the sender has data to send. To end up the transmission sender transmits end of transmission (EOT) frame. The main advantage of stop & wait protocols is its accuracy. Next frame is transmitted only when the first frame is acknowledged. So there is no chance of frame being lost. The main disadvantage of this method is that it is inefficient. It makes the transmission process slow. In this method single frame travels from source to destination and single acknowledgment travels from destination to source. As a result each frame sent and received uses the entire time needed to traverse the link. Moreover, if two devices are distance apart, a lot of time is wasted waiting for ACKs that leads to increase in total transmission time



Stop & Wait Method.

4. Gigabit Ethernet, a transmission technology based on the Ethernet frame format and protocol used in local area networks (LANs), provides a data rate of 1 billion bits per second (one gigabit). Gigabit Ethernet is defined in the IEEE 802.3 standard and is currently being used as the backbone in many enterprise networks. Gigabit Ethernet is carried primarily on optical fiber (with very short distances possible on

copper media). Existing Ethernet LANs with 10 and 100 Mbps cards can feed into a Gigabit Ethernet backbone. An alternative technology that competes with Gigabit Ethernet is ATM. A newer standard, 10-Gigabit Ethernet, is also becoming available.

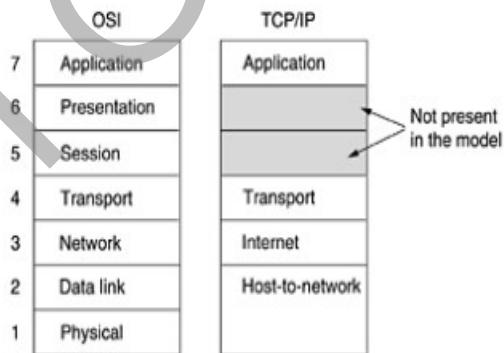
PART B

5.The Internet Layer

All these requirements led to the choice of a packet-switching network based on a connectionless internetwork layer. This layer, called the internet layer, is the linchpin that holds the whole architecture together. Its job is to permit hosts to inject packets into any network and have them travel independently to the destination. They may even arrive in a different order than they were sent, in which case it is the job of higher layers to rearrange them, if in-order delivery is desired. Note that "internet" is used here in a generic sense, even though this layer is present in the Internet.

The analogy here is with the (snail) mail system. A person can drop a sequence of international letters into a mail box in one country, and with a little luck, most of them will be delivered to the correct address in the destination country. Probably the letters will travel through one or more international mail gateways along the way, but this is transparent to the users. Furthermore, that each country (i.e., each network) has its own stamps, preferred envelope sizes, and delivery rules is hidden from the users.

The internet layer defines an official packet format and protocol called IP (Internet Protocol). The job of the internet layer is to deliver IP packets where they are supposed to go. Packet routing is clearly the major issue here, as is avoiding congestion.



The Transport Layer

The layer above the internet layer in the TCP/IP model is now usually called the transport layer. It is designed to allow peer entities on the source and destination hosts to carry on a conversation, just as in the OSI transport layer. Two end-to-end transport protocols have been defined here. The first one, TCP (Transmission Control Protocol), is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error on any other machine in

the internet. It fragments the incoming byte stream into discrete messages and passes each one on to the internet layer. At the destination, the receiving TCP process reassembles the received messages into the output stream. TCP also handles flow control to make sure a fast sender cannot swamp a slow receiver with more messages than it can handle.

The second protocol in this layer, UDP (User Datagram Protocol), is an unreliable, connectionless protocol for applications that do not want TCP's sequencing or flow control and wish to provide their own. It is also widely used for one-shot, client-server-type request-reply queries and applications in which prompt delivery is more important than accurate delivery, such as transmitting speech or video.

The Application Layer

The TCP/IP model does not have session or presentation layers. No need for them was perceived, so they were not included. Experience with the OSI model has proven this view correct: they are of little use to most applications. On top of the transport layer is the application layer. It contains all the higher-level protocols. The early ones included virtual terminal (TELNET), file transfer (FTP), and electronic mail (SMTP). The virtual terminal protocol allows a user on one machine to log onto a distant machine and work there. The file transfer protocol provides a way to move data efficiently from one machine to another. Electronic mail was originally just a kind of file transfer, but later a specialized protocol (SMTP) was developed for it. Many other protocols have been added to these over the years: the Domain Name System (DNS) for mapping host names onto their network addresses, NNTP, the protocol for moving USENET news articles around, and HTTP, the protocol for fetching pages on the World Wide Web, and many others.

The Host-to-Network Layer

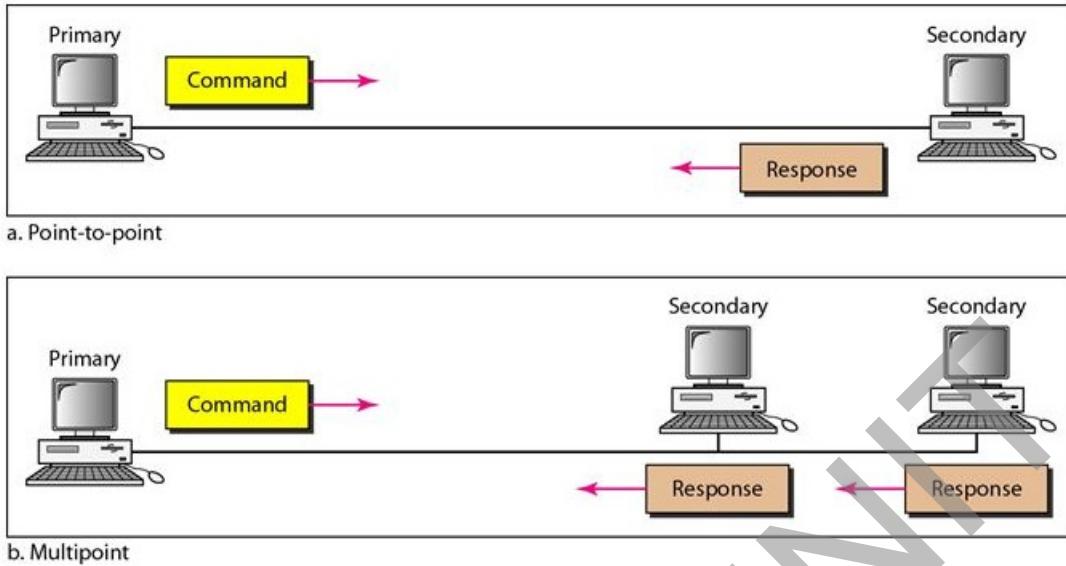
Below the internet layer is a great void. The TCP/IP reference model does not really say much about what happens here, except to point out that the host has to connect to the network using some protocol so it can send IP packets to it. This protocol is not defined and varies from host to host and network to network.

6.a)

HDLC provides two common transfer modes that can be used in different configurations: normal response mode (NRM) and asynchronous balanced mode (ABM).

Normal Response Mode:

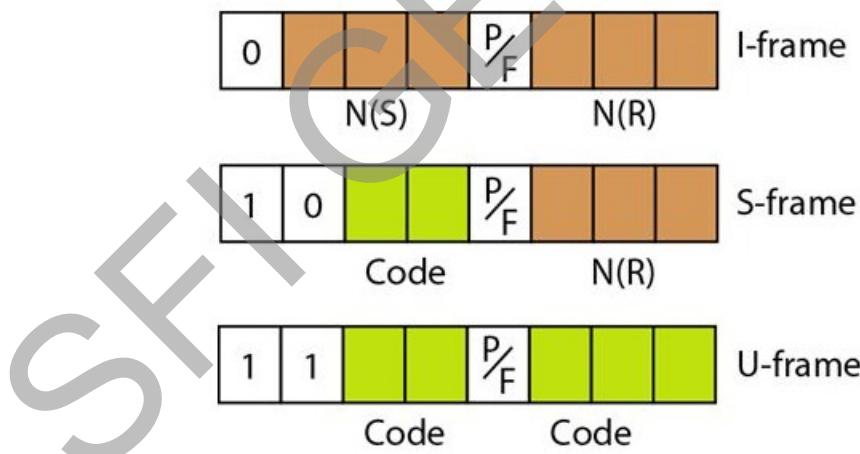
In normal response mode (NRM), the station configuration is unbalanced. We have one primary station and multiple secondary stations. A primary station can send commands, a secondary station can only respond. The NRM is used for both point-to-point and multiple-point links, as shown in the following figure.



Asynchronous Balanced Mode:

In asynchronous balanced mode (ABM), the configuration is balanced. The link is point-to-point, and each station can function as a primary and a secondary (acting as peers). This is the common mode today.

The control field determines the type of frame and defines its functionality. So let us discuss the format of this field in greater detail. The format is specific for the type of frame, as shown in the following figure.



Control Field for I-Frames:

I-frames are designed to carry user data from the network layer. In addition, they can include flow and error control information (piggybacking). The subfields in the control field are used to define these functions.

1. The first bit defines the type. If the first bit of the control field is 0, this means the frame is an I-frame.
2. The next 3 bits, called N(S), define the sequence number of the frame. Note that with 3 bits, we can

define a sequence number between 0 and 7; but in the extension format, in which the control field is 2 bytes, this field is larger.

3. The last 3 bits, called N(R), correspond to the acknowledgment number when piggybacking is used.

4. The single bit between N(S) and N(R) is called the P/F bit. The P/F field is a single bit with a dual purpose. It has meaning only when it is set (bit = 1) and can mean poll or final. It means poll when the frame is sent by a primary station to a secondary (when the address field contains the address of the receiver). It means final when the frame is sent by a secondary to a primary (when the address field contains the address of the sender).

Control Field for S-Frames:

Supervisory frames are used for flow and error control whenever piggybacking is either impossible or inappropriate (e.g., when the station either has no data of its own to send or needs to send a command or response other than an acknowledgment). S-frames do not have information fields.

1. If the first 2 bits of the control field is 10, this means the frame is an S-frame.
2. The last 3 bits, called N(R), corresponds to the acknowledgment number (ACK) or negative acknowledgment number (NAK) depending on the type of S-frame.
3. The 2 bits called code is used to define the type of S-frame itself. With 2 bits, we can have four types of S-frames, as described below:

1. Receive ready (RR):

If the value of the code subfield is 00, it is an RR S-frame. This kind of frame acknowledges the receipt of a safe and sound frame or group of frames. In this case, the value N(R) field defines the acknowledgment number.

2. Receive not ready (RNR):

If the value of the code subfield is 10, it is an RNR S-frame. This kind of frame is an RR frame with additional functions. It acknowledges the receipt of a frame or group of frames, and it announces that the receiver is busy and cannot receive more frames. It acts as a kind of congestion control mechanism by asking the sender to slow down. The value of N(R) is the acknowledgment number.

3. Reject (REJ):

If the value of the code subfield is 01, it is a REJ S-frame. This is a NAK frame, but not like the one used for Selective Repeat ARQ. It is a NAK that can be used in Go-Back-N ARQ to improve the efficiency of the process by informing the sender, before the sender time expires, that the last frame is lost or damaged. The value of N(R) is the negative acknowledgment number.

4. Selective reject (SREJ):

If the value of the code subfield is 11, it is an SREJ S-frame. This is a NAK frame used in Selective Repeat ARQ. Note that the HDLC Protocol uses the term selective reject instead of selective repeat. The value of N(R) is the negative acknowledgment number.

Control Field for U-Frames:

Unnumbered frames are used to exchange session management and control information between connected devices. Unlike S-frames, U-frames contain an information field, but one used for system management information, not user data. As with S-frames, however, much of the information carried by U-frames is contained in codes included in the control field. U-frame codes are divided into two sections: a 2-bit prefix before the P/F bit and a 3-bit suffix after the P/F bit. Together, these two segments (5 bits) can be used to create up to 32 different types of U-frames.

b) PPP is most commonly used data link protocol. It is used to connect the Home PC to the server of ISP via a modem.

- This protocol offers several facilities that were not present in SLIP. Some of these facilities are:
 1. PPP defines the format of the frame to be exchanged between the devices.
 2. It defines link control protocol (LCP) for:-
 - (a) Establishing the link between two devices.
 - (b) Maintaining this established link.
 - (c) Configuring this link.
 - (d) Terminating this link after the transfer.
 3. It defines how network layer data are encapsulated in data link frame.
 4. PPP provides error detection.
 5. Unlike SLIP that supports only IP, PPP supports multiple protocols.
 6. PPP allows the IP address to be assigned at the connection time i.e. dynamically. Thus a temporary IP address can be assigned to each host.
 7. PPP provides multiple network layer services supporting a variety of network layer protocol. For this PPP uses a protocol called NCP (Network Control Protocol).
 8. It also defines how two devices can authenticate each other.

7.a)

To reduce their design complexity, most networks are organized as a stack of layers or levels, each one built upon the one below it. The number of layers, the name of each layer, the contents of each layer, and the function of each layer differ from network to network. The purpose of each layer is to offer certain services to the higher layers, shielding those layers from the details of how the offered services are actually implemented. In a sense, each layer is a kind of virtual machine, offering certain services to the layer above it. This concept is actually a familiar one and used throughout computer science, where it is variously known as information hiding, abstract data types, data encapsulation, and object-oriented programming. The fundamental idea is that a particular piece of software (or hardware) provides a service to its users but keeps the details of its internal state and algorithms hidden from them.

Design Issues for the Layers

Some of the key design issues that occur in computer networks are present in several layers. Below, we will briefly mention some of the more important ones. Every layer needs a mechanism for identifying senders and receivers. Since a network normally has many computers, some of which have multiple processes, a means is needed for a process on one machine to specify with whom it wants to talk. As a consequence of having multiple destinations, some form of addressing is needed in order to specify a specific destination.

Another set of design decisions concerns the rules for data transfer. In some systems, data only travel in one direction; in others, data can go both ways. The protocol must also determine how many logical channels the connection corresponds to and what their priorities are. Many networks provide at least two logical channels per connection, one for normal data and one for urgent data.

Error control is an important issue because physical communication circuits are not perfect. Many error-detecting and error-correcting codes are known, but both ends of the connection must agree on which one is being used. In addition, the receiver must have some way of telling the sender which messages have been correctly received and which have not. Not all communication channels preserve the order of messages sent on them. To deal with a possible loss of sequencing, the protocol must make explicit provision for the receiver to allow the pieces to be reassembled properly. An obvious solution is to number the pieces, but this solution still leaves open the question of what should be done with pieces that arrive out of order.

An issue that occurs at every level is how to keep a fast sender from swamping a slow receiver with data. Various solutions have been proposed and will be discussed later. Some of them involve some kind of feedback from the receiver to the sender, either directly or indirectly, about the receiver's current situation. Others limit the sender to an agreed-on transmission rate. This subject is called flow control. Another problem that must be solved at several levels is the inability of all processes to accept arbitrarily long messages. This property leads to mechanisms for disassembling, transmitting, and then reassembling messages. A related issue is the problem of what to do when processes insist on transmitting data in units that are so small that sending each one separately is inefficient. Here the solution is to gather several small messages heading toward a common destination into a single large message and dismember the large message at the other side.

When it is inconvenient or expensive to set up a separate connection for each pair of communicating processes, the underlying layer may decide to use the same connection for multiple, unrelated conversations. As long as this multiplexing and demultiplexing is done transparently, it can be used by any layer. Multiplexing is needed in the physical layer, for example, where all the traffic for all connections has to be sent over at most a few physical circuits. When there are multiple paths between source and destination, a route must be chosen. Sometimes this decision must be split over two or more layers. For example, to send data from London to Rome, a high-level decision might have to be made to transit France or Germany based on their respective privacy laws. Then a low-level decision might have to be made to select one of the available circuits based on the current traffic load. This topic is called routing.

b)

Medium access basics

Before transmitting frames, a station must first gain access to the medium, which is a radio channel that stations share. The 802.11 standard defines two forms of medium access, distributed coordination function (DCF) and point coordination function (PCF). DCF is mandatory and based on the CSMA/CA (carrier sense multiple access with collision avoidance) protocol. With DCF, 802.11 stations contend for

access and attempt to send frames when there is no other station transmitting. If another station is sending a frame, stations are polite and wait until the channel is free.

As a condition to accessing the medium, the MAC Layer checks the value of its network allocation vector (NAV), which is a counter resident at each station that represents the amount of time that the previous frame needs to send its frame. The NAV must be zero before a station can attempt to send a frame. Prior to transmitting a frame, a station calculates the amount of time necessary to send the frame based on the frame's length and data rate. The station places a value representing this time in the duration field in the header of the frame. When stations receive the frame, they examine this duration field value and use it as the basis for setting their corresponding NAVs. This process reserves the medium for the sending station.

An important aspect of the DCF is a random back off timer that a station uses if it detects a busy medium. If the channel is in use, the station must wait a random period of time before attempting to access the medium again. This ensures that multiple stations wanting to send data don't transmit at the same time. The random delay causes stations to wait different periods of time and avoids all of them sensing the medium at exactly the same time, finding the channel idle, transmitting, and colliding with each other. The back off timer significantly reduces the number of collisions and corresponding retransmissions, especially when the number of active users increases.

With radio-based LANs, a transmitting station can't listen for collisions while sending data, mainly because the station can't have its receiver on while transmitting the frame. As a result, the receiving station needs to send an acknowledgement (ACK) if it detects no errors in the received frame. If the sending station doesn't receive an ACK after a specified period of time, the sending station will assume that there was a collision (or RF interference) and retransmit the frame.

802.11 MAC Layer Functions

Scanning, Authentication, Association, WEP, RTS/CTS, Power save mode, Fragmentation

18.

Flooding is a simple routing technique in computer networks where a source or node sends packets through every outgoing link.

Flooding, which is similar to broadcasting, occurs when source packets (without routing data) are transmitted to all attached network nodes. Because flooding uses every path in the network, the shortest path is also used. The flooding algorithm is easy to implement.

Network routing data is not initially included in data packets. A hop count algorithm is used to track network topology, or visited network routes. A packet tries to access all available network routes and ultimately reaches its destination, but there is always the potential for packet duplication. Hop count and some selective flooding techniques are used to avoid communication delay and duplication.

Flooding is also used as a denial of service attack by flooding network traffic to bring down a network service. The service is flooded with many incomplete server connection requests. Due to the number of flooded requests, the server or host is not able to process genuine requests at the same time. A flooding attack fills the server or host memory buffer; once it is full, further connections cannot be made, which results in denial of service.

9.

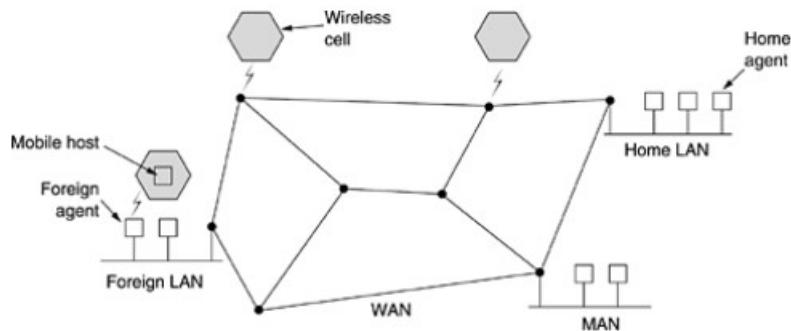
In the model of Fig. 5-18, the world is divided up (geographically) into small units. Let us call

them areas, where an area is typically a LAN or wireless cell. Each area has one or more foreign agents, which are processes that keep track of all mobile hosts visiting the area. In addition, each area has a home agent, which keeps track of hosts whose home is in the area, but who are currently visiting another area.

When a new host enters an area, either by connecting to it (e.g., plugging into the LAN) or just wandering into the cell, his computer must register itself with the foreign agent there. The registration procedure typically works like this:

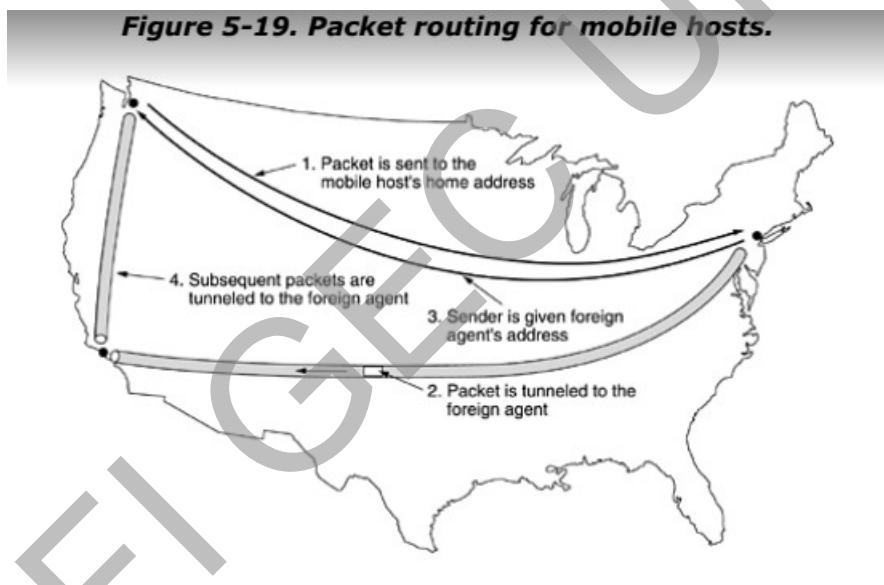
1. Periodically, each foreign agent broadcasts a packet announcing its existence and address. A newly-arrived mobile host may wait for one of these messages, but if none arrives quickly enough, the mobile host can broadcast a packet saying: Are there any foreign agents around?
2. The mobile host registers with the foreign agent, giving its home address, current data link layer address, and some security information
3. The foreign agent contacts the mobile host's home agent and says: One of your hosts is over here. The message from the foreign agent to the home agent contains the foreign agent's network address. It also includes the security information to convince the home agent that the mobile host is really there.
4. The home agent examines the security information, which contains a timestamp, to prove that it was generated within the past few seconds. If it is happy, it tells the foreign agent to proceed.
5. When the foreign agent gets the acknowledgement from the home agent, it makes an entry in its tables and informs the mobile host that it is now registered.

Ideally, when a host leaves an area, that, too, should be announced to allow deregistration, but many users abruptly turn off their computers when done.



The home agent then does two things. First, it encapsulates the packet in the payload field of an outer packet and sends the latter to the foreign agent (step 2 in Fig. 5-19). This mechanism is called tunneling; we will look at it in more detail later. After getting the encapsulated packet, the foreign agent removes the original packet from the payload field and sends it to the mobile host as a data link frame.

Second, the home agent tells the sender to henceforth send packets to the mobile host by encapsulating them in the payload of packets explicitly addressed to the foreign agent instead of just sending them to the mobile host's home address (step 3). Subsequent packets can now be routed directly to the host via the foreign agent (step 4), bypassing the home location entirely.



10. Congestion, in the context of networks, refers to a network state where a node or link carries so much data that it may deteriorate network service quality, resulting in queuing delay, frame or data packet loss and the blocking of new connections. In a congested network, response time slows with reduced network throughput. Congestion occurs when bandwidth is insufficient and network data traffic exceeds capacity.

Data packet loss from congestion is partially countered by aggressive network protocol retransmission, which maintains a network congestion state after reducing the initial data load. This can create two stable states under the same data traffic load - one dealing with the initial load and the other maintaining reduced network throughput.

Transmission Control Protocol (TCP) uses a network congestion-avoidance algorithm that includes various aspects of an additive increase/multiplicative decrease (AIMD) scheme, with other schemes such as slow-start and congestion window to achieve congestion avoidance. The TCP congestion-avoidance

algorithm is the primary basis for congestion control in the Internet. According to the end-to-end principle, congestion control is largely a function of internet hosts, not the network itself. There are several variations and versions of the algorithm implemented in protocol stacks of operating systems of computers that connect to the Internet.

11.

IPv4	IPv6
IPv4 addresses are 32 bit length.	IPv6 addresses are 128 bit length.
IPv4 addresses are binary numbers represented in decimals.	IPv6 addresses are binary numbers represented in hexadecimals.
IPSec support is only optional.	Inbuilt IPSec support.
Fragmentation is done by sender and forwarding routers.	Fragmentation is done only by sender.
No packet flow identification.	Packet flow identification is available within the IPv6 header using the Flow Label field.
Checksum field is available in IPv4 header	No checksum field in IPv6 header.
Options fields are available in IPv4 header.	No option fields, but IPv6 Extension headers are available.
Address Resolution Protocol (ARP) is available to map IPv4 addresses to MAC addresses.	Address Resolution Protocol (ARP) is replaced with a function of Neighbor Discovery Protocol (NDP).
Internet Group Management Protocol (IGMP) is used to manage multicast group membership.	IGMP is replaced with Multicast Listener Discovery (MLD) messages.
Broadcast messages are available.	Broadcast messages are not available. Instead a link-local scope "All nodes" multicast IPv6 address (FF02::1) is used for broadcast similar functionality.
Manual configuration (Static) of IPv4 addresses or DHCP (Dynamic configuration) is required to configure IPv4 addresses.	Auto-configuration of addresses is available.

5.2.4 Distance Vector Routing

Modern computer networks generally use dynamic routing algorithms rather than the static ones described above because static algorithms do not take the current network load into account. Two dynamic algorithms in particular, distance vector routing and link state routing, are the most popular. In this section we will look at the former algorithm. In the following section we will study the latter algorithm.

Distance vector routing algorithms operate by having each router maintain a table (i.e., a vector) giving the best known distance to each destination and which line to use to get there. These tables are updated by exchanging information with the neighbors.

The distance vector routing algorithm is sometimes called by other names, most commonly the distributed Bellman-Ford routing algorithm and the Ford-Fulkerson algorithm, after the researchers who developed it (Bellman, 1957; and Ford and Fulkerson, 1962). It was the original ARPANET routing algorithm and was also used in the Internet under the name RIP.

In distance vector routing, each router maintains a routing table indexed by, and containing one entry for, each router in the subnet. This entry contains two parts: the preferred outgoing line to use for that destination and an estimate of the time or distance to that destination. The metric used might be number of hops, time delay in milliseconds, total number of packets queued along the path, or something similar.

The router is assumed to know the "distance" to each of its neighbors. If the metric is hops, the distance is just one hop. If the metric is queue length, the router simply examines each queue. If the metric is delay, the router can measure it directly with special ECHO packets that the receiver just timestamps and sends back as fast as it can.

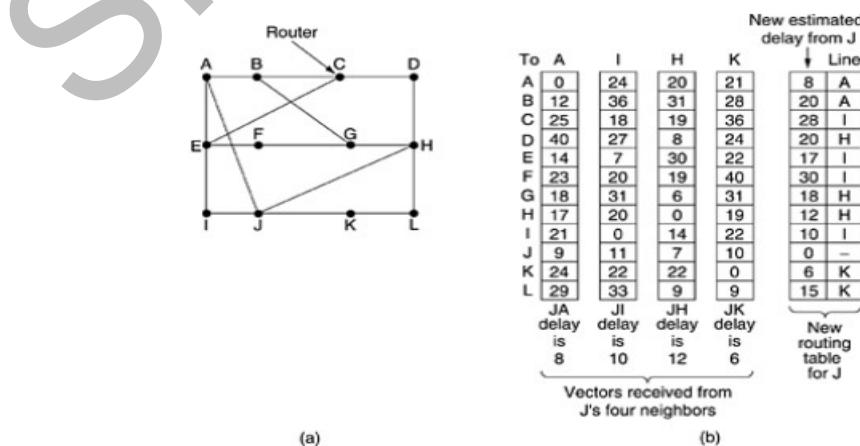
As an example, assume that delay is used as a metric and that the router knows the delay to each of its neighbors. Once every T msec each router sends to each neighbor a list of its estimated delays to each destination. It also receives a similar list from each neighbor.

Imagine that one of these tables has just come in from neighbor X , with X_i being X 's estimate of how long it takes to get to router i . If the router knows that the delay to X is m msec, it also knows that it can reach router i via X in $X_i + m$ msec. By performing this calculation for each neighbor, a router can find out which estimate seems the best and use that estimate and the

corresponding line in its new routing table. Note that the old routing table is not used in the calculation.

This updating process is illustrated in Fig. 5-9. Part (a) shows a subnet. The first four columns of part (b) show the delay vectors received from the neighbors of router J. A claims to have a 12-msec delay to B, a 25-msec delay to C, a 40-msec delay to D, etc. Suppose that J has measured or estimated its delay to its neighbors, A, I, H, and K as 8, 10, 12, and 6 msec, respectively.

Figure 5-9. (a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.



Consider how J computes its new route to router G. It knows that it can get to A in 8 msec, and A claims to be able to get to G in 18 msec, so J knows it can count on a delay of 26 msec to G if it forwards packets bound for G to A. Similarly, it computes the delay to G via I, H, and K as 41 (31 + 10), 18 (6 + 12), and 37 (31 + 6) msec, respectively. The best of these values is 18, so it makes an entry in its routing table that the delay to G is 18 msec and that the route to use is via H. The same calculation is performed for all the other destinations, with the new routing table shown in the last column of the figure.

13

5.2.5 Link State Routing

Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing. Two primary problems caused its demise. First, since the delay metric was queue length, it did not take line bandwidth into account when choosing routes. Initially, all the lines were 56 kbps, so line bandwidth was not an issue, but after some lines had been upgraded to 230 kbps and others to 1.544 Mbps, not taking bandwidth into account was a major problem. Of course, it would have been possible to change the delay metric to factor in line bandwidth, but a second problem also existed, namely, the algorithm often took too long to converge (the count-to-infinity problem). For these reasons, it was replaced by an entirely new algorithm, now called link state routing. Variants of link state routing are now widely used.

The idea behind link state routing is simple and can be stated as five parts. Each router must do the following:

1. Discover its neighbors and learn their network addresses.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router.

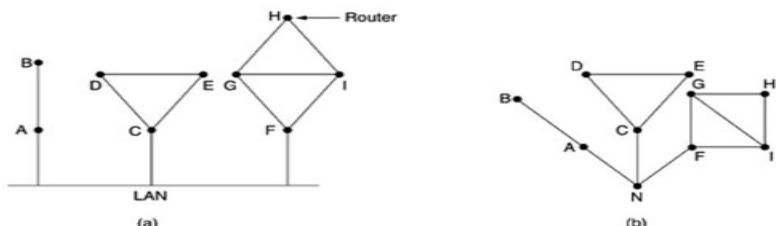
In effect, the complete topology and all delays are experimentally measured and distributed to every router. Then Dijkstra's algorithm can be run to find the shortest path to every other router. Below we will consider each of these five steps in more detail.

Learning about the Neighbors

When a router is booted, its first task is to learn who its neighbors are. It accomplishes this goal by sending a special HELLO packet on each point-to-point line. The router on the other end is expected to send back a reply telling who it is. These names must be globally unique because when a distant router later hears that three routers are all connected to F, it is essential that it can determine whether all three mean the same F.

When two or more routers are connected by a LAN, the situation is slightly more complicated. Fig. 5-11(a) illustrates a LAN to which three routers, A, C, and F, are directly connected. Each of these routers is connected to one or more additional routers, as shown.

Figure 5-11. (a) Nine routers and a LAN. (b) A graph model of (a).



One way to model the LAN is to consider it as a node itself, as shown in Fig. 5-11(b). Here we have introduced a new, artificial node, N, to which A, C, and F are connected. The fact that it is possible to go from A to C on the LAN is represented by the path ANC here.

Measuring Line Cost

The link state routing algorithm requires each router to know, or at least have a reasonable estimate of, the delay to each of its neighbors. The most direct way to determine this delay is to send over the line a special ECHO packet that the other side is required to send back

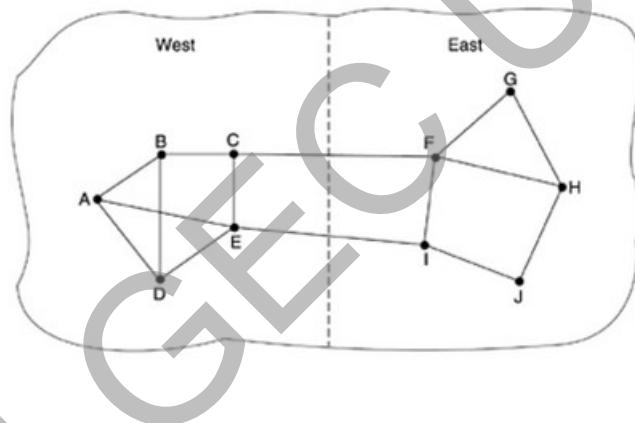
immediately. By measuring the round-trip time and dividing it by two, the sending router can get a reasonable estimate of the delay. For even better results, the test can be conducted several times, and the average used. Of course, this method implicitly assumes the delays are symmetric, which may not always be the case.

An interesting issue is whether to take the load into account when measuring the delay. To factor the load in, the round-trip timer must be started when the ECHO packet is queued. To ignore the load, the timer should be started when the ECHO packet reaches the front of the queue.

Arguments can be made both ways. Including traffic-induced delays in the measurements means that when a router has a choice between two lines with the same bandwidth, one of which is heavily loaded all the time and one of which is not, the router will regard the route over the unloaded line as a shorter path. This choice will result in better performance.

Unfortunately, there is also an argument against including the load in the delay calculation. Consider the subnet of [Fig. 5-12](#), which is divided into two parts, East and West, connected by two lines, *CF* and *EI*.

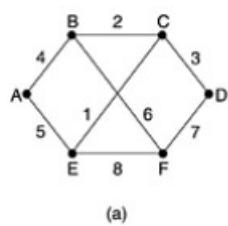
Figure 5-12. A subnet in which the East and West parts are connected by two lines.



Suppose that most of the traffic between East and West is using line *CF*, and as a result, this line is heavily loaded with long delays. Including queueing delay in the shortest path calculation will make *EI* more attractive. After the new routing tables have been installed, most of the East-West traffic will now go over *EI*, overloading this line. Consequently, in the next update, *CF* will appear to be the shortest path. As a result, the routing tables may oscillate wildly, leading to erratic routing and many potential problems. If load is ignored and only bandwidth is considered, this problem does not occur. Alternatively, the load can be spread over both lines, but this solution does not fully utilize the best path. Nevertheless, to avoid oscillations in the choice of best path, it may be wise to distribute the load over multiple lines, with some known fraction going over each line.

Building Link State Packets

Once the information needed for the exchange has been collected, the next step is for each router to build a packet containing all the data. The packet starts with the identity of the sender, followed by a sequence number and age (to be described later), and a list of neighbors. For each neighbor, the delay to that neighbor is given. An example subnet is given in [Fig. 5-13\(a\)](#) with delays shown as labels on the lines. The corresponding link state packets for all six routers are shown in [Fig. 5-13\(b\)](#).

Figure 5-13. (a) A subnet. (b) The link state packets for this subnet.

(a)

Link	State	Packets
A	B	E
	Seq.	F
	Age	Seq.
	B 4	Seq.
	E 5	Age
	C	A
	Seq.	5
	Age	6
	B 2	7
	D 3	8
	F 7	1
	E 1	2
	C 3	3
	D 7	4
	F 8	5
	E 8	6

(b)

Building the link state packets is easy. The hard part is determining when to build them. One possibility is to build them periodically, that is, at regular intervals. Another possibility is to build them when some significant event occurs, such as a line or neighbor going down or coming back up again or changing its properties appreciably.

Distributing the Link State Packets

The trickiest part of the algorithm is distributing the link state packets reliably. As the packets are distributed and installed, the routers getting the first ones will change their routes. Consequently, the different routers may be using different versions of the topology, which can lead to inconsistencies, loops, unreachable machines, and other problems.

First we will describe the basic distribution algorithm. Later we will give some refinements. The fundamental idea is to use flooding to distribute the link state packets. To keep the flood in check, each packet contains a sequence number that is incremented for each new packet sent. Routers keep track of all the (source router, sequence) pairs they see. When a new link state packet comes in, it is checked against the list of packets already seen. If it is new, it is forwarded on all lines except the one it arrived on. If it is a duplicate, it is discarded. If a packet with a sequence number lower than the highest one seen so far ever arrives, it is rejected as being obsolete since the router has more recent data.

This algorithm has a few problems, but they are manageable. First, if the sequence numbers wrap around, confusion will reign. The solution here is to use a 32-bit sequence number. With one link state packet per second, it would take 137 years to wrap around, so this possibility can be ignored.

the duplicate is discarded. If they are different, the older one is thrown out. To guard against errors on the router-router lines, all link state packets are acknowledged. When a line goes idle, the holding area is scanned in round-robin order to select a packet or acknowledgement to send.

The data structure used by router B for the subnet shown in Fig. 5-13(a) is depicted in Fig. 5-14. Each row here corresponds to a recently-arrived, but as yet not fully-processed, link state packet. The table records where the packet originated, its sequence number and age, and the data. In addition, there are send and acknowledgement flags for each of B's three lines (to A, C, and F, respectively). The send flags mean that the packet must be sent on the indicated line. The acknowledgement flags mean that it must be acknowledged there.

Figure 5-14. The packet buffer for router B in Fig. 5-13.

Source	Seq.	Age	Send flags			ACK flags			Data
			A	C	F	A	C	F	
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

In Fig. 5-14, the link state packet from A arrives directly, so it must be sent to C and F and acknowledged to A, as indicated by the flag bits. Similarly, the packet from F has to be forwarded to A and C and acknowledged to F.

However, the situation with the third packet, from E, is different. It arrived twice, once via EAB and once via EFB. Consequently, it has to be sent only to C but acknowledged to both A and F, as indicated by the bits.

the duplicate is discarded. If they are different, the older one is thrown out. To guard against errors on the router-router lines, all link state packets are acknowledged. When a line goes idle, the holding area is scanned in round-robin order to select a packet or acknowledgement to send.

The data structure used by router *B* for the subnet shown in [Fig. 5-13\(a\)](#) is depicted in [Fig. 5-14](#). Each row here corresponds to a recently-arrived, but as yet not fully-processed, link state packet. The table records where the packet originated, its sequence number and age, and the data. In addition, there are send and acknowledgement flags for each of *B*'s three lines (to *A*, *C*, and *F*, respectively). The send flags mean that the packet must be sent on the indicated line. The acknowledgement flags mean that it must be acknowledged there.

Figure 5-14. The packet buffer for router B in [Fig. 5-13](#).

Source	Seq.	Age	Send flags			ACK flags			Data
			A	C	F	A	C	F	
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

In [Fig. 5-14](#), the link state packet from *A* arrives directly, so it must be sent to *C* and *F* and acknowledged to *A*, as indicated by the flag bits. Similarly, the packet from *F* has to be forwarded to *A* and *C* and acknowledged to *F*.

However, the situation with the third packet, from *E*, is different. It arrived twice, once via *EAB* and once via *EFB*. Consequently, it has to be sent only to *C* but acknowledged to both *A* and *F*, as indicated by the bits.

If a duplicate arrives while the original is still in the buffer, bits have to be changed. For example, if a copy of *C*'s state arrives from *F* before the fourth entry in the table has been forwarded, the six bits will be changed to 100011 to indicate that the packet must be acknowledged to *F* but not sent there.

Computing the New Routes

Once a router has accumulated a full set of link state packets, it can construct the entire subnet graph because every link is represented. Every link is, in fact, represented twice, once for each direction. The two values can be averaged or used separately.

Now Dijkstra's algorithm can be run locally to construct the shortest path to all possible destinations. The results of this algorithm can be installed in the routing tables, and normal operation resumed.

For a subnet with n routers, each of which has k neighbors, the memory required to store the input data is proportional to kn . For large subnets, this can be a problem. Also, the computation time can be an issue. Nevertheless, in many practical situations, link state routing works well.

However, problems with the hardware or software can wreak havoc with this algorithm (also with other ones). For example, if a router claims to have a line it does not have or forgets a line it does have, the subnet graph will be incorrect. If a router fails to forward packets or

corrupts them while forwarding them, trouble will arise. Finally, if it runs out of memory or does the routing calculation wrong, bad things will happen. As the subnet grows into the range of tens or hundreds of thousands of nodes, the probability of some router failing occasionally becomes nonnegligible. The trick is to try to arrange to limit the damage when the inevitable happens. Perlman (1988) discusses these problems and their solutions in detail.

Link state routing is widely used in actual networks, so a few words about some example protocols using it are in order. The OSPF protocol, which is widely used in the Internet, uses a link state algorithm. We will describe OSPF in [Sec. 5.6.4](#).

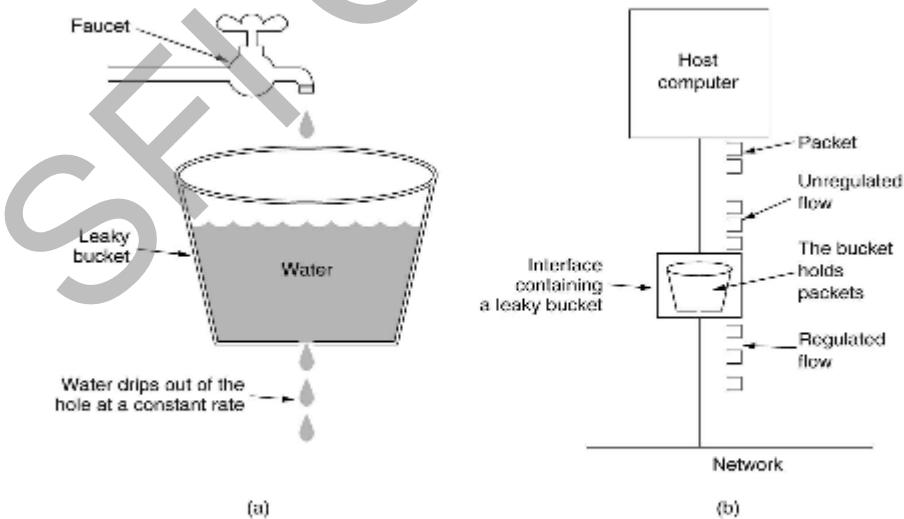
Another link state protocol is IS-IS (Intermediate System-Intermediate System), which was designed for DECnet and later adopted by ISO for use with its connectionless network layer protocol, CLNP. Since then it has been modified to handle other protocols as well, most notably, IP. IS-IS is used in some Internet backbones (including the old NSFNET backbone) and in some digital cellular systems such as CDPD. Novell NetWare uses a minor variant of IS-IS (NLSP) for routing IPX packets.

Basically IS-IS distributes a picture of the router topology, from which the shortest paths are computed. Each router announces, in its link state information, which network layer addresses it can reach directly. These addresses can be IP, IPX, AppleTalk, or any other addresses. IS-IS can even support multiple network layer protocols at the same time.

Many of the innovations designed for IS-IS were adopted by OSPF (OSPF was designed several years after IS-IS). These include a self-stabilizing method of flooding link state updates, the concept of a designated router on a LAN, and the method of computing and supporting path splitting and multiple metrics. As a consequence, there is very little difference between IS-IS and OSPF. The most important difference is that IS-IS is encoded in such a way that it is easy and natural to simultaneously carry information about multiple network layer protocols, a feature OSPF does not have. This advantage is especially valuable in large multiprotocol environments.

14 a)

The Leaky Bucket Algorithm



- The **Leaky Bucket Algorithm** used to control rate in a network. It is implemented as a single-server queue with constant service time.
- If the bucket (buffer) overflows then packets are discarded.
- http://www.cisco.com/en/US/products/hw/switches/ps1893/products_feature_guide_chapter09186a008007e39e.html

Leaky Bucket Algorithm

- The leaky bucket algorithm uses two parameters to control traffic flow:
 - **Average rate:** The average number of cells per second that "leak" from the hole in the bottom of the bucket and enter the network.
 - **Burst rate:** The rate at which cells are allowed to accumulate in the bucket, expressed in cells per second. For example, if the average burst rate is 10 cells per second, a burst of 10 seconds allows 100 cells to accumulate in the bucket.

- The leaky bucket algorithm also uses two state variables:
 - **Current time:** The current wall clock time.
 - **Virtual time:** A measure of how much data has accumulated in the bucket, expressed in seconds.

- For example, if the average rate is 10 cells per second and 100 cells have accumulated in the bucket, then the virtual time is 10 seconds ahead of the current time.

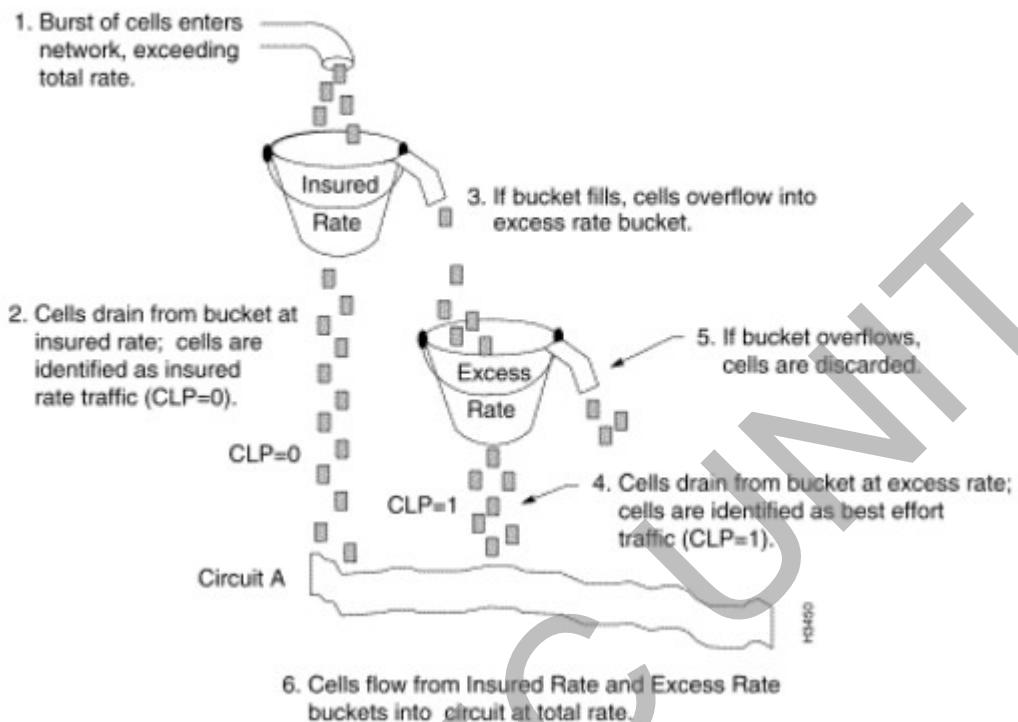
Leaky Bucket Algorithm

- If, for example, the average rate is 10 cells per second, and the burst is 50 cells, the virtual time and current time remain the same as long as the input rate remains at or below 10 cells per second.
- If an instantaneous burst of 25 cells is received, the virtual time moves ahead of the current time by 2.5 seconds. If this is followed immediately by a second burst of 30 cells, the virtual time moves ahead of the current time by 5 cells, and the last 5 of the 30 cells are dropped.
- For packet traffic, the unit of incoming data is larger than a single ATM cell. For packet interfaces, the leaky bucket algorithm takes the packet size into account, as shown in the following formula:

```

virtual time = max (virtual time, current time)
if (virtual time + (packet size / average rate) > current time + burst)
    drop the incoming packet
else
    segment the packet into cells
    put the cells in the bucket
    virtual time = virtual time + (packet size/average rate)
  
```

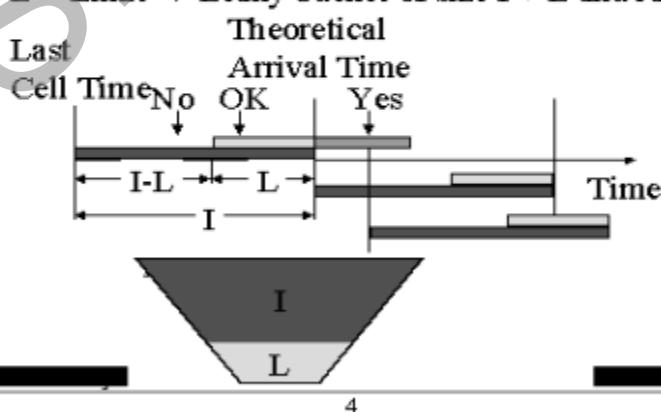
Leaky Bucket Algorithm



GCRA

Generic Cell Rate Algorithm: GCRA(I, L)

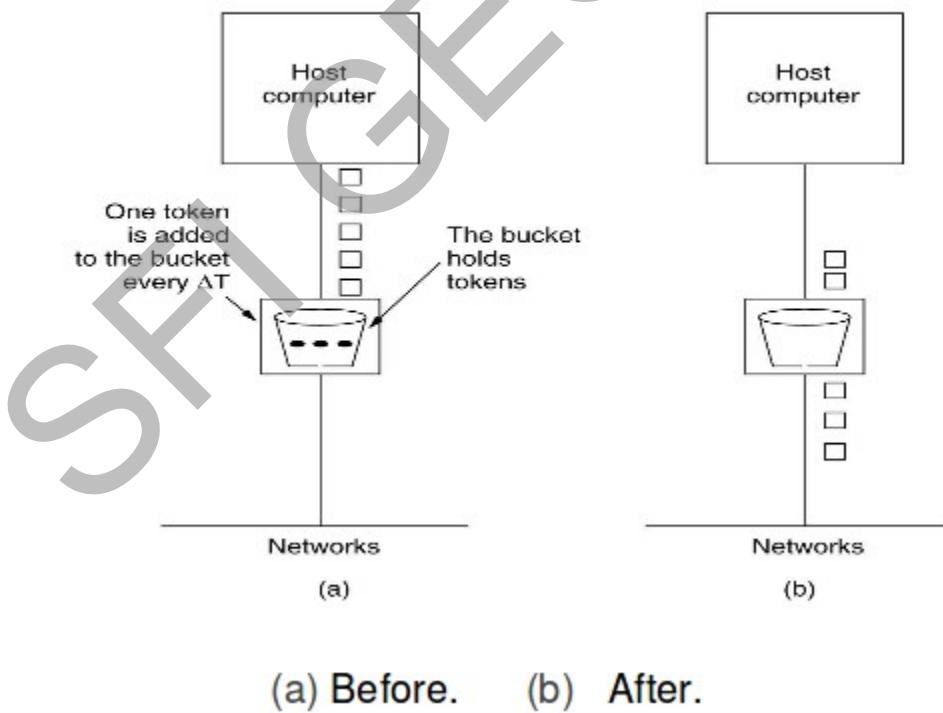
- I = Increment = Inter-cell Time = Cell size/PCR
- L = Limit \Rightarrow Leaky bucket of size $I + L$ and rate 1



Token Bucket Algorithm

- | In contrast to the LB, the Token Bucket Algorithm, allows the output rate to vary, depending on the size of the burst.
- | In the TB algorithm, the bucket holds tokens. To transmit a packet, the host must capture and destroy one token.
- | Tokens are generated by a clock at the rate of one token every Δt sec.
- | Idle hosts can capture and save up tokens (up to the max. size of the bucket) in order to send larger bursts later.
- | Example from: <http://www.opalsoft.net/qos/CDS-22-A1.htm>

The Token Bucket Algorithm



b)

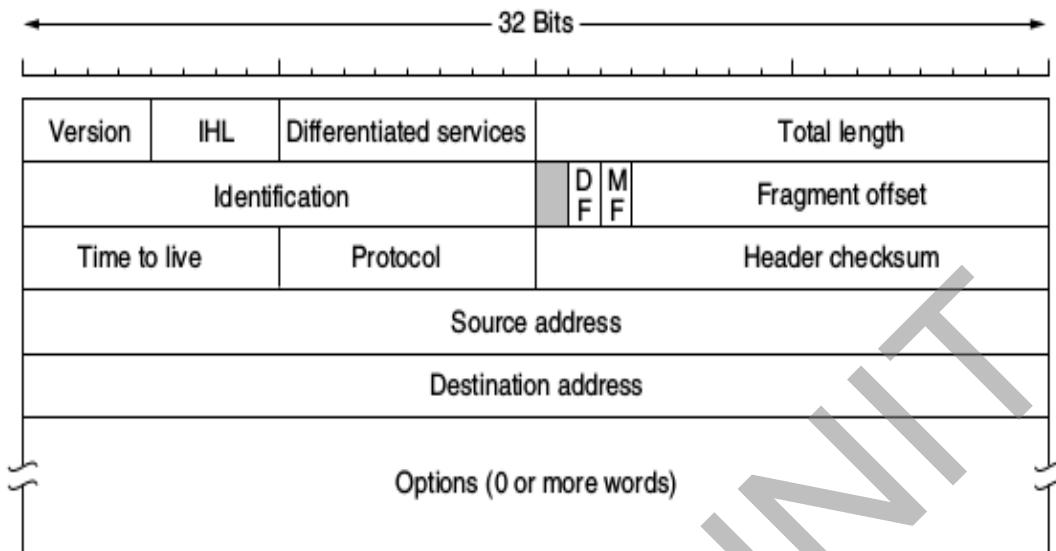


Figure 5-46. The IPv4 (Internet Protocol) header.

The *Version* field keeps track of which version of the protocol the datagram belongs to. Version 4 dominates the Internet today, and that is where we have started our discussion. By including the version at the start of each datagram, it becomes possible to have a transition between versions over a long period of time. In fact, IPv6, the next version of IP, was defined more than a decade ago, yet is only just beginning to be deployed. We will describe it later in this section. Its use will eventually be forced when each of China's almost 2^{31} people has a desktop PC, a laptop, and an IP phone. As an aside on numbering, IPv5 was an experimental real-time stream protocol that was never widely used.

Since the header length is not constant, a field in the header, *IHL*, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, which limits the header to 60 bytes, and thus the *Options* field to 40 bytes. For some options, such as one that records the route a packet has taken, 40 bytes is far too small, making those options useless.

The *Differentiated services* field is one of the few fields that has changed its meaning (slightly) over the years. Originally, it was called the *Type of service* field. It was and still is intended to distinguish between different classes of service. Various combinations of reliability and speed are possible. For digitized voice, fast delivery beats accurate delivery. For file transfer, error-free transmission is more important than fast transmission. The *Type of service* field provided 3 bits to signal priority and 3 bits to signal whether a host cared more about delay, throughput, or reliability. However, no one really knew what to do with these bits at routers, so they were left unused for many years. When differentiated services were designed, IETF threw in the towel and reused this field. Now, the top 6 bits are used to mark the packet with its service class; we described the expedited and assured services earlier in this chapter. The bottom 2 bits are used to carry explicit congestion notification information, such as whether the packet has experienced congestion; we described explicit congestion notification as part of congestion control earlier in this chapter.

The *Total length* includes everything in the datagram—both header and data. The maximum length is 65,535 bytes. At present, this upper limit is tolerable, but with future networks, larger datagrams may be needed.

The *Identification* field is needed to allow the destination host to determine which packet a newly arrived fragment belongs to. All the fragments of a packet contain the same *Identification* value.

Next comes an unused bit, which is surprising, as available real estate in the IP header is extremely scarce. As an April Fool's joke, Bellovin (2003) proposed using this bit to detect malicious traffic. This would greatly simplify security, as packets with the “evil” bit set would be known to have been sent by attackers and could just be discarded. Unfortunately, network security is not this simple.

Then come two 1-bit fields related to fragmentation. *DF* stands for Don't Fragment. It is an order to the routers not to fragment the packet. Originally, it was intended to support hosts incapable of putting the pieces back together again. Now it is used as part of the process to discover the path MTU, which is the largest packet that can travel along a path without being fragmented. By marking the datagram with the *DF* bit, the sender knows it will either arrive in one piece, or an error message will be returned to the sender.

MF stands for More Fragments. All fragments except the last one have this bit set. It is needed to know when all fragments of a datagram have arrived.

The *Fragment offset* tells where in the current packet this fragment belongs. All fragments except the last one in a datagram must be a multiple of 8 bytes, the

elementary fragment unit. Since 13 bits are provided, there is a maximum of 8192 fragments per datagram, supporting a maximum packet length up to the limit of the *Total length* field. Working together, the *Identification*, *MF*, and *Fragment offset* fields are used to implement fragmentation as described in Sec. 5.5.5.

The *TTL* (*Time to live*) field is a counter used to limit packet lifetimes. It was originally supposed to count time in seconds, allowing a maximum lifetime of 255 sec. It must be decremented on each hop and is supposed to be decremented multiple times when a packet is queued for a long time in a router. In practice, it just counts hops. When it hits zero, the packet is discarded and a warning packet is sent back to the source host. This feature prevents packets from wandering around forever, something that otherwise might happen if the routing tables ever become corrupted.

When the network layer has assembled a complete packet, it needs to know what to do with it. The *Protocol* field tells it which transport process to give the packet to. TCP is one possibility, but so are UDP and some others. The numbering of protocols is global across the entire Internet. Protocols and other assigned numbers were formerly listed in RFC 1700, but nowadays they are contained in an online database located at www.iana.org.

Since the header carries vital information such as addresses, it rates its own checksum for protection, the *Header checksum*. The algorithm is to add up all the 16-bit halfwords of the header as they arrive, using one's complement arithmetic, and then take the one's complement of the result. For purposes of this algorithm, the *Header checksum* is assumed to be zero upon arrival. Such a checksum is useful for detecting errors while the packet travels through the network. Note that it must be recomputed at each hop because at least one field always changes (the *Time to live* field), but tricks can be used to speed up the computation.

The *Source address* and *Destination address* indicate the IP address of the source and destination network interfaces. We will discuss Internet addresses in the next section.

The *Options* field was designed to provide an escape to allow subsequent versions of the protocol to include information not present in the original design, to permit experimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed. The options are of variable length. Each begins with a 1-byte code identifying the option. Some options are followed by a 1-byte option length field, and then one or more data bytes. The *Options* field is padded out to a multiple of 4 bytes. Originally, the five options listed in Fig. 5-47 were defined.

The *Security* option tells how secret the information is. In theory, a military router might use this field to specify not to route packets through certain countries the military considers to be “bad guys.” In practice, all routers ignore it, so its only practical function is to help spies find the good stuff more easily.

The *Strict source routing* option gives the complete path from source to destination as a sequence of IP addresses. The datagram is required to follow that

Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

Figure 5-47. Some of the IP options.

exact route. It is most useful for system managers who need to send emergency packets when the routing tables have been corrupted, or for making timing measurements.

The *Loose source routing* option requires the packet to traverse the list of routers specified, in the order specified, but it is allowed to pass through other routers on the way. Normally, this option will provide only a few routers, to force a particular path. For example, to force a packet from London to Sydney to go west instead of east, this option might specify routers in New York, Los Angeles, and Honolulu. This option is most useful when political or economic considerations dictate passing through or avoiding certain countries.

The *Record route* option tells each router along the path to append its IP address to the *Options* field. This allows system managers to track down bugs in the routing algorithms (“Why are packets from Houston to Dallas visiting Tokyo first?”). When the ARPANET was first set up, no packet ever passed through more than nine routers, so 40 bytes of options was plenty. As mentioned above, now it is too small.

Finally, the *Timestamp* option is like the *Record route* option, except that in addition to recording its 32-bit IP address, each router also records a 32-bit timestamp. This option, too, is mostly useful for network measurement.

15.a)i,

Basis for Comparison	ARP	RARP
Full Form	Address Resolution Protocol.	Reverse Address Resolution Protocol.
Basic	Retrieves the physical address of the receiver.	Retrieves the logical address for a computer from the server.
Mapping	ARP maps 32-bit logical (IP) address to 48-bit physical address.	RARP maps 48-bit physical address to 32-bit logical (IP) address.

15.a)ii)

	TCP	UDP
Acronym for	Transmission Control Protocol	User Datagram Protocol or Universal Datagram Protocol
Connection	TCP is a connection-oriented protocol.	UDP is a connectionless protocol.
Function	As a message makes its way across the internet from one computer to another. This is connection based.	UDP is also a protocol used in message transport or transfer. This is not connection based which means that one program can send a load of packets to another and that would be the end of the relationship.
Usage	TCP is suited for applications that require high reliability, and transmission time is relatively less critical.	UDP is suitable for applications that need fast, efficient transmission, such as games. UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients.
Use by other protocols	HTTP, HTTPS, FTP, SMTP, Telnet	DNS, DHCP, TFTP, SNMP, RIP, VOIP.
Ordering of data packets	TCP rearranges data packets in the order specified.	UDP has no inherent order as all packets are independent of each other. If ordering is required, it has to be managed by the application layer.
Speed of transfer	The speed for TCP is slower than UDP.	UDP is faster because error recovery is not attempted. It is a "best effort" protocol.

Reliability	There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent.	There is no guarantee that the messages or packets sent would reach at all.
Header Size	TCP header size is 20 bytes	UDP Header size is 8 bytes.
Common Header Fields	Source port, Destination port, Check Sum	Source port, Destination port, Check Sum
Streaming of data	<ul style="list-style-type: none"> Data is read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries. 	Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent
Weight	TCP is heavy-weight. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.	UDP is lightweight. There is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP.
Data Flow Control	TCP does Flow Control. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.	UDP does not have an option for flow control
Error Checking	TCP does error checking and error recovery. Erroneous packets are retransmitted from the source to the destination.	UDP does error checking but simply discards erroneous packets. Error recovery is not attempted.

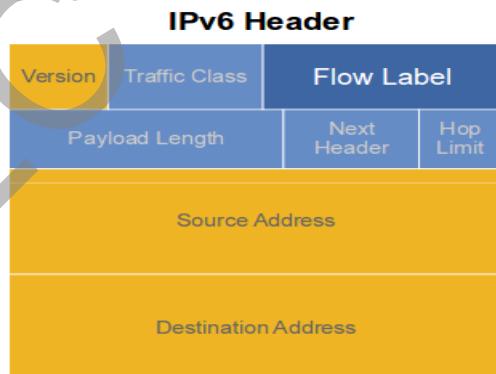
Fields	1. Sequence Number, 2. AcK number, 3. Data offset, 4. Reserved, 5. Control bit, 6. Window, 7. Urgent Pointer 8. Options, 9. Padding, 10. Check Sum, 11. Source port, 12. Destination port	1. Length, 2. Source port, 3. Destination port, 4. Check Sum
Acknowledgement	Acknowledgement segments	No Acknowledgment
Handshake	SYN, SYN-ACK, ACK	No handshake (connectionless protocol)

b)

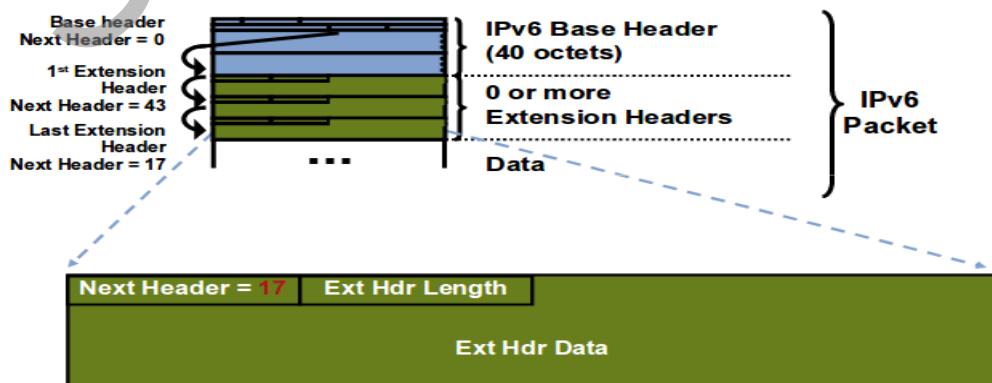
IPv6 Header New Field—Flow Label (RFC3697)

20-Bit Flow Label Field to Identify Specific Flows
Needing Special QoS

- Flow classifiers had been based on 5-tuple: Source/destination address, protocol type and port numbers of transport
- Some of these fields may be unavailable due to fragmentation, encryption or locating them past extension headers
- With flow label, each source chooses its own flow label values; routers use source addr + flow label to identify distinct flows
- Flow label value of 0 used when no special QoS requested (the common case today)



Extension Headers



IPv6 Addressing

IPv4 32-bits

IPv6 128-bits

$$2^{32} = 4,294,967,296$$

$$2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456$$

$$2^{128} = 2^{32} \times 2^{96}$$

$2^{96} = 79,228,162,514,264,337,593,543,950,336$ times the
number of possible IPv4 Addresses
(79 trillion trillion)

Addressing Format

Representation

- 16-bit hexadecimal numbers
- Numbers are separated by (:)
- Hex numbers are not case sensitive
- Abbreviations are possible

Leading zeros in contiguous block could be represented by (::)

Example:

2001:0db8:0000:130F:0000:0000:087C:140B

2001:0db8:0:130F::87C:140B

Double colon only appears once in the address

IPv6—Addressing Model

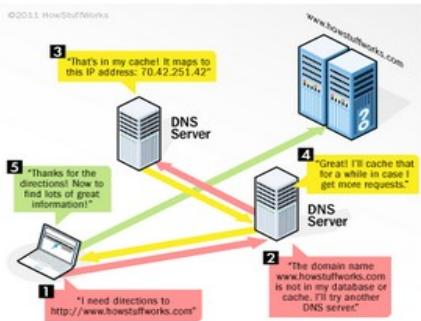
- Addresses are assigned to interfaces
Change from IPv4 mode:
 - Interface “expected” to have multiple addresses
 - Addresses have scope

Link Local
Unique Local
Global



Types of IPv6 Addresses

- Unicast
Address of a single interface. One-to-one delivery to single interface
- Multicast
Address of a set of interfaces. One-to-many delivery to all interfaces in the set
- Anycast
Address of a set of interfaces. One-to-one-of-many delivery to a single interface in the set that is closest
- No more broadcast addresses



When you enter a URL into your Web browser, your DNS server uses its resources to resolve the name into the IP address for the appropriate Web server. See more [computer networking pictures](#).

©HOWSTUFFWORKS.COM

If you've ever used the [Internet](#), it's a good bet that you've used the **Domain Name System**, or **DNS**, even without realizing it. DNS is a protocol within the set of standards for how computers exchange data on the Internet and on many private networks, known as the TCP/IP protocol suite. Its basic job is to turn a user-friendly **domain name** like "howstuffworks.com" into an [Internet Protocol \(IP\) address](#) like 70.42.251.42 that computers use to identify each other on the network. It's like your computer's GPS for the Internet.

Computers and other network devices on the Internet use an IP address to route your request to the site you're trying to reach. This is similar to dialing a phone number to connect to the person you're trying to call. Thanks to DNS, though, you don't have to keep your own address book of IP addresses. Instead, you just

connect through a **domain name server**, also called a **DNS server** or **name server**, which manages a massive database that maps domain names to IP addresses.

Whether you're accessing a Web site or sending [e-mail](#), your computer uses a DNS server to look up the domain name you're trying to access. The proper term for this process is **DNS name resolution**, and you would say that the DNS server resolves the domain name to the IP address. For example, when you enter "https://www.howstuffworks.com" in your browser, part of the network connection includes resolving the domain name "howstuffworks.com" into an IP address, like 70.42.251.42, for HowStuffWorks' Web servers.

You can always bypass a DNS lookup by entering 70.42.251.42 directly in your browser (give it a try). However, you're probably more likely to remember "howstuffworks.com" when you want to return later. In addition, a Web site's IP address can change over time, and some sites associate multiple IP addresses with a single domain name.

Without DNS servers, the Internet would shut down very quickly. But how does your computer know what DNS server to use? Typically, when you connect to your [home network](#), Internet service provider (ISP) or WiFi network, the modem or router that assigns your computer's network address also sends some important network configuration information to your computer or mobile device. That configuration includes one or more DNS servers that the device should use when translating DNS names to IP address.

So far, you've read about some important DNS basics. The rest of this article dives deeper into domain name servers and name resolution. It even includes an introduction to managing your own DNS server. Let's start by looking at how IP addresses are structured and how that's important to the name resolution process.

17 E-mail System

E-mail system comprises of the following three components:

Mailer, Mail Server, Mailbox, Mailer

It is also called mail program, mail application or mail client. It allows us to manage, read and compose e-mail.

Mail Server

The function of mail server is to receive, store and deliver the email. It is must for mail servers to be sunning all the time because if it crashes or is down, email can be lost.

Mailboxes

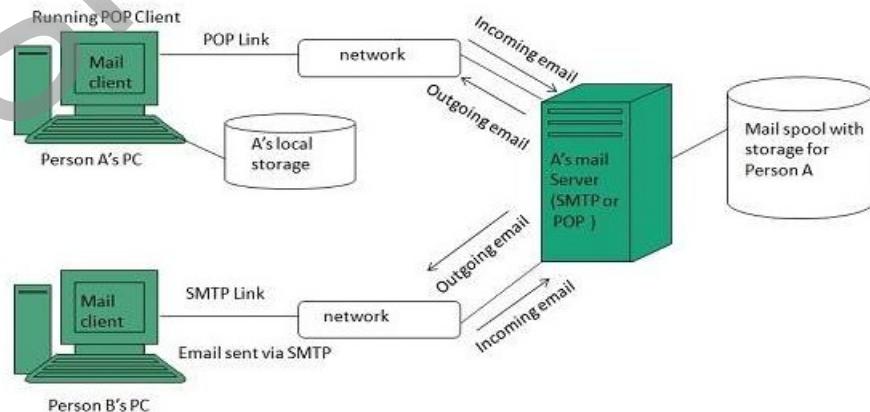
Mailbox is generally a folder that contains emails and information about them.

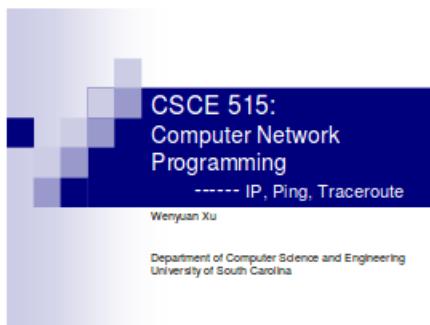
Working of E-mail

Email working follows the client server approach. In this client is the mailer i.e. the mail application or mail program and server is a device that manages emails. Following example will take you through the basic steps involved in sending and receiving emails and will give you a better understanding of working of email system:

Suppose person A wants to send an email message to person B. Person A composes the messages using a mailer program i.e. mail client and then select Send option. The message is routed to Simple Mail Transfer Protocol to person B's mail server. The mail server stores the email message on disk in an area designated for person B. The disk space area on mail server is called mail spool. Now, suppose person B is running a POP client and knows how to communicate with B's mail server. It will periodically poll the POP server to check if any new email has arrived for B. As in this case, person B has sent an email for person B, so email is forwarded over the network to B's PC. This message is now stored on person B's PC.

The following diagram gives pictorial representation of the steps discussed above:



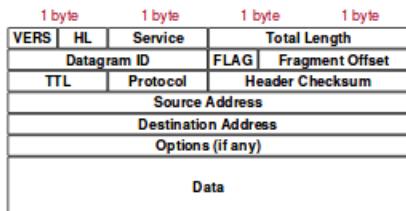


ICMP Internet Control Message Protocol

- ICMP is a protocol used for exchanging control messages.
- Two main categories
 - Query message
 - Error message
- Usage of an ICMP message is determined by **type** and **code** fields.
- ICMP uses IP to deliver messages.
- ICMP messages are usually generated and processed by the IP software, not the user process.

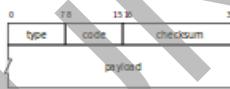


IP Datagram



CSCE515 - Computer Network Programming

ICMP Message Format



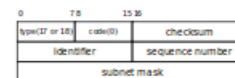
ICMP Message Types

- Echo Request
- Echo Response
- Destination Unreachable
- Redirect
- Time Exceeded
- there are more ...

CSCE515 - Computer Network Programming

ICMP Address Mask Request and Reply

- intended for a diskless system to obtain its subnet mask.
- Id and seq can be any values, and these values are returned in the reply.
 - Match replies with request



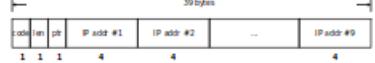
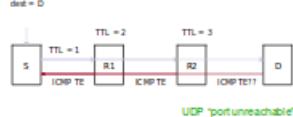
ping Program

- Available at /usr/sbin/ping
- Test whether another host is reachable
- Send ICMP echo_request to a network host
- -n option to set number of echo request to send
- -i option to set TTL
- -R option to record route (apollon.cse.sc.edu)
- -t option to set timestamp
- -w option to set timeout to wait for each reply
- Check manual, different ping versions have different options

CSCE515 - Computer Network Programming

ICMP Echo Request and Reply



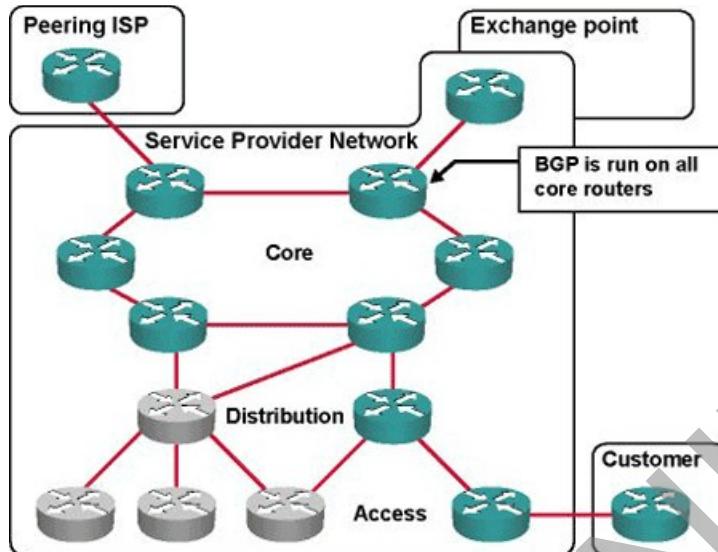
<p>IP Record Route Option</p> <ul style="list-style-type: none"> ■ ping - R : Record route <ul style="list-style-type: none"> □ Every router that handles the datagram adds its IP address to a list in the options field □ The final destination copies the IP addresses into the outgoing ICMP echo reply □ All routers on the return path add their IP address to the list ■ Problems?  <p style="text-align: center;">GSEC515 – Computer Network Programming</p>	<p>traceroute Program</p> <ul style="list-style-type: none"> ■ Available at /usr/sbin/traceroute ■ Display the route that IP datagrams follow from one host to another ■ Compare with ping: <ul style="list-style-type: none"> □ Doesn't require any special or optional features at any intermediate routers □ Only requires a working UDP module at the destination □ Uses ICMP and the TTL field in the IP header ■ -g option to specify intermediate routers to be used with loose source routing (up to 8 times) ■ -c option to specify intermediate routers to be used with strict source routing (up to 8 times)  <p style="text-align: center;">GSEC515 – Computer Network Programming</p>
<p>traceroute Program</p> <ul style="list-style-type: none"> ■ TTL + ICMP <ul style="list-style-type: none"> □ Each router decrements the TTL at least by 1 □ A IP datagram whose TTL is either 0 or 1 will not be forwarded. □ An ICMP "time exceeded" message will be sent back to the originating host.  <p style="text-align: center;">UDP "port unreachable"</p>	<p>UDP port unreachable</p> <ul style="list-style-type: none"> ■ ICMP error message <ul style="list-style-type: none"> □ IP header □ 8 bytes of the IP datagram that caused the error ■ WHY?  <p style="text-align: center;">GSEC515 – Computer Network Programming</p>

19.

BGP (Border Gateway Protocol) is protocol that manages how packets are routed across the internet through the exchange of routing and reachability information between edge routers. BGP directs packets between autonomous systems (AS) -- networks managed by a single enterprise or service provider. Traffic that is routed within a single network AS is referred to as internal BGP, or iBGP. More often, BGP is used to connect one AS to other autonomous systems, and it is then referred to as an external BGP, or eBGP.

BGP offers network stability that guarantees routers can quickly adapt to send packets through another reconnection if one internet path goes down. BGP makes routing decisions based on paths, rules or network policies configured by a network administrator. Each BGP router maintains a standard routing table used to direct packets in transit. This table is used in conjunction with a separate routing table, known as the routing information base (RIB), which is a data table stored on a server on the BGP router. The RIB contains route information both from directly connected external peers, as well as internal peers, and continually updates the routing table as changes occur. BGP is based on TCP/IP and uses client-server topology to communicate routing information, with the client-server initiating a BGP session by sending a request to the server.

BGP sends updated router table information only when something changes -- and even then, it sends only the affected information. BGP has no automatic discovery mechanism, which means connections between peers have to be set up manually, with peer addresses programmed in at both ends.

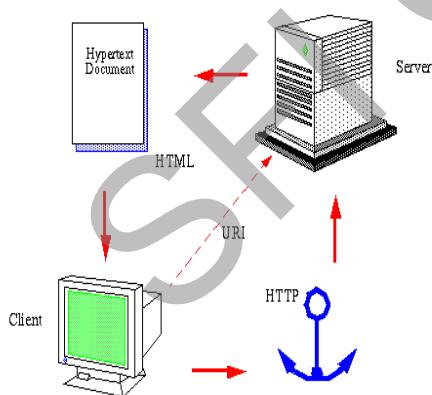


BGP makes best-path decisions based on current reachability, hop counts and other path characteristics. In situations where multiple paths are available -- as within a major hosting facility -- BGP can be used to communicate an organization's own preferences in terms of what path traffic should follow in and out of its networks. BGP even has a mechanism for defining arbitrary tags, called communities, which can be used to control route advertisement behavior by mutual agreement among peers.

20.

Basic World-Wide Web Model

The basic idea behind the World-Wide Web is based on a client server application and hypertext documents as illustrated in the figure below. The model is simplified in that it only contains elements that exists within the World-Wide Web concept. Later the model will be expanded to a generic resource accessing model.



The Client

The client is the user's interface to the Internet. Whatever type of service requested this interface stays the same, so users do not need to understand the differences between the many different access schemes in common use on the Internet. This principle is the same as is seen from other popular applications such as Microsoft Windows, Macintosh etc. where the user is always presented to the same GUI interface.

Uniform Resource Identifier URI

The user initiates a request by specifying a Uniform Resource Identifier or a "hyperlink". This link can specify any accessible information or resource on the Internet as long as it can be uniquely identified as an object. The word "Web" refers to the combination of accessible objects and the links pointing to them throughout the Internet.

The Server

The server is responsible for handling the request sent from the client. This can either be a local accessible resource or the server can request the resource from another server in which case the first server temporarily turns into a client.

Hypertext Transport Protocol HTTP

The client sends the user request to a WWW server using the Hypertext Transfer Protocol (HTTP). This is a typical client-server application based on a stateless connection between the client requesting the URI and the server handling the request.

Hypertext Markup Language HTML

On a successful request, a data object is returned from the server to the client. The object is written in the Hypertext Markup Language (HTML) which is a hypertext language with the possibility of containing hyperlinks that the user can follow.

The model basically reflects the first version of the World-Wide Web as it is described in the HTTP Protocol version 0.9 and HTML version 1.0. However, the WWW specifications have been rapidly changing during the last 3-4 years, even though the current model is still based on a client-server approach. From being a HTML and HTTP based model, the World-Wide Web is now capable of handling virtually any existing data format on the Internet using a large set of access methods apart from HTTP such as FTP, Gopher, WAIS, Telnet etc. In other words, the World-Wide Web represents a generic information exchange tool capable of accessing information throughout the Internet. Though, before the more advanced model is presented, it is necessary to get an overview of the basic elements in the WWW model mentioned above.

Universal Resource Identifiers

In order to address a data object or more general, a resource, in the model above it is necessary to define a name space that not only contains information about hosts but also about resources available on each host. The World-Wide Web model defines Uniform Resource Identifiers or URIs that specifies a syntax for encoding the names and addresses of data objects on the Internet and how they can be accessed. The set of URIs covers



Universal Resource Identifier (URI)

A generic set of all addresses in the address space of all resources on the Internet. They describe a hierarchical naming scheme that together with the HTTP protocol makes a significant difference between the World-Wide Web model and other Internet access schemes such as FTP that has a flat address space.

Uniform Resource Locator (URL)

The term "URI" has been introduced by the IETF and is a general description of all URL that are not persistent. In practice the URLs consist of the current set of Internet protocols supported by the WWW, i.e., HTTP, FTP, Gopher, WAIS, etc., followed by a directory path, a file name, and possibly a search directive.

Uniform Resource Name (URN)

However, the ultimate goal for URIs is to be a persistent naming scheme independent of the mean of access, i.e., the protocol used and of the physical structure of resources on the specific host. The only way to obtain this is to have a naming scheme like the Internet Domain Name Service. URNs are currently under consideration in IETF but little is known about the status of the research.

Uniform Resource Citation (URC)

This is meta information about a URI. They consist of pairs of attribute/value which can contain information on the author, publisher etc. The URC are currently not used.

Hypertext Transfer Protocol

The Hypertext Transport Protocol (HTTP) is a generic stateless presentation layer protocol with elements from other Internet presentation layer protocols. The HTTP protocol is built on a client-server model where the client initiates a request and the server replies with a response.

The basic format of the HTTP protocol is based on the MIME Protocol with a set of HTTP Headers possibly followed by a message body containing a data object in any 7-bit or 8-bit accepted by the client. The client specifies what data format it can handle by having a list of accept headers in the request.

The basic WWW-model indicates that the client initiates a request and the server responds by sending a data object to the client. However, often the client wants to post a data object to the server, e.g. to post a mail message to an email address, to a news group, or to create a new file on the remote server. The HTTP protocol provides two methods for the client to transfer a data object to the server. Though, the client is not guaranteed that the request can be fulfilled - even on a successful return code. The action can at all times be cancelled by the responsible person of the remote server.

One of the characteristics of the HTTP protocol is that it is a superset of the other Presentation Layer supported by the WWW-model. This means that messages generated by other protocols can be handled by the HTTP protocol by wrapping a set of HTTP/MIME headers around the message. This is an essential feature for the concept of Proxy servers.

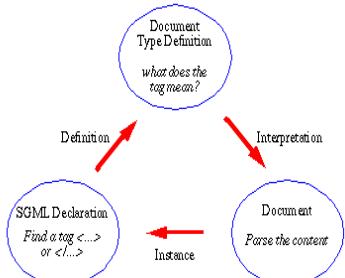
The current version 1.0 of the HTTP protocol is built on top of the TCP Protocol that is a connection oriented protocol with a 3 way handshake connection establishment. This causes an substantial overhead in a client-server oriented environment like the HTTP protocol. It would therefore be an excessive optimization if the HTTP protocol was moved to a lighter Transport Layer protocol such as the Transactional TCP Protocol which still provides a reliable stream transport service.

Hypertext Markup Language

The Hypertext Markup language (HTML) is the users interface to create information on the World-Wide Web. The description of the World-Wide Web has until now focused on the technology that due to specifications and conventions provide the functionality necessary to request and serve information across the Internet. HTML is defined to be the hypertext language of communication which actually flows over the network. There is no requirement that files are stored in HTML. Servers may store files in any other formats and then generate a HTML on the fly upon a client request. This gives the possibility of having virtual documents instead of static documents on rapidly changing information like weather reports etc. HTML can be used to represent:

- Hypertext news, mail, online documentation, and collaborative hypermedia
- Menus and options
- Database query results
- Simple structured documents with inlined multi media elements like images, audio and movie
- URI-Links to other resources on the Internet.

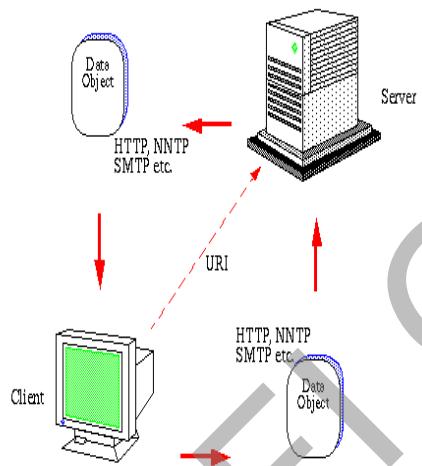
HTML is built on top of the International Standard ISO 8879 Standard Generalized Markup Language (SGML). SGML is a system for defining structured document types and markup languages to represent instances of the document types. That is, HTML is a [Document Type Definition \(DTD\)](#) used on top of a SGML parser. Every SGML based document contains three elements as illustrated in the figure:



HTML is now superseded by [HTML+](#) that is an enriched DTD with possibilities of handling tables, math, images etc. Currently many browsers support a subset of the HTML+ specifications in addition to the basic HTML features.

Interactive World-Wide Web Model

The description of the [Universal Resource Identifiers](#), the [Hypertext Transfer Protocol](#), and the [Hypertext Markup Language](#) now calls for an update of the [Basic WWW-model](#) as illustrated in the figure.



This model is a generic resource exchange model based on a client-server concept. Instead of the limited model with data flowing only from the server to the client, the client is capable of posting data to the server if the server allows this kind of service. Furthermore, the format of the data transferred in the message body can have any format from 7-bit ASCII text to 8-bit binary data. The transfer carrier can be any protocol supported by the World-Wide Web but the main protocol is HTTP as it can be used to encapsulate the other protocols supported, even a FTP message that is a highly state dependent protocol.

Reg. No. _____

Name: _____

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

SIXTH SEMESTER B.TECH DEGREE MODEL EXAMINATION, APRIL 2018

Course Code: CS 306

Course Name: Computer Networks

Max. Marks: 100

Duration: 3 Hours

Part A

(4x3=12)

(Answer All Questions. Each caries 3 marks)

1. Explain IEEE 802.5 standard?
2. Explain the terms interface ,services and protocols.
3. Explain the states of operation of PPP protocol.
4. Distinguish between Fast Ethernet and Gigabit Ethernet standards.

Part B

(2x9=18)

(Answer any two. Each caries 9 marks)

5. Identify the layers in OSI reference model and illustrate their functions.
6. How packet loss is detected in Go-Back-N ARQ technique and Selective Repeat protocol.
7. Explain the architecture and communication model of HDLC protocol.

Part C

(4x3=12)

(Answer All Questions. Each caries 3 marks)

8. Explain the different classes of IP addresses.
9. Explain optimality principle.
10. Explain the concept of flooding.
11. Draw the header structure of IPv4 protocol and explain the fields.

.

Part D

(2x9=18)

(Answer any two. Each carries 9 marks)

12. Discuss about the techniques to improve QoS in internetworking.
13. Illustrate the routing procedure in mobile networks.
14. Explain the various congestion control techniques.
15. Explain Link State routing algorithm.

Part E

(4x10=40)

(Answer any four. Each carries 10 marks)

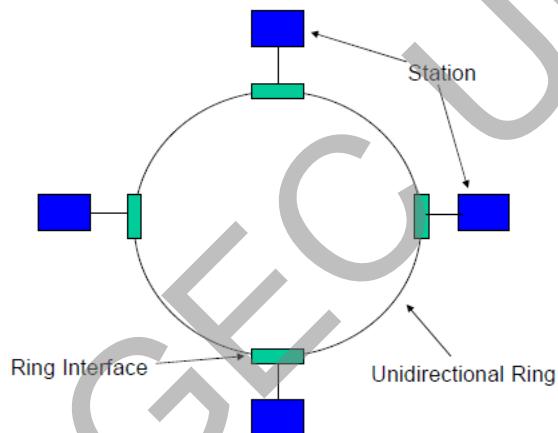
16. Illustrate the working of OSPF protocols.
17. Explain how control messages are sent using ICMP protocol.
- 18 a. Give the format of TCP header and discuss the relevance of various fields
b. How transport layer connection is established in TCP? Illustrate with state diagrams.
19. Explain the process of address resolution by using ARP and RARP protocol.
20. Explain how network is managed by using SNMP protocol.
21. Explain the architecture of World Wide Web.

Part A
(Answer All Questions. Each carries 3 marks)

(4x3=12)

1. Explain IEEE 802.5 standard?

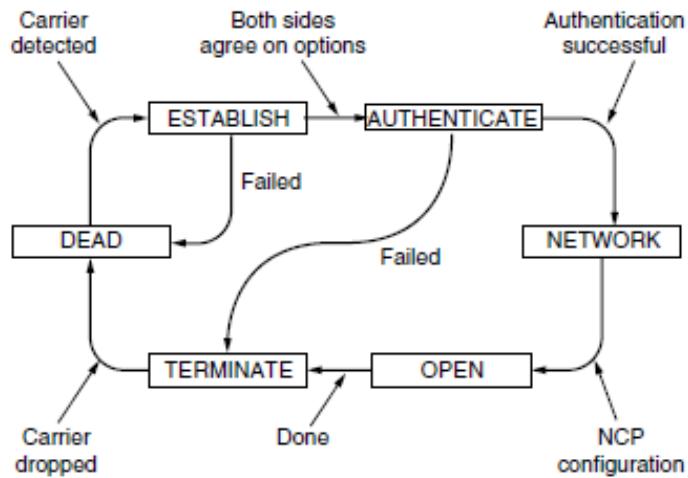
- Ring is not a broadcast medium but a collection of point-to-point links forming a circle.
- Token is a special Frame which gives the holder station the “Right to Transmit”
- All stations are connected are organized in a Logical ring
- Frames are passed from the Predecessor to the successor after a specified time interval
- When there is no data to be sent the token circulates around the logical ring
- Whenever a station has data to send, it waits for a token to arrive
- Station then captures the token and keeps transmitting data until allocated time for keeping the token expires
- After the specified time the token must be passed on to the successor
- Fair operation with an upper bound on channel access
- Channel access problem is solved with the help of a special frame called a “Token”



2. Explain the terms interface ,services and protocols .

To reduce the complexity of design , most networks are organized as a stack of **Layers**. When layer n on one machine carries on a conversation with layer n on another machine, the rules and conventions used in this conversation are collectively known as the layer n protocol. Basically, a **protocol** is an agreement between the communicating parties on how communication is to proceed. The interface defines which primitive operations and services the lower layer makes available to the upper one. A service is a set of primitives (operations) that a layer provides to the layer above it. The service defines what operations the layer is prepared to perform on behalf of its users, but it says nothing at all about how these operations are implemented. A service relates to an interface between two layers, with the lower layer being the service provider and the upper layer being the service user. A protocol, in contrast, is a set of rules governing the format and meaning of the packets, or messages that are exchanged by the peer entities within a layer. Entities use protocols to implement their service definitions.

3. Explain the states of operation of PPP protocol.

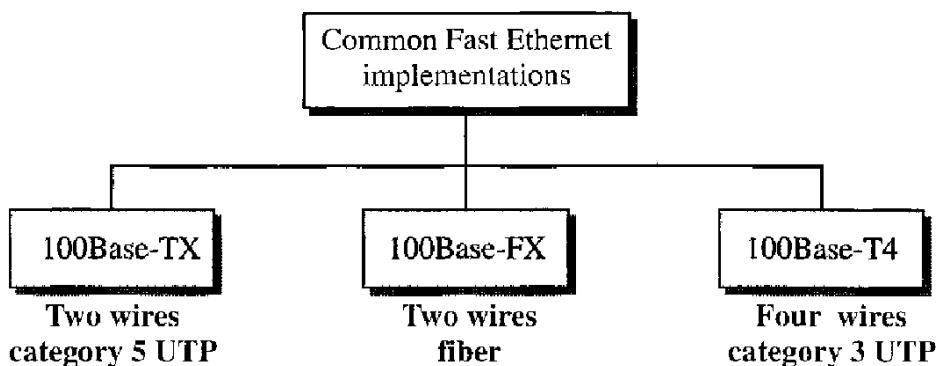


The link starts in the DEAD state, which means that there is no connection at the physical layer. When a physical layer connection is established, the link moves to ESTABLISH. At this point, the PPP peers exchange a series of LCP packets, each carried in the Payload field of a PPP frame, to select the PPP options for the link from the possibilities mentioned above. The initiating peer proposes options, and the responding peer either accepts or rejects them, in whole or part. The responder can also make alternative proposals. If LCP option negotiation is successful, the link reaches the AUTHENTICATE state. Now the two parties can check each other's identities, if desired. If authentication is successful, the NETWORK state is entered and a series of NCP packets are sent to configure the network layer. Once OPEN is reached, data transport can take place. It is in this state that IP packets are carried in PPP frames across the SONET line. When data transport is finished, the link moves into the TERMINATE state, and from there it moves back to the DEAD state when the physical layer connection is dropped.

4. Distinguish between Fast Ethernet and Gigabit Ethernet standards.

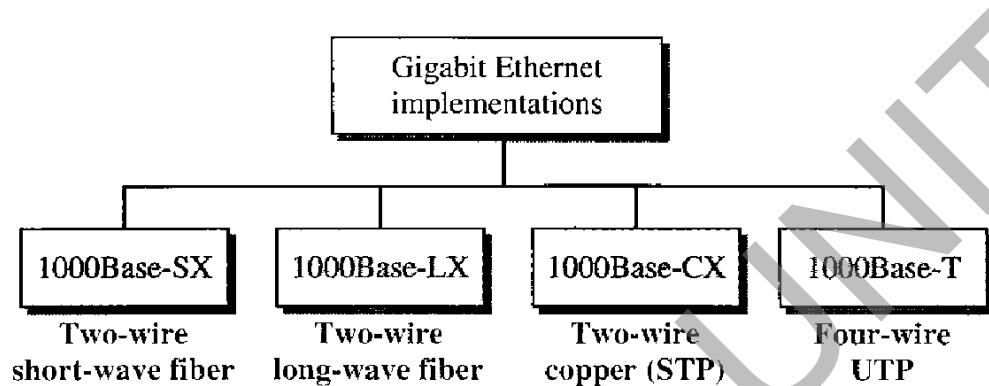
The goal of fast Ethernet can be summarized as

1. Upgrade the data rate to 100 Mbps.
2. Make it compatible with Standard Ethernet.
3. Keep the same 48-bit address.
4. Keep the same frame format.
5. Keep the same minimum and maximum frame lengths.



Gigabit Ethernet design can be summarized as follows:

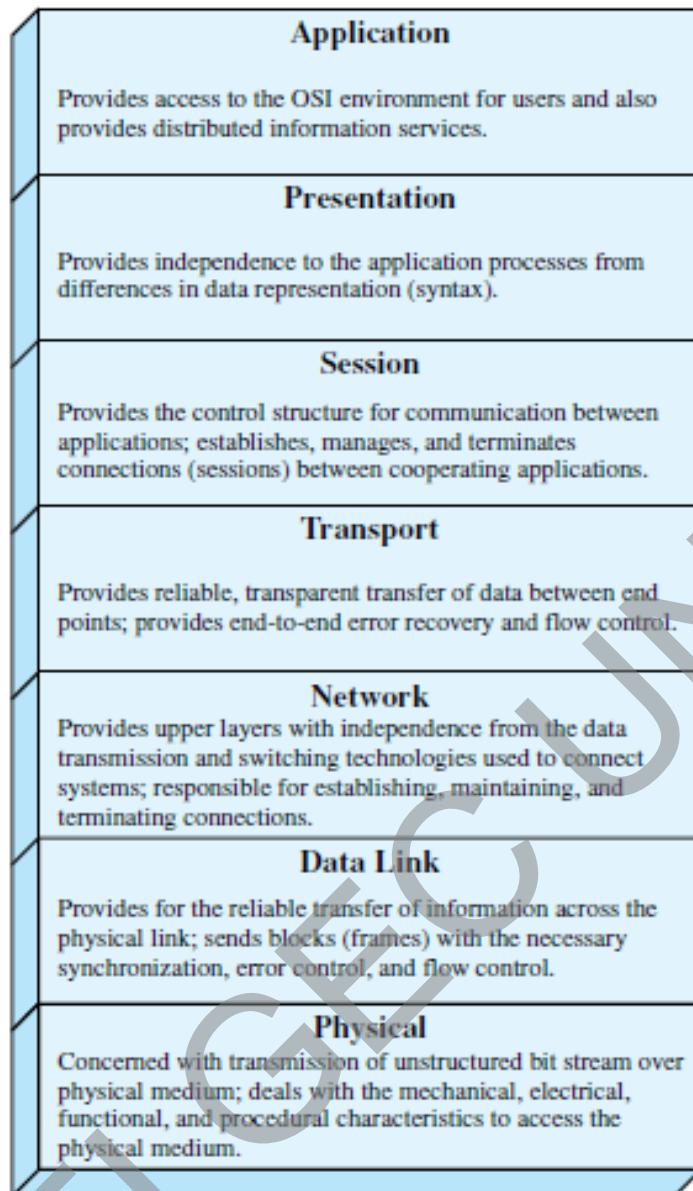
1. Upgrade the data rate to 1 Gbps.
2. Make it compatible with Standard or Fast Ethernet.
3. Use the same 48-bit address.
4. Use the same frame format.
5. Keep the same minimum and maximum frame lengths.
6. To support autonegotiation as defined in Fast Ethernet.



Part B
(Answer any two. Each carries 9 marks)

(2x9=18)

5. Identify the layers in OSI reference model and illustrate their functions.



6. How packet loss is detected in Go-Back-N ARQ technique and Selective Repeat protocol.

- detection and correction of errors such as:
 - lost frames
 - damaged frames
- common techniques use:
 - error detection
 - positive acknowledgment
 - retransmission after timeout
 - negative acknowledgement & retransmission
- **Automatic Repeat Request (ARQ)** collective name for such error control mechanisms, including :stop and wait, go back N and selective reject (selective retransmission)

1. Go-Back-N is based on sliding window

- if no error, ACK as usual
- use window to control number of outstanding frames

- if error, reply with rejection
- discard that frame and all future frames until error frame received correctly
- transmitter must go back and retransmit that frame and all subsequent frames

Damaged Frame

- error in frame i so receiver rejects frame i
- transmitter retransmits frames from i

Lost Frame

- frame i lost and either
- transmitter sends i+1 and receiver gets frame i+1 out of seq and rejects frame i
- or transmitter times out and send ACK with P bit set which receiver responds to with ACK i
- transmitter then retransmits frames from i

Damaged Acknowledgement

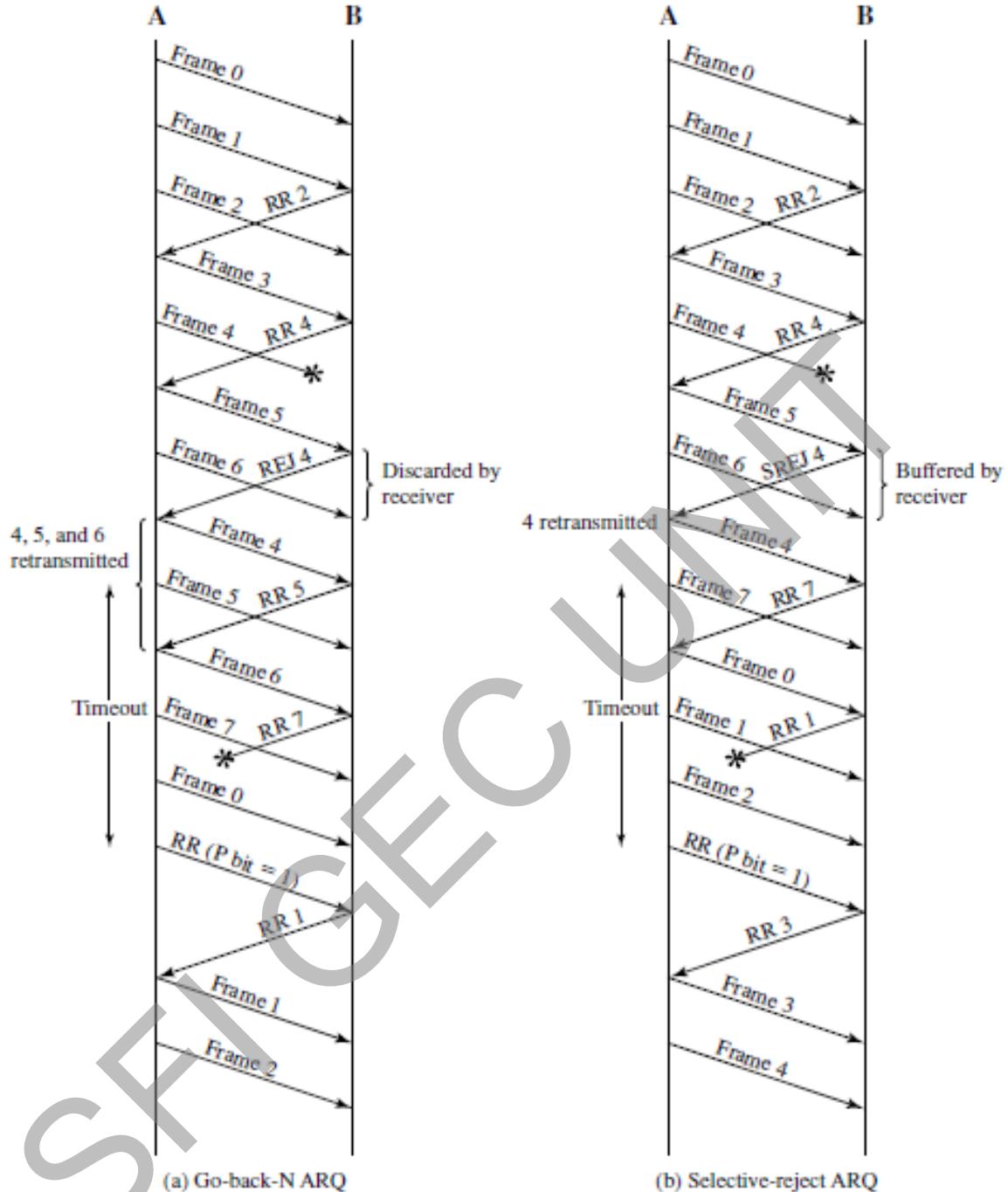
- receiver gets frame i, sends ack (i+1) which is lost
- acks are cumulative, so next ack (i+n) may arrive before transmitter times out on frame i
- if transmitter times out, it sends ack with P bit set
- can be repeated a number of times before a reset procedure is initiated

Damaged Rejection

- reject for damaged frame is lost
- handled as for lost frame when transmitter times out

2. Selective Reject is also called selective retransmission

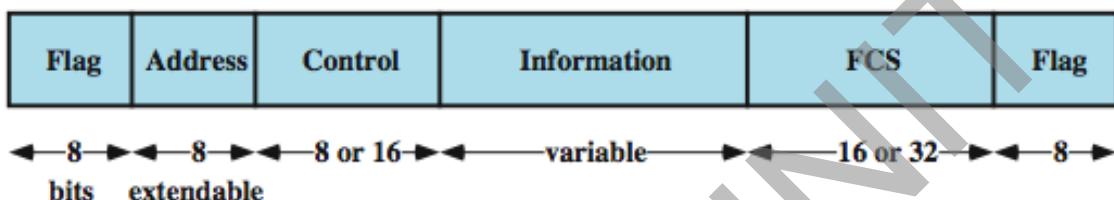
- only rejected frames are retransmitted
- subsequent frames are accepted by the receiver and buffered
- minimizes retransmission
- receiver must maintain large enough buffer
- more complex logic in transmitter
- hence less widely used
- useful for satellite links with long propagation delays



7. Explain the architecture and communication model of HDLC protocol.

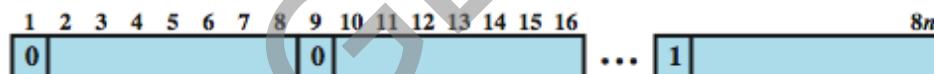
- High Level Data Link Control (HDLC) is an important data link control protocol
- specified as ISO 33009, ISO 4335
- station types:
 - Primary - controls operation of link
 - Secondary - under control of primary station
 - Combined - issues commands and responses
- link configurations
 - Unbalanced - 1 primary, multiple secondary
 - Balanced - 2 combined stations

- Normal Response Mode (NRM)
 - unbalanced config, primary initiates transfer
 - used on multi-drop lines, eg host + terminals
- Asynchronous Balanced Mode (ABM)
 - balanced config, either station initiates transmission, has no polling overhead, widely used
- Asynchronous Response Mode (ARM)
 - unbalanced config, secondary may initiate transmit without permission from primary, rarely used
- synchronous transmission of frames
- single frame format used



(a) Frame format

- Address field identifies secondary station that sent or will receive frame
 - usually 8 bits long
 - may be extended to multiples of 7 bits
 - LSB indicates if it is the last octet (1) or not (0)
 - all ones address 11111111 is broadcast



(b) Extended Address Field

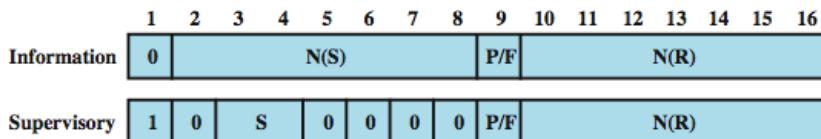
- Control Field is different for different frame type
 - Information - data transmitted to user (next layer up)
 - Flow and error control piggybacked on information frames
 - Supervisory - ARQ when piggyback not used
 - Unnumbered - supplementary link control
- first 1-2 bits of control field identify frame type

	1	2	3	4	5	6	7	8
I: Information	0		N(S)	P/F	N(R)			
S: Supervisory	1	0	S	P/F	N(R)			
U: Unnumbered	1	1	M	P/F	M			

N(S) = Send sequence number
 N(R) = Receive sequence number
 S = Supervisory function bits
 M = Unnumbered function bits
 P/F = Poll/final bit

(c) 8-bit control field format

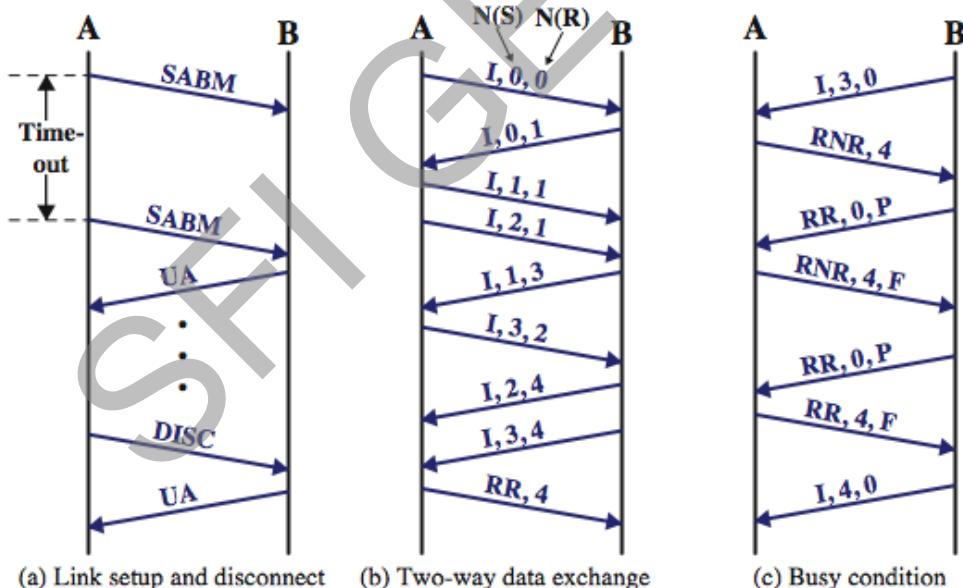
- use of Poll/Final bit depends on context
- in command frame is P bit set to 1 to solicit (poll) response from peer
- in response frame is F bit set to 1 to indicate response to soliciting command
- seq number usually 3 bits
 - can extend to 8 bits as shown below

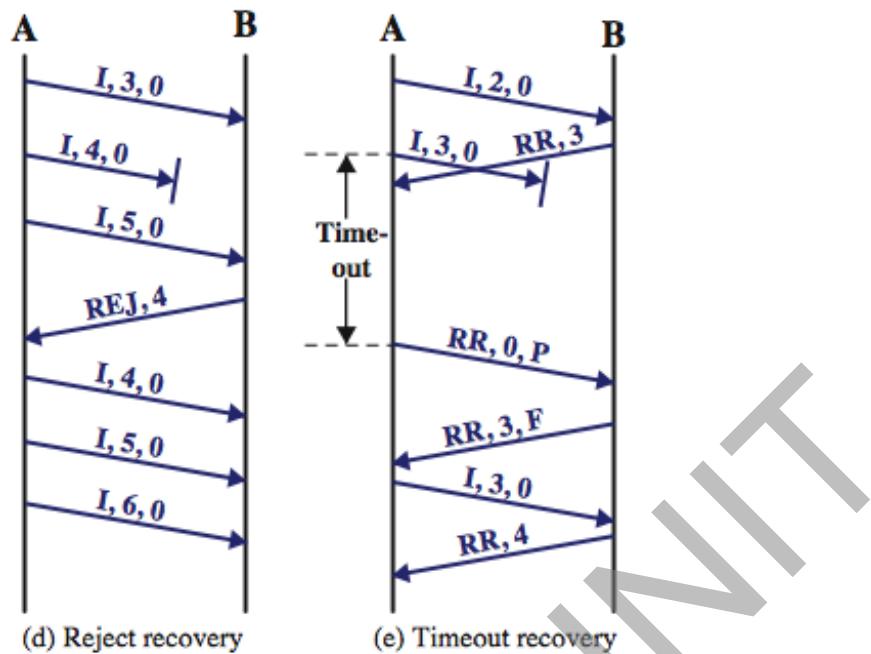


(d) 16-bit control field format

- Information Field
 - in information and some unnumbered frames
 - must contain integral number of octets
 - variable length
- Frame Check Sequence Field (FCS)
 - used for error detection
 - either 16 bit CRC or 32 bit CRC
- HDLC operation consist of exchange of information, supervisory and unnumbered frames
- have three phases
 - initialization
 - by either side, set mode & seq
 - data transfer
 - with flow and error control
 - using both I & S-frames (RR, RNR, REJ, SREJ)
 - disconnect
 - when requested or fault noted

Name	Command/ Response	Description
Information (I)	C/R	Exchange user data
Supervisory (S)		
Receive ready (RR)	C/R	Positive acknowledgment; ready to receive I-frame
Receive not ready (RNR)	C/R	Positive acknowledgment; not ready to receive
Reject (REJ)	C/R	Negative acknowledgment; go back N
Selective reject (SREJ)	C/R	Negative acknowledgment; selective reject
Unnumbered (U)		
Set normal response/extended mode (SNRM/SNRME)	C	Set mode; extended = 7-bit sequence numbers
Set asynchronous response/extended mode (SARM/SARME)	C	Set mode; extended = 7-bit sequence numbers
Set asynchronous balanced/extended mode (SABM, SABME)	C	Set mode; extended = 7-bit sequence numbers
Set initialization mode (SIM)	C	Initialize link control functions in addressed station
Disconnect (DISC)	C	Terminate logical link connection
Unnumbered Acknowledgment (UA)	R	Acknowledge acceptance of one of the set-mode commands
Disconnected mode (DM)	R	Responder is in disconnected mode
Request disconnect (RD)	R	Request for DISC command
Request initialization mode (RIM)	R	Initialization needed; request for SIM command
Unnumbered information (UI)	C/R	Used to exchange control information
Unnumbered poll (UP)	C	Used to solicit control information
Reset (RSET)	C	Used for recovery; resets N(R), N(S)
Exchange identification (XID)	C/R	Used to request/report status
Test (TEST)	C/R	Exchange identical information fields for testing
Frame reject (FRMR)	R	Report receipt of unacceptable frame





Part C (4x3=12)
(Answer All Questions. Each carries 3 marks)

8. Explain the different classes of IP addresses.

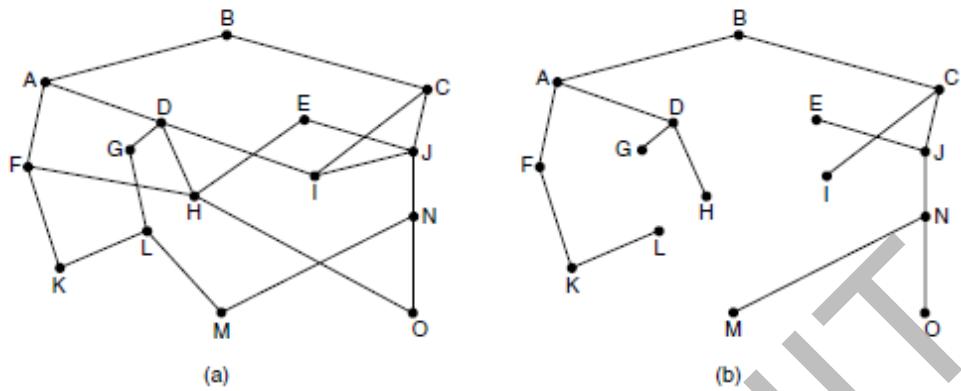
Class	32 Bits			Range of host addresses
	Network	Host		
A	0	Network	Host	1.0.0.0 to 127.255.255.255
B	10	Network	Host	128.0.0.0 to 191.255.255.255
C	110	Network	Host	192.0.0.0 to 223.255.255.255
D	1110	Multicast address		224.0.0.0 to 239.255.255.255
E	1111	Reserved for future use		240.0.0.0 to 255.255.255.255

Figure 5-53. IP address formats.

9. Explain optimality principle.

Optimality principle states that if router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route. To see this, call the part of the route from I to J r1 and the rest of the route r2. If a route better than r2 existed from J to K, it could be concatenated with r1 to improve the route from I to K, contradicting our statement that r1r2 is optimal. As a direct consequence of the optimality principle, we can see

that the set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a **sink tree** and is illustrated in Fig., where the distance metric is the number of hops. The goal of all routing algorithms is to discover and use the sink trees for all routers.



10. Explain the concept of flooding.

Flooding is the technique in which every incoming packet is sent out on every outgoing line except the one it arrived on. Flooding obviously generates vast numbers of duplicate packets, in fact, an infinite number unless some measures are taken to damp the process. One such measure is to have a hop counter contained in the header of each packet that is decremented at each hop, with the packet being discarded when the counter reaches zero. Ideally, the hop counter should be initialized to the length of the path from source to destination. If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the network. Flooding with a hop count can produce an exponential number of duplicate packets as the hop count grows and routers duplicate packets they have seen before. A better technique for damping the flood is to have routers keep track of which packets have been flooded, to avoid sending them out a second time.

11. Draw the header structure of IPv4 protocol and explain the fields.

The Version field keeps track of which version of the protocol the datagram belongs to. IHL, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, which limits the header to 60 bytes, and thus the Options field to 40 bytes. The Differentiated services field was intended to distinguish between different classes of service. The Total length includes everything in the datagram—both header and data. The maximum length is 65,535 bytes. The Identification field is needed to allow the destination host to determine which packet a newly arrived fragment belongs to. DF stands for Don't Fragment. It is an order to the routers not to fragment the packet. MF stands for More Fragments. All fragments except the last one have this bit set. The Fragment offset tells where in the current packet this fragment belongs. The TTL (Time to live) field is a counter used to limit packet lifetimes. The Protocol field tells it which transport process to give the packet to. Since the header carries vital information such as addresses, it rates its own checksum for protection, the Header checksum. The Security option tells how secret the information is. The Strict source routing option gives the complete path from source to destination as a sequence of IP addresses. The Loose source routing option requires the packet to traverse the list of routers specified, in the order specified, but it is allowed to pass through other routers on the way. The Record route option tells each router along the path to append its IP address to the Options field. Timestamp option is like the Record

route option, except that in addition to recording its 32-bit IP address, each router also records a 32-bit timestamp.

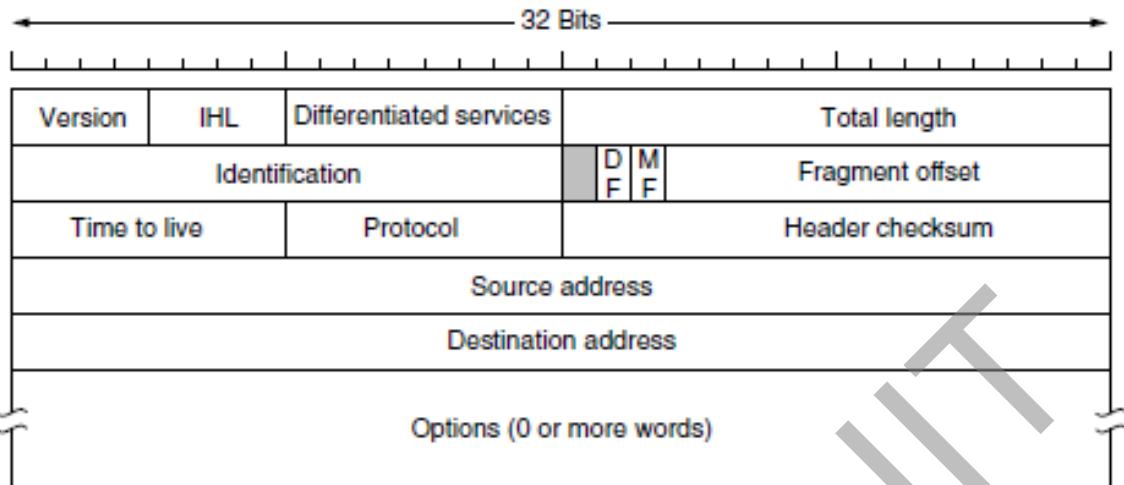


Figure 5-46. The IPv4 (Internet Protocol) header.

Part D

(2x9=18)

(Answer any two. Each carries 9 marks)

12. Discuss about the techniques to improve QoS in internetworking.

Quality of service mechanisms let a network with less capacity meet application requirements just as well at a lower cost. A stream of packets from a source to a destination is called a **flow**. A flow might be all the packets of a connection in a connection-oriented network, or all the packets sent from one process to another process in a connectionless network. The needs of each flow can be characterized by four primary parameters: bandwidth, delay, jitter, and loss. Together, these determine the **QoS (Quality of Service)** the flow requires. To accommodate a variety of applications, networks may support different categories of QoS. flow of data that enters the network.

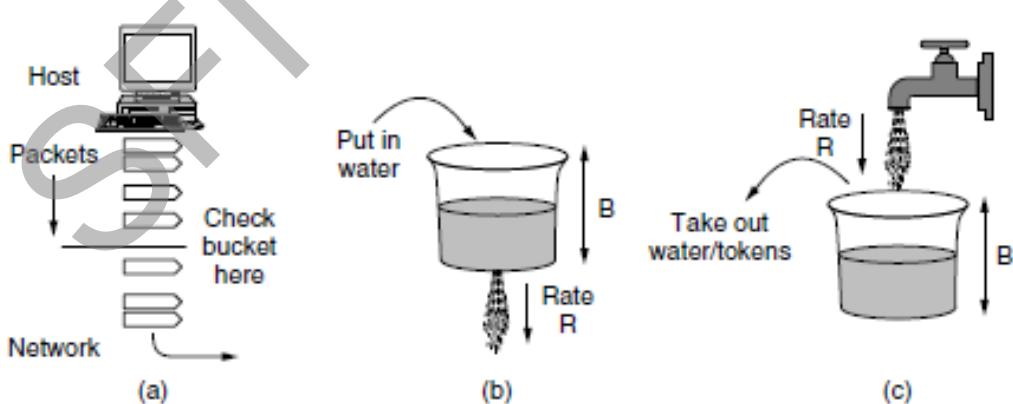
Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

The goal is to allow applications to transmit a wide variety of traffic that suits their needs, including some bursts, yet have a simple and useful way to describe the possible traffic patterns to the network. When a flow is set up, the user and the network agree on a certain traffic pattern (i.e., shape) for that flow. In effect, the customer says to the provider “My transmission pattern will look like this; can you handle it?” Sometimes this agreement is called an **SLA (Service Level Agreement)**, especially when it is made over aggregate flows and long periods of time, such as all of the traffic for a given customer. As long as the customer fulfills her part of the bargain and only sends packets according to the agreed-on contract, the provider promises to deliver them all in a timely fashion. Traffic shaping reduces congestion and thus helps the network live up to its promise.

Leaky and Token Buckets is a bucket with a small hole in the bottom, as illustrated in Fig. No matter the rate at which water enters the bucket, the outflow is at a constant rate, R , when there is any water in the bucket and zero when the bucket is empty. Also, once the bucket is full to capacity B , any additional water entering it spills over the sides and is lost.

This bucket can be used to shape or police packets entering the network, as shown in Fig. Conceptually, each host is connected to the network by an interface containing a leaky bucket. To send a packet into the network, it must be possible to put more water into the bucket. If a packet arrives when the bucket is full, the packet must either be queued until enough water leaks out to hold it or be discarded. The former might happen at a host shaping its traffic for the network as part of the operating system. The latter might happen in hardware at a provider network interface that is policing traffic entering the network. This technique was proposed by Turner and is called the **leaky bucket algorithm**.

A different but equivalent formulation is to imagine the network interface as a bucket that is being filled, as shown in Fig. 5-28(c). The tap is running at rate R and the bucket has a capacity of B , as before. Now, to send a packet we must be able to take water, or tokens, as the contents are commonly called, out of the bucket (rather than putting water into the bucket). No more than a fixed number of tokens, B , can accumulate in the bucket, and if the bucket is empty, we must wait until more tokens arrive before we can send another packet. This algorithm is called the **token bucket algorithm**.



Leaky and token buckets limit the long-term rate of a flow but allow short term bursts up to a maximum regulated length to pass through unaltered and without suffering any artificial delays. Large bursts will be smoothed by a leaky bucket traffic shaper to reduce congestion in the network. Using all of these buckets can be a bit tricky. When token buckets are used for

traffic shaping at hosts, packets are queued and delayed until the buckets permit them to be sent. When token buckets are used for traffic policing at routers in the network, the algorithm is simulated to make sure that no more packets are sent than permitted.

Packet Scheduling

Algorithms that allocate router resources among the packets of a flow and between competing flows are called **packet scheduling algorithms**. Three different kinds of resources can potentially be reserved for different flows:

1. Bandwidth.
2. Buffer space.
3. CPU cycles.

Packet scheduling algorithms allocate bandwidth and other router resources by determining which of the buffered packets to send on the output line next. We already described the most straightforward scheduler when explaining how routers work. Each router buffers packets in a queue for each output line until they can be sent, and they are sent in the same order that they arrived. This algorithm is known as **FIFO (First-In First-Out)**, or equivalently **FCFS (First-Come First-Serve)**. FIFO routers usually drop newly arriving packets when the queue is full. Since the newly arrived packet would have been placed at the end of the queue, this behavior is called **tail drop**. It is intuitive, and you may be wondering what alternatives exist. RED algorithm chose a newly arriving packet to drop at random when the average queue length grew large.

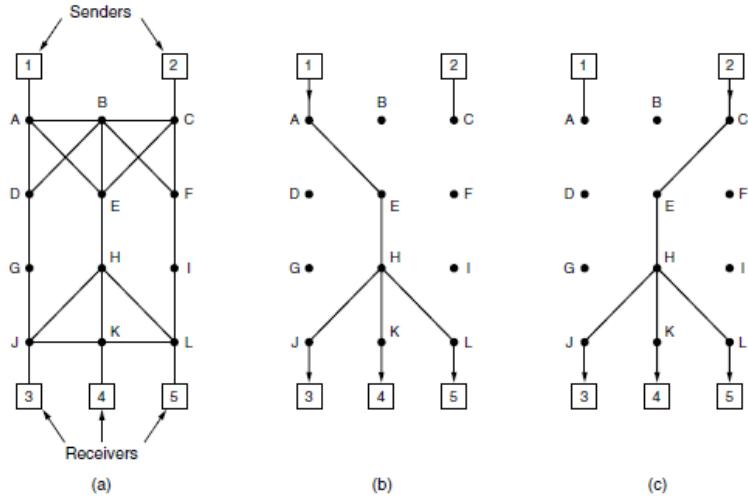
Admission Control

The user offers a flow with an accompanying QoS requirement to the network. The network then decides whether to accept or reject the flow based on its capacity and the commitments it has made to other flows. If it accepts, the network reserves capacity in advance at routers to guarantee QoS when traffic is sent on the new flow. The reservations must be made at all of the routers along the route that the packets take through the network. Any routers on the path without reservations might become congested, and a single congested router can break the QoS guarantee.

Integrated Services

RSVP—The Resource reSerVation Protocol

The main part of the integrated services architecture that is visible to the users of the network is **RSVP**. It is described in RFCs 2205–2210. This protocol is used for making the reservations; other protocols are used for sending the data. RSVP allows multiple senders to transmit to multiple groups of receivers, permits individual receivers to switch channels freely, and optimizes bandwidth use while at the same time eliminating congestion. In its simplest form, the protocol uses multicast routing using spanning trees. Each group is assigned a group address. To send to a group, a sender puts the group's address in its packets. The standard multicast routing algorithm then builds a spanning tree covering all group members. The routing algorithm is not part of RSVP. The only difference from normal multicasting is a little extra information that is multicast to the group periodically to tell the routers along the tree to maintain certain data structures in their memories.

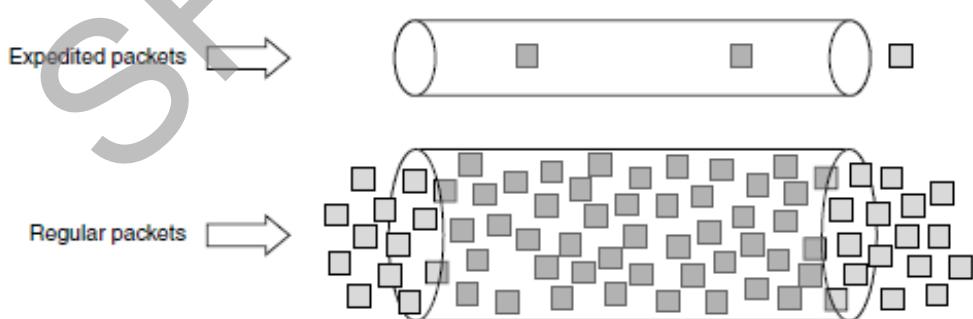


Differentiated Services

Differentiated services can be offered by a set of routers forming an administrative domain . The administration defines a set of service classes with corresponding forwarding rules. If a customer subscribes to differentiated services, customer packets entering the domain are marked with the class to which they belong. This information is carried in the Differentiated services field of IPv4 and IPv6 packets. The classes are defined as **per hop behaviors** because they correspond to the treatment the packet will receive at each router, not a guarantee across the network. Better service is provided to packets with some per-hop behaviors (e.g., premium service) than to others (e.g., regular service). Traffic within a class may be required to conform to some specific shape, such as a leaky bucket with some specified drain rate.

Expedited Forwarding

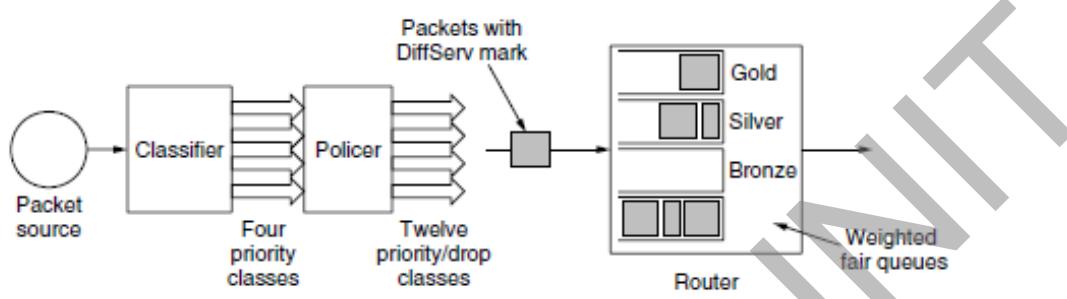
The idea behind expedited forwarding is very simple. Two classes of service are available: regular and expedited. The vast majority of the traffic is expected to be regular, but a limited fraction of the packets are expedited. The expedited packets should be able to transit the network as though no other packets were present. In this way they will get low loss, low delay and low jitter service.



Assured Forwarding

Assured forwarding specifies that there shall be four priority classes, each class having its own resources. The top three classes might be called gold, silver, and bronze. In addition, it defines three discard classes for packets that are experiencing congestion: low, medium,

and high. Taken together, these two factors define 12 service classes. Figure shows one way packets might be processed under assured forwarding. The first step is to classify the packets into one of the four priority classes. As before, this step might be done on the sending host or in the ingress router, and the rate of higher-priority packets may be limited by the operator as part of the service offering. The next step is to determine the discard class for each packet. This is done by passing the packets of each priority class through a traffic policer such as a token bucket. The policer lets all of the traffic through, but it identifies packets that fit within small bursts as low discard, packets that exceed small bursts as medium discard, and packets that exceed large bursts as high discard. The combination of priority and discard class is then encoded in each packet. Finally, the packets are processed by routers in the network with a packet scheduler that distinguishes the different classes.



13. Illustrate the routing procedure in mobile networks.

Design goals to be considered for mobile IP are

- Each mobile host must be able to use its home IP address anywhere.
- Software changes to the fixed hosts were not permitted.
- Changes to the router software and tables were not permitted.
- Most packets for mobile hosts should not make detours on the way.
- No overhead should be incurred when a mobile host is at home.

Mobile Node (MN)

- system (node) that can change the point of connection to the network without changing its IP address
- Home Agent (HA)
 - system in the home network of the MN, typically a router
 - registers the location of the MN, tunnels IP datagrams to the COA
- Foreign Agent (FA)
 - system in the current foreign network of the MN, typically a router
 - forwards the tunneled datagrams to the MN, typically also the default router for the MN
- Care-of Address (COA)
 - address of the current tunnel end-point for the MN (at FA or MN)
 - actual location of the MN from an IP point of view can be chosen, e.g., via DHCP

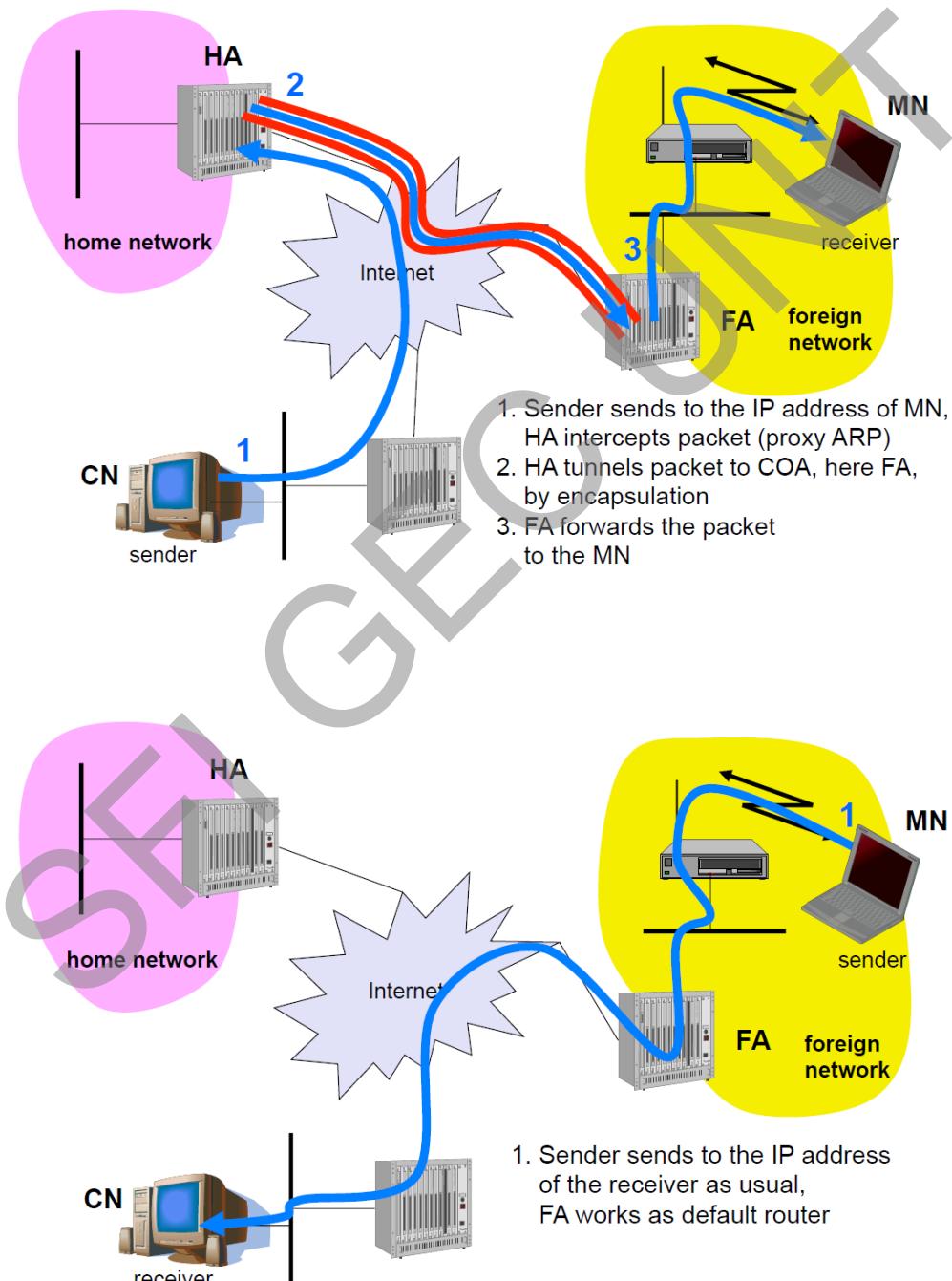
Correspondent Node (CN)

- communication partner

Agent Advertisement

- HA and FA periodically send advertisement messages into their physical subnets

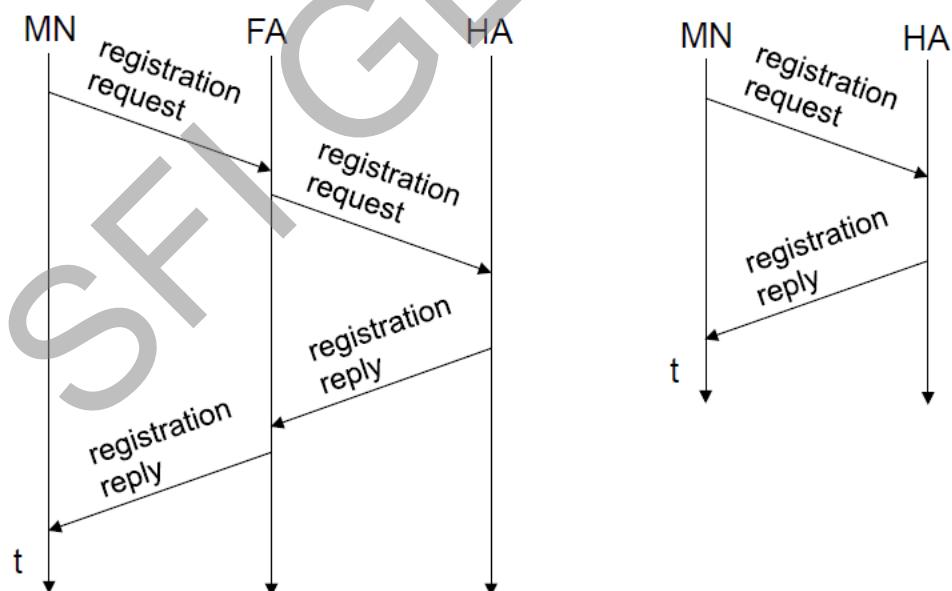
- MN listens to these messages and detects, if it is in the home or a foreign network
- MN reads a COA from the FA advertisement messages
- Registration (always limited lifetime!)
- MN signals COA to the HA via the FA, HA acknowledges via FA to MN
- these actions have to be secured by authentication
- Advertisement
 - HA advertises the IP address of the MN i.e. standard routing information
 - routers adjust their entries, these are stable for a longer time (HA responsible for a MN over a longer period of time)
 - packets to the MN are sent to the HA, • independent of changes in COA/FA



Agent Advertisement Frame Structure

0	7	8	15	16	23	24	31
type		code			checksum		
#addresses		addr. size			lifetime		
		router address 1					
		preference level 1					
		router address 2					
		preference level 2					
		...					
		type = 16	length		sequence number		
		registration lifetime	R B H F M G r T		reserved		
			COA 1				
			COA 2				

Registration Process



Encapsulation Frame Structure

ver.	IHL	DS (TOS)	length			
		IP identification	flags	fragment offset		
TTL		IP-in-IP	IP checksum			
IP address of HA						
Care-of address COA						
ver.	IHL	DS (TOS)	length			
		IP identification	flags	fragment offset		
TTL		lay. 4 prot.	IP checksum			
IP address of CN						
IP address of MN						
TCP/UDP/ ... payload						

14. Explain the various congestion control techniques.

The presence of congestion means that the load is (temporarily) greater than the resources (in a part of the network) can handle. Two solutions come to mind: increase the resources or decrease the load.

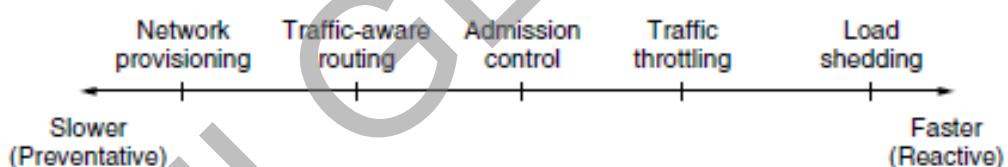


Figure 5-22. Timescales of approaches to congestion control.

Traffic-Aware Routing

The first approach we will examine is traffic-aware routing. The routing schemes we looked at in Sec 5.2 used fixed link weights. These schemes adapted to changes in topology, but not to changes in load. The goal in taking load into account when computing routes is to shift traffic away from hotspots that will be the first places in the network to experience congestion. The most direct way to do this is to set the link weight to be a function of the (fixed) link bandwidth and propagation delay plus the (variable) measured load or average queuing delay. Least-weight paths will then favor paths that are more lightly loaded, all else being equal. Internet routing protocols do not generally adjust their routes depending on the load. Instead, adjustments are made outside the routing protocol by slowly changing its inputs. This is called **traffic engineering**.

Admission Control

One technique that is widely used in virtual-circuit networks to keep congestion at bay is **admission control**. The idea is simple: do not set up a new virtual circuit unless the network can carry the added traffic without becoming congested. Thus, attempts to set up a virtual circuit may fail. This is better than the alternative, as letting more people in when the network is busy just makes matters worse. By analogy, in the telephone system, when a switch gets overloaded it practices admission control by not giving dial tones. Admission control can also be combined with traffic-aware routing by considering routes around traffic hotspots as part of the setup procedure.

Traffic Throttling

In the Internet and many other computer networks, senders adjust their transmissions to send as much traffic as the network can readily deliver. In this setting, the network aims to operate just before the onset of congestion. When congestion is imminent, it must tell the senders to throttle back their transmissions and slow down. This feedback is business as usual rather than an exceptional situation. The term **congestion avoidance** is sometimes used to contrast this operating point with the one in which the network has become (overly) congested. both datagram networks and virtual-circuit networks. Each approach must solve two problems. First, routers must determine when congestion is approaching, ideally before it has arrived. To do so, each router can continuously monitor the resources it is using. Three possibilities are the utilization of the output links, the buffering of queued packets inside the router, and the number of packets that are lost due to insufficient buffering. The second problem is that routers must deliver timely feedback to the senders that are causing the congestion. Congestion is experienced in the network, but relieving congestion requires action on behalf of the senders that are using the network. To deliver feedback, the router must identify the appropriate senders. It must then warn them carefully, without sending many more packets into the already congested network. Different schemes use different feedback mechanisms.

Choke Packets

The most direct way to notify a sender of congestion is to tell it directly. In this approach, the router selects a congested packet and sends a **choke packet** back to the source host, giving it the destination found in the packet. The original packet may be tagged (a header bit is turned on) so that it will not generate any more choke packets farther along the path and then forwarded in the usual way. To avoid increasing load on the network during a time of congestion, the router may only send choke packets at a low rate.

When the source host gets the choke packet, it is required to reduce the traffic sent to the specified destination, for example, by 50%. In a datagram network, simply picking packets at random when there is congestion is likely to cause choke packets to be sent to fast senders, because they will have the most packets in the queue. The feedback implicit in this protocol can help prevent congestion yet not throttle any sender unless it causes trouble. For the same reason, it is likely that multiple choke packets will be sent to a given host and destination. The host should ignore these additional chokes for the fixed time interval until its reduction in traffic takes effect. After that period, further choke packets indicate that the network is still congested.

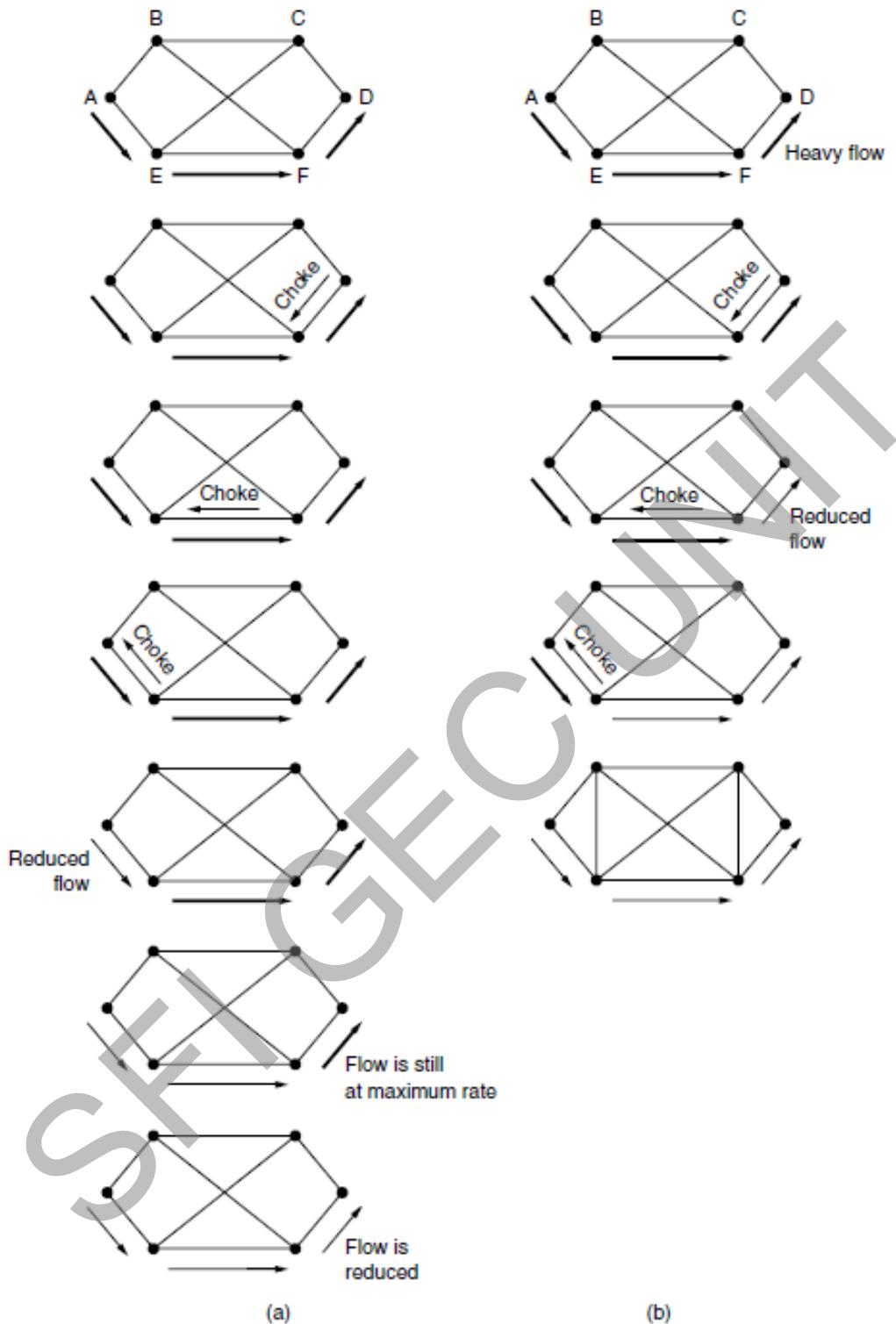


Figure 5-26. (a) A choke packet that affects only the source. (b) A choke packet that affects each hop it passes through.

Explicit Congestion Notification

Instead of generating additional packets to warn of congestion, a router can tag any packet it forwards (by setting a bit in the packet's header) to signal that it is experiencing congestion. When the network delivers the packet, the destination can note that there is congestion and inform the sender when it sends a reply packet. The sender can then throttle its transmissions as before. This design is called **ECN (Explicit Congestion Notification)** and is used in the Internet.

Load shedding

It is a fancy way of saying that when routers are being inundated by packets that they cannot handle, they just throw them away. The term comes from the world of electrical power generation, where it refers to the practice of utilities intentionally blacking out certain areas to save the entire grid from collapsing on hot summer days when the demand for electricity greatly exceeds the supply. The key question for a router drowning in packets is which packets to drop. The preferred choice may depend on the type of applications that use the network.

Random Early Detection

By having routers drop packets early, before the situation has become hopeless, there is time for the source to take action before it is too late. A popular algorithm for doing this is called **RED (Random Early Detection)**. To determine when to start discarding, routers maintain a running average of their queue lengths. When the average queue length on some link exceeds a threshold, the link is said to be congested and a small fraction of the packets are dropped at random. Picking packets at random makes it more likely that the fastest senders will see a packet drop; this is the best option since the router cannot tell which source is causing the most trouble in a datagram network. The affected sender will notice the loss when there is no acknowledgement, and then the transport protocol will slow down. The lost packet is thus delivering the same message as a choke packet, but implicitly, without the router sending any explicit signal. RED routers improve performance compared to routers that drop packets only when their buffers are full, though they may require tuning to work well.

15. Explain Link State routing algorithm.

The idea behind link state routing is fairly simple and can be stated as five parts. Each router must do the following things to make it work:

1. Discover its neighbors and learn their network addresses.
2. Set the distance or cost metric to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to and receive packets from all other routers.
5. Compute the shortest path to every other router.

In effect, the complete topology is distributed to every router. Then Dijkstra's algorithm can be run at each router to find the shortest path to every other router.

Learning about the Neighbors

When a router is booted, its first task is to learn who its neighbors are. It accomplishes this goal by sending a special HELLO packet on each point-to-point line. The router on the other end is expected to send back a reply giving its name. These names must be globally unique because when a distant router later hears that three routers are all connected to F, it is essential that it can determine whether all three mean the same F.

Setting Link Costs

The link state routing algorithm requires each link to have a distance or cost metric for finding shortest paths. The cost to reach neighbors can be set automatically, or configured by the network operator. If the network is geographically spread out, the delay of the links may be factored into the cost so that paths over shorter links are better choices. The most direct way to determine this delay is to send over the line a special ECHO packet that the other side is required to send back immediately. By measuring the round-trip time and dividing it by two, the sending router can get a reasonable estimate of the delay.

Building Link State Packets

Once the information needed for the exchange has been collected, the next step is for each router to build a packet containing all the data. The packet starts with the identity of the sender, followed by a sequence number and a list of neighbors. The cost to each neighbor is also given.

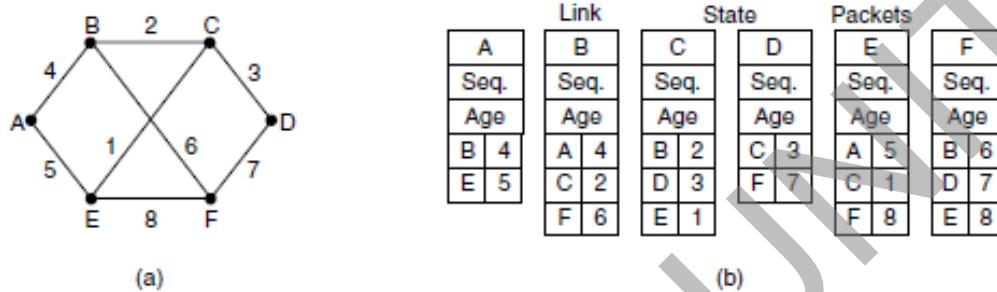


Figure 5-12. (a) A network. (b) The link state packets for this network.

Distributing the Link State Packets

The fundamental idea is to use flooding to distribute the link state packets to all routers. To keep the flood in check, each packet contains a sequence number that is incremented for each new packet sent. Routers keep track of all the (source router, sequence) pairs they see. When a new link state packet comes in, it is checked against the list of packets already seen. If it is new, it is forwarded on all lines except the one it arrived on. If it is a duplicate, it is discarded. If a packet with a sequence number lower than the highest one seen so far ever arrives, it is rejected as being obsolete as the router has more recent data.

Computing the New Routes

Once a router has accumulated a full set of link state packets, it can construct the entire network graph because every link is represented. Every link is, in fact, represented twice, once for each direction. The different directions may even have different costs. The shortest-path computations may then find different paths from router A to B than from router B to A.

Now Dijkstra's algorithm can be run locally to construct the shortest paths to all possible destinations. The results of this algorithm tell the router which link to use to reach each destination. This information is installed in the routing tables, and normal operation is resumed. Compared to distance vector routing, link state routing requires more memory

and computation. For a network with n routers, each of which has k neighbors, the memory required to store the input data is proportional to kn , which is at least as large as a routing table listing all the destinations. Also, the computation time grows faster than kn , even with the most efficient data structures, an issue in large networks.

Part E

(4x10=40)

(Answer any four. Each carries 10 marks)

16. Illustrate the working of OSPF protocol.

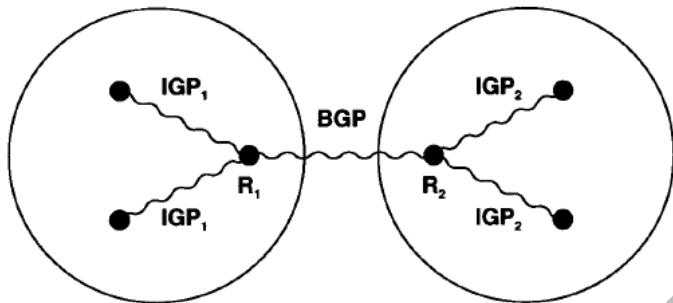


Figure 16.3 Conceptual view of two autonomous systems each using its own IGP internally, but using BGP to communicate between an exterior router and the other system.

- Interior Gateway Protocol (IGP) as a generic description that refers to any algorithm that interior routers use when they exchange network reachability and routing information. Once the reachability information for an entire autonomous system has been assembled, one of the routers in the system can advertise it to other autonomous systems using an Exterior Gateway Protocol.
- Link state routing algorithm, which uses SPF to compute shortest paths, scales better than a distance-vector algorithm. IETF has designed an IGP that uses the link state algorithm called Open SPF (OSPF), the new protocol tackles several ambitious goals.
 1. Making it an open standard that anyone can implement without paying license fees has encouraged many vendors to support OSPF.
 2. OSPF includes type of service routing.
 3. OSPF provides load balancing.
 4. To permit growth and make the networks at a site easier to manage, OSPF allows a site to partition its networks and routers into subsets called areas. Each area is self-contained; knowledge of an area's topology remains hidden from other areas.
 5. The OSPF protocol specifies that all exchanges between routers can be authenticated.
 6. OSPF includes support for host-specific, subnet-specific, and classless routes as well as classful network-specific routes.
 7. To accommodate multi-access networks like Ethernet, OSPF extends the SPF algorithm
 8. To permit maximum flexibility, OSPF allows managers to describe a virtual network topology that abstracts away from the details of physical connections.
 9. OSPF allows routers to exchange routing information learned from other (external) sites.

OSPF Message Format

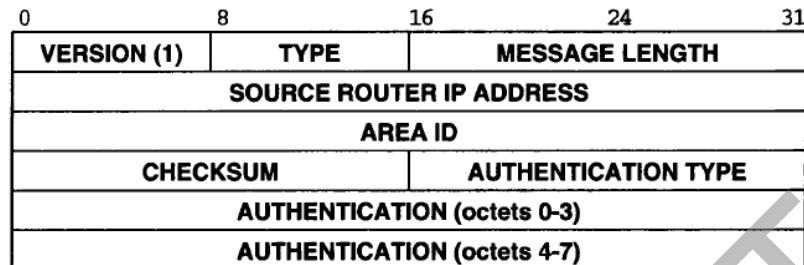


Figure 16.7 The fixed 24-octet OSPF message header.

Type	Meaning
1	Hello (used to test reachability)
2	Database description (topology)
3	Link status request
4	Link status update
5	Link status acknowledgement

- VERSION specifies the version of the protocol.
- TYPE identifies the message type
- SOURCE ROUTER IP ADDRESS gives the address of the sender
- AREA ID gives the 32-bit identification number for the area.
- AUTHENTICATION TYPE specifies which authentication scheme is used (currently, 0 means no authentication and 1 means a simple password is used).

OSPF Hello Message Format

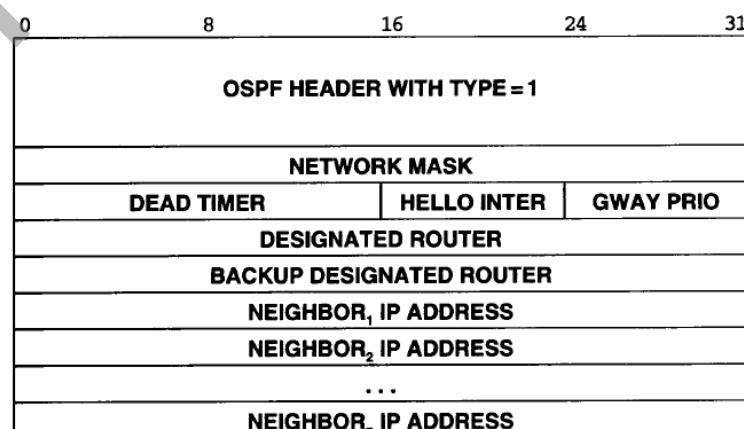


Figure 16.8 OSPF *hello* message format. A pair of neighbor routers exchanges these messages periodically to test reachability.

- NETWORK MASK contains a mask for the network over which the message has been sent
- DEAD TIMER gives a time in seconds after which a nonresponding neighbor is considered dead.
- HELLOPO INTER is the normal period, in seconds, between hello messages.
- GWAYPRIO is the integer priority of this router, and is used in selecting a backup designated router.
- DESIGNATED ROUTER and BACKUP DESIGNATEDR OUTER contain IP addresses that give the sender's view of the designated router and backup designated router for the network over which the message is sent.
- NEIGHBOR, IP ADDRESS give the IP addresses of all neighbors from which the sender has recently received hello messages.

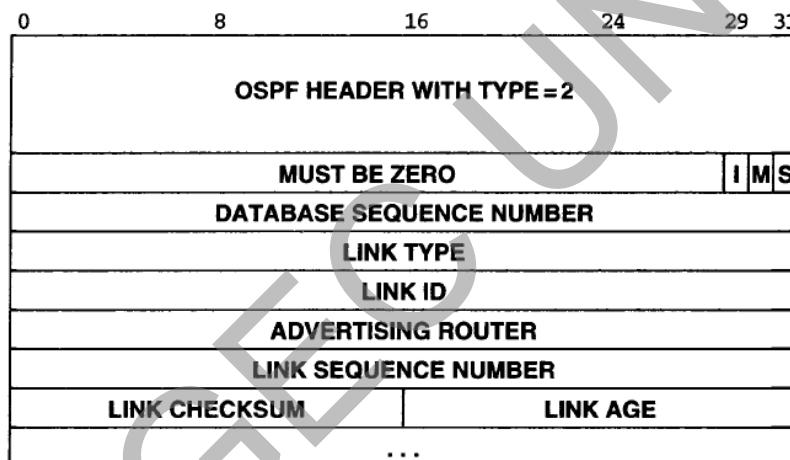


Figure 16.9 OSPF *database description* message format. The fields starting at *LINK TYPE* are repeated for each link being specified.

Link Type	Meaning
1	Router link
2	Network link
3	Summary link (IP network)
4	Summary link (link to border router)
5	External link (link to another site)

- LINK TYPE through LINK AGE describe one link in the network topology; they are repeated for each link.
- LINK ID gives an identification for the link (IP address of a router or network)
- ADVERTISING ROUTER specifies the address of the router advertising this link
- LINK SEQUENCE NUMBER contains an integer generated by that router to ensure that messages are not missed or received out of order.

- LINK CHECKSUM provides further assurance that the link information has not been corrupted.
- LINK AGE also helps order messages it gives the time in seconds since the link was established.

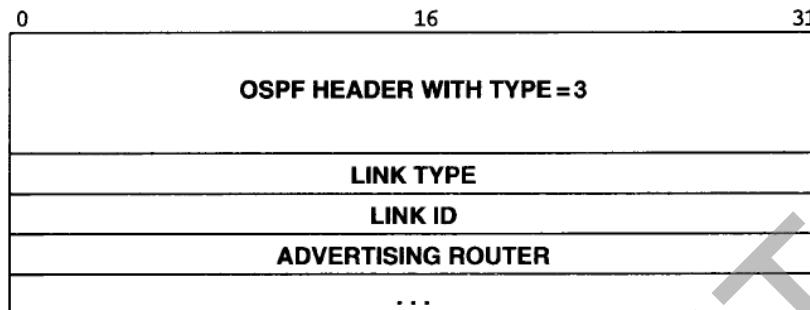


Figure 16.10 OSPF *link status request* message format. A router sends this message to a neighbor to request current information about a specific set of links.

- After exchanging database description messages with a neighbor, a router may discover that parts of its database are out of date.
- To request that the neighbor supply updated information, the router sends a link status request message.
- The neighbor responds with the most current information it has about those links.
- More than one request message may be needed if the list of requests is long.

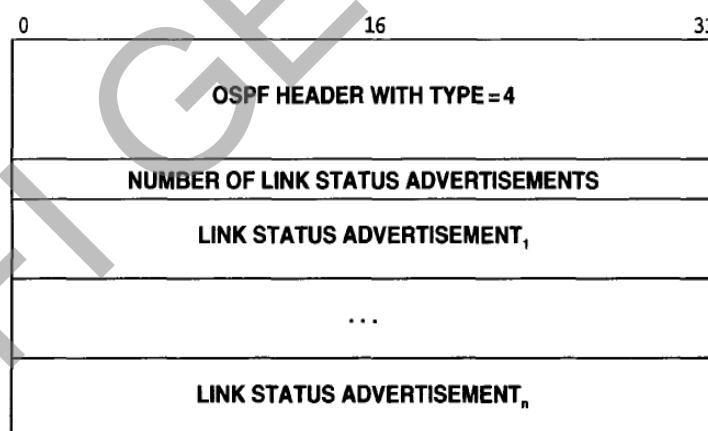


Figure 16.11 OSPF *link status update* message format. A router sends such a message to broadcast information about its directly connected links to all other routers.

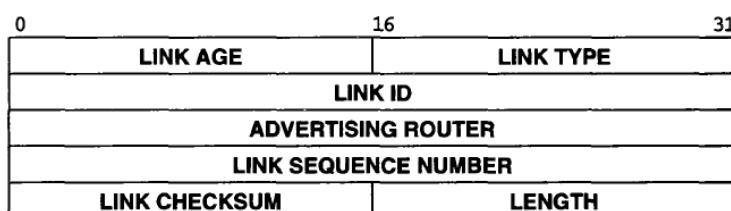


Figure 16.12 The format of the header used for all link status advertisements.

- Routers broadcast the status of links with a link status update message.
- LINK TYPE field in the link status header specifies which of the formats has been used. Thus, a router that receives a link status update message knows exactly which of the described destinations lie inside the site and which are external.

17. Explain how control messages are sent using ICMP protocol.

- To allow routers in an internet to report errors or provide information about unexpected circumstances, Internet Control Message Protocol (ICMP), must be included in every IP implementation.
- ICMP messages travel across the internet in the data portion of IP datagrams.
- The ultimate destination of an ICMP message is not an application program or user on the destination machine but the Internet Protocol software on that machine.
- ICMP error message arrives, the ICMP software module handles it.
- When a datagram causes an error, ICMP can only report the error condition back to the original source of the datagram; the source must relate the error to an individual program or take other action to correct the problem.

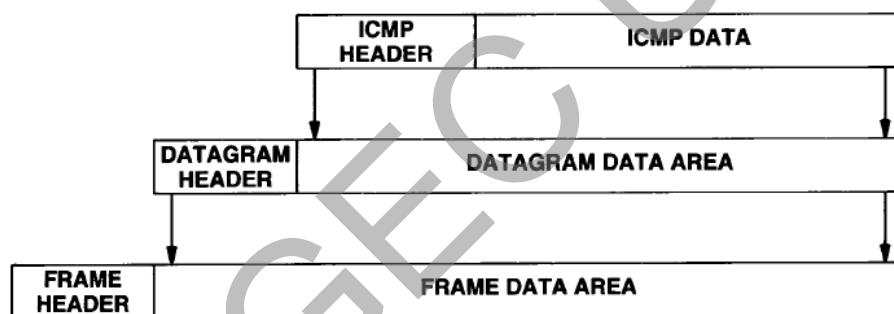


Figure 9.1 Two levels of ICMP encapsulation. The ICMP message is encapsulated in an IP datagram, which is further encapsulated in a frame for transmission. To identify ICMP, the datagram protocol field contains the value 1.

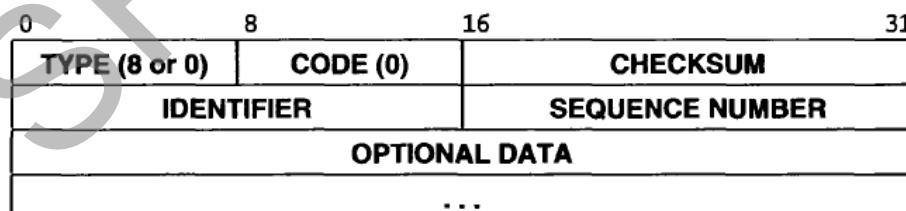


Figure 9.2 ICMP echo request or reply message format.

Type Field	ICMP Message Type
0	Echo Reply
3	Destination Unreachable
4	Source Quench
5	Redirect (change a route)
8	Echo Request
9	Router Advertisement
10	Router Solicitation
11	Time Exceeded for a Datagram
12	Parameter Problem on a Datagram
13	Timestamp Request
14	Timestamp Reply
15	Information Request (obsolete)
16	Information Reply (obsolete)
17	Address Mask Request
18	Address Mask Reply

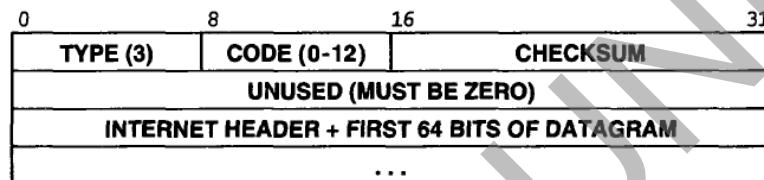


Figure 9.3 ICMP destination unreachable message format.

- Whenever an error prevents a router from routing or delivering a datagram, the router sends a destination unreachable message back to the source and then **drops (i.e., discards) the datagram**.
- Network unreachable errors usually imply routing failures**
- Host unreachable errors imply delivery failure.**
- Destinations may be unreachable because hardware is temporarily out of service, because the sender specified a nonexistent destination address, or because the router does not have a route to the destination network.

Code Value	Meaning
0	Network unreachable
1	Host unreachable
2	Protocol unreachable
3	Port unreachable
4	Fragmentation needed and DF set
5	Source route failed
6	Destination network unknown
7	Destination host unknown
8	Source host isolated
9	Communication with destination network administratively prohibited
10	Communication with destination host administratively prohibited
11	Network unreachable for type of service
12	Host unreachable for type of service

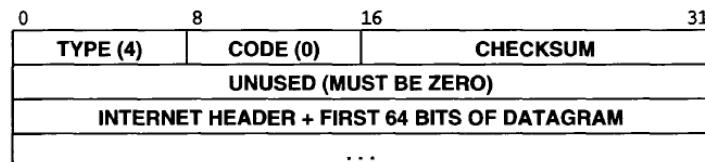


Figure 9.4 ICMP source quench message format. A congested router sends one source quench message each time it discards a datagram; the datagram prefix identifies the datagram that was dropped.

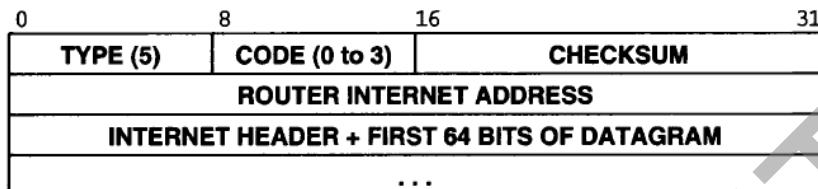


Figure 9.6 ICMP redirect message format.

- ROUTER INTERNET ADDRESS field contains the address of a router that the host is to use to reach the destination mentioned in the datagram header.
- The INTERNETHEADER field contains the IP header plus the next 64 bits of the datagram that triggered the message. Thus, a host receiving an ICMP redirect examines the datagram prefix to determine the datagram's destination address.
- The CODE field of an ICMP redirect message further specifies how to interpret the destination address, based on values assigned as follows:

Code Value	Meaning
0	Redirect datagrams for the Net (now obsolete)
1	Redirect datagrams for the Host
2	Redirect datagrams for the Type of Service† and Net
3	Redirect datagrams for the Type of Service and Host

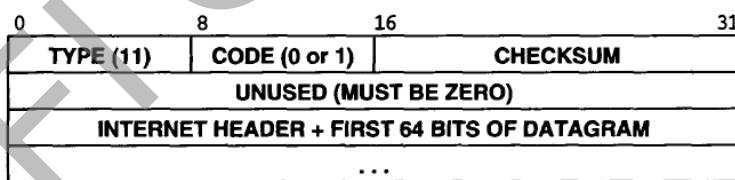


Figure 9.7 ICMP time exceeded message format. A router sends this message whenever a datagram is discarded because the time-to-live field in the datagram header has reached zero or because its reassembly timer expired while waiting for fragments.

- To prevent datagrams from circling forever in a TCP/IP internet, each IP datagram contains a time-to-live counter, sometimes called a hop count.
- A router decrements the time-to-live counter whenever it processes the datagram and discards the datagram when the count reaches zero.
- Whenever a router discards a datagram because its hop count has reached zero or because a timeout occurred while waiting for fragments of a datagram, it sends an ICMP time exceeded message back to the datagram's source.

0	8	16	31
TYPE (12)	CODE (0 or 1)	CHECKSUM	
POINTER		UNUSED (MUST BE ZERO)	
		INTERNET HEADER + FIRST 64 BITS OF DATAGRAM	
			...

Figure 9.8 ICMP parameter problem message format. Such messages are only sent when the problem causes the datagram to be dropped.

When a router or host finds problems with a datagram not covered by previous ICMP error messages, it sends a parameter problem message to the original source. One possible cause of such problems occurs when arguments to an option are incorrect. It is only sent when the problem is so severe that the datagram must be discarded. Sender uses the POINTER field in the message header to identify the octet in the datagram that caused the problem. Code 1 is used to report that a required option is missing . POINTER field is not used for code 1.

0	8	16	31
TYPE (13 or 14)	CODE (0)	CHECKSUM	
	IDENTIFIER	SEQUENCE NUMBER	
	ORIGINATE TIMESTAMP		
	RECEIVE TIMESTAMP		
	TRANSMIT TIMESTAMP		

Figure 9.9 ICMP timestamp request or reply message format.

ICMP message to obtain the time from another machine and used to synchronize clocks . A requesting machine sends an ICMP timestamp request message to another machine, asking that the second machine return its current value for the time of day. The receiving machine returns a timestamp reply back to the machine making the request. TYPE field identifies the message as a request (13) or a reply (14); the IDENTIFIER and SEQUENCE NUMBER fields are used by the source to associate replies with requests. Remaining fields specify times, given in milliseconds since midnight, Universal Time. The ORIGINATE TIMESTAMP field is filled in by the original sender just before the packet is transmitted, the RECEIVE TIMESTAMP field is filled immediately upon receipt of a request, and the TRANSMIT TIMESTAMP field is filled immediately before the reply is transmitted. Hosts use the three timestamp fields to compute estimates of the delay time between them and to synchronize their clocks. Because the reply includes the ORIGINATE TIMESTAMP field, a host can compute the total time required for a request to travel to a destination, because the reply carries both the time at which the request entered the remote machine, as well as the time at which the reply left, the host can compute the network transit time, and from that, estimate the differences in remote and local clocks. The ICMP information request and information reply messages (types 15 and 16) are now considered obsolete and should not be used. They were originally intended to allow hosts to discover their internet address at system startup.

0	8	16	31
TYPE (17 or 18)	CODE (0)	CHECKSUM	
IDENTIFIER	SEQUENCE NUMBER		
ADDRESS MASK			

Figure 9.10 ICMP address mask request or reply message format. Usually, hosts broadcast a request without knowing which specific router will respond.

To participate in subnet addressing, a host needs to know which bits of the 32-bit internet address correspond to the physical network and which correspond to host identifiers. The information needed to interpret the address is represented in a 32-bit quantity called the subnet mask. To learn the subnet mask used for the local network, a machine can send an address mask request message to a router and receive an address mask reply. The machine making the request can either send the message directly, if it knows the router's address, or broadcast the message if it does not. The ICMP router discovery scheme helps in two ways. First, instead of providing a statically configured router address via a bootstrap protocol, the scheme allows a host to obtain information directly from the router itself. Second, the mechanism uses a softstate technique with timers to prevent hosts from retaining a route after a router crashes. Routers advertise their information periodically, and a host discards a route if the timer for a route expires. Besides the TYPE, CODE, and CHECKSUM fields, the message contains a field NUM ADDRS that specifies the number of address entries which follow (often 1), an ADDR SIZE field that specifies the size of an address in 32-bit units, and a LIFETIME field that specifies the time in seconds a host may use the advertised address(es). The default value for LIFETIME is 30 minutes, and the default value for periodic retransmission is 10 minutes, which means that a host will not discard a route if the host misses a single advertisement message. The remainder of the message consists of NUM ADDRS pairs of fields, where each pair contains a ROUTER ADDRESS and an integer PRECEDENCE LEVEL for the route. The precedence value is a two's complement integer; a host chooses the route with highest precedence. If the router and the network support multicast, a router multicast ICMP router advertisement messages to the all-systems multicast address (i.e., 224.0.0.1). If not, the router sends the messages to the limited broadcast address (i.e., the all 1's address). Of course, a host must never send a router advertisement message.

0	8	16	31
TYPE (10)	CODE (0)	CHECKSUM	
RESERVED			

Figure 9.12 ICMP router solicitation message. A host sends a solicitation after booting to request that routers on the local net immediately respond with an ICMP router advertisement.

Although the designers provided a range of values to be used as the delay between successive router advertisements, they chose the default of 10 minutes. The value was selected as a compromise between rapid failure detection and low overhead. A smaller value would allow more rapid detection of router failure, but would increase network traffic; a larger value would decrease traffic, but would delay failure detection. One of the issues the designers considered was how to accommodate a large number of routers on the same network. From the point of view of a host, the default delay has a severe disadvantage: a host cannot afford to wait many minutes for an advertisement when it first boots. To avoid such delays, the designers included an ICMP router solicitation message that allows a host to request an immediate advertisement.

18 a. Give the format of TCP header and discuss the relevance of various fields.

TCP (Transmission Control Protocol) was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork. The sending and receiving TCP entities exchange data in the form of segments. A **TCP segment** consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes. The TCP software decides how big segments should be. It can accumulate data from several writes into one segment or can split data from one write over multiple segments. Two limits restrict the segment size. First, each segment, including the TCP header, must fit in the 65,515- byte IP payload. Second, each link has an **MTU (Maximum Transfer Unit)**. Each segment must fit in the MTU at the sender and receiver so that it can be sent and received in a single, unfragmented packet. In practice, the MTU is generally 1500 bytes (the Ethernet payload size) and thus defines the upper bound on segment size.

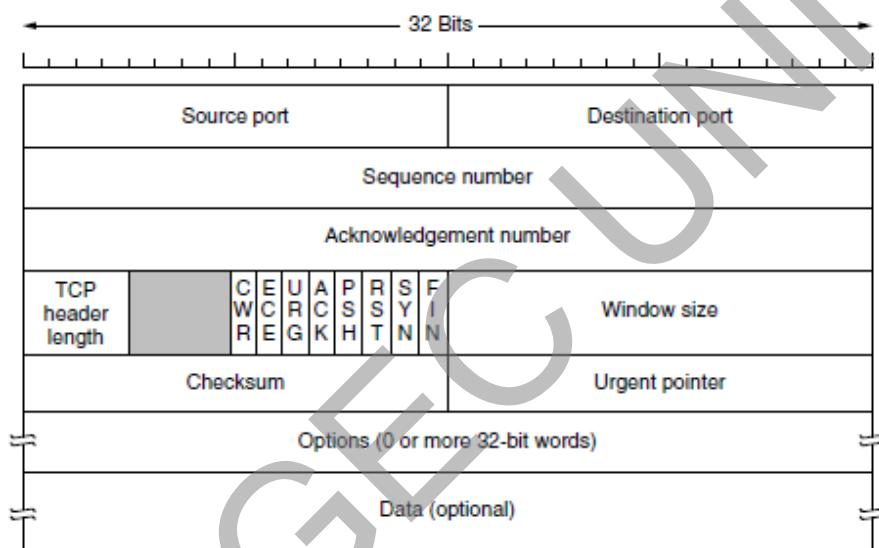


Figure 6-36. The TCP header.

The basic protocol used by TCP entities is the sliding window protocol with a dynamic window size. When a sender transmits a segment, it also starts a timer. When the segment arrives at the destination, the receiving TCP entity sends back a segment (with data if any exist, and otherwise without) bearing an acknowledgement number equal to the next sequence number it expects to receive and the remaining window size. If the sender's timer goes off before the acknowledgement is received, the sender transmits the segment again. Every segment begins with a fixed-format, 20-byte header. The fixed header may be followed by header options. After the options, if any, up to 65,495 data bytes may follow, where the first 20 refer to the IP header and the second to the TCP header. Segments without any data are legal and are commonly used for acknowledgements and control messages. The Source port and Destination port fields identify the local end points of the connection. A TCP port plus its host's IP address forms a 48-bit unique end point. The source and destination end points together identify the connection. This connection identifier is called a **5 tuple** because it consists of five pieces of information: the protocol (TCP), source IP and source port, and destination IP and destination port. The Sequence number and Acknowledgement number fields perform their usual functions. Note that the latter specifies the next in-order byte expected, not the last byte correctly received. It is a **cumulative acknowledgement** because it

summarizes the received data with a single number. It does not go beyond lost data. Both are 32 bits because every byte of data is numbered in a TCP stream. The TCP header length tells how many 32-bit words are contained in the TCP header. This information is needed because the Options field is of variable length, so the header is, too. Technically, this field really indicates the start of the data within the segment, measured in 32-bit words, but that number is just the header length in words, so the effect is the same.

Next comes a 4-bit field that is not used. CWR and ECE are used to signal congestion when ECN (Explicit Congestion Notification) is used, as specified in RFC 3168.

ECE is set to signal an ECN-Echo to a TCP sender to tell it to slow down when the TCP receiver gets a congestion indication from the network. CWR is set to signal Congestion Window Reduced from the TCP sender to the TCP receiver so that it knows the sender has slowed down and can stop sending the ECN-Echo. URG is set to 1 if the Urgent pointer is in use. The Urgent pointer is used to indicate a byte offset from the current sequence number at which urgent data are to be found. This facility is in lieu of interrupt messages. As we mentioned above, this facility is a bare-bones way of allowing the sender to signal the receiver without getting TCP itself involved in the reason for the interrupt, but it is

seldom used. The ACK bit is set to 1 to indicate that the Acknowledgement number is valid. This is the case for nearly all packets. If ACK is 0, the segment does not contain an acknowledgement, so the Acknowledgement number field is ignored. The PSH bit indicates PUSHed data. The receiver is hereby kindly requested to deliver the data to the application upon arrival and not buffer it until a full buffer has been received (which it might otherwise do for efficiency). The RST bit is used to abruptly reset a connection that has become confused due to a host crash or some other reason. It is also used to reject an invalid segment or refuse an attempt to open a connection. In general, if you get a segment with the RST bit on, you have a problem on your hands. The SYN bit is used to establish connections. The connection request has SYN = 1 and ACK = 0 to indicate that the piggyback acknowledgement field is not in use. The connection reply does bear an acknowledgement, however, so it has SYN = 1 and ACK = 1. In essence, the SYN bit is used to denote both CONNECTION REQUEST and CONNECTION ACCEPTED, with the ACK bit used to distinguish between those two possibilities. The FIN bit is used to release a connection. It specifies that the sender has no more data to transmit. However, after closing a connection, the closing process may continue to receive data indefinitely. Both SYN and FIN segments have sequence numbers and are thus guaranteed to be processed in the correct order.

Flow control in TCP is handled using a variable-sized sliding window. The Window size field tells how many bytes may be sent starting at the byte acknowledged. A Window size field of 0 is legal and says that the bytes up to and including Acknowledgement number – 1 have been received, but that the receiver has not had a chance to consume the data and would like no more data for the moment. The receiver can later grant permission to send by transmitting a segment with the same Acknowledgement number and a nonzero Window size field.

The Options field provides a way to add extra facilities not covered by the regular header. Many options have been defined and several are commonly used. The options are of variable length, fill a multiple of 32 bits by using padding with zeros, and may extend to 40 bytes to accommodate the longest TCP header that can be specified. Some options are carried when a connection is established to negotiate or inform the other side of capabilities. Other options are carried on packets during the lifetime of the connection. Each option has a Type-Length-Value encoding. A widely used option is the one that allows each host to specify the **MSS** (**Maximum Segment Size**) it is willing to accept. The **window scale** option allows the sender and receiver to negotiate a window scale factor at the start of a connection. Both sides use the scale factor to shift the Window size field up to 14 bits to the left, thus allowing windows of

up to 230 bytes. Most TCP implementations support this option. The **timestamp** option carries a timestamp sent by the sender and echoed by the receiver. It is included in every packet, once its use is established during connection setup, and used to compute round-trip time samples that are used to estimate when a packet has been lost. It is also used as a logical extension of the 32-bit sequence number. On a fast connection, the sequence number may wrap around quickly, leading to possible confusion between old and new data. The **PAWS (Protection Against Wrapped Sequence numbers)** scheme discards arriving segments with old timestamps to prevent this problem. Finally, the **SACK (Selective ACKnowledgement)** option lets a receiver tell a sender the ranges of sequence numbers that it has received. It supplements the Acknowledgement number and is used after a packet has been lost but subsequent (or duplicate) data has arrived. The new data is not reflected by the Acknowledgement number field in the header because that field gives only the next in-order byte that is expected. With SACK, the sender is explicitly aware of what data the receiver has and hence can determine what data should be retransmitted.

18 b. How transport layer connection is established in TCP? Illustrate with state diagrams.

TCP uses this three-way handshake to establish connections. This establishment protocol involves one peer checking with the other that the connection request is indeed current. The normal setup procedure when host 1 initiates is shown in Fig. 6-11. Host 1 chooses a sequence number, x , and sends a CONNECTION REQUEST segment containing it to host 2. Host 2 replies with an ACK segment acknowledging x and announcing its own initial sequence number, y . Finally, host 1 acknowledges host 2's choice of an initial sequence number in the first data segment that it sends.

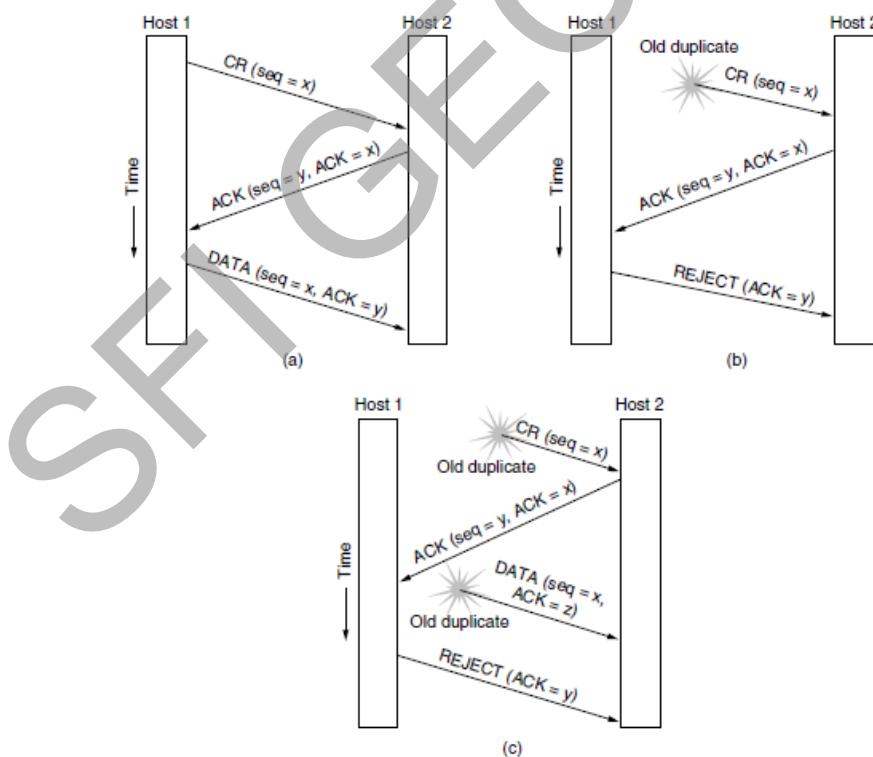


Figure 6-11. Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST. (a) Normal operation. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere. (c) Duplicate CONNECTION REQUEST and duplicate ACK.

In Fig. (b), the first segment is a delayed duplicate CONNECTION REQUEST from an old connection. This segment arrives at host 2 without host 1's knowledge. Host 2 reacts to this segment by sending host 1 an ACK segment, in effect asking for verification that host 1 was indeed trying to set up a new connection. When host 1 rejects host 2's attempt to establish a connection, host 2 realizes that it was tricked by a delayed duplicate and abandons the connection. In this way, a delayed duplicate does no damage.

The worst case is when both a delayed CONNECTION REQUEST and an ACK are floating around in the subnet. This case is shown in Fig. As in the previous example, host 2 gets a delayed CONNECTION REQUEST and replies to it. At this point, it is crucial to realize that host 2 has proposed using y as the initial sequence number for host 2 to host 1 traffic, knowing full well that no segments containing sequence number y or acknowledgements to y are still in existence. When the second delayed segment arrives at host 2, the fact that z has been acknowledged rather than y tells host 2 that this, too, is an old duplicate.

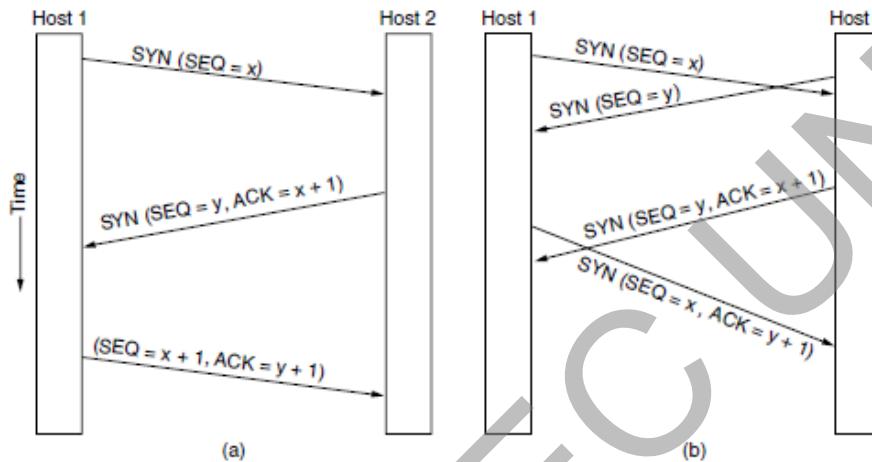


Figure 6-37. (a) TCP connection establishment in the normal case. (b) Simultaneous connection establishment on both sides.

Connections are established in TCP by means of the three-way handshake. To establish a connection, one side, say, the server, passively waits for an incoming connection by executing the LISTEN and ACCEPT primitives in that order, either specifying a specific source or nobody in particular. The other side, say, the client, executes a CONNECT primitive, specifying the IP address and port to which it wants to connect, the maximum TCP segment size it is willing to accept, and optionally some user data (e.g., a password). The CONNECT Primitive sends a TCP segment with the SYN bit on and ACK bit off and waits for a response. When this segment arrives at the destination, the TCP entity there checks to see if there is a process that has done a LISTEN on the port. If not, it sends a reply with the RST bit on to reject the connection. If some process is listening to the port, that process is given the incoming TCP segment. It can either accept or reject the connection. If it accepts, an acknowledgement segment is sent back. The sequence of TCP segments sent in the normal case is shown in Fig. 6-37(a). SYN segment consumes 1 byte of sequence space so that it can be acknowledged unambiguously. In the event that two hosts simultaneously attempt to establish a connection between the same two sockets, the sequence of events is as illustrated in Fig. The result of these events is that just one connection is established, not two, because connections are identified by their end points. If the first setup results in a connection identified by (x, y) and the second one does too, only one table entry is made, namely, for (x, y) .

19. Explain the process of address resolution by using ARP and RARP protocol.

Designers of TCP/IP protocols found a creative solution to the address resolution problem for networks like the Ethernet that have broadcast capability. The solution allows new hosts or routers to be added to the network without recompiling code, and does not require maintenance of a centralized database. To avoid maintaining a table of mappings, the designers chose to use a low-level protocol to bind addresses dynamically. Termed the **Address Resolution Protocol (ARP)**, the protocol provides a mechanism that is both reasonably efficient and easy to maintain. The Address Resolution Protocol, ARP, allows a host to **find** the physical address of a target host on the same physical network, given only the target's IP address.

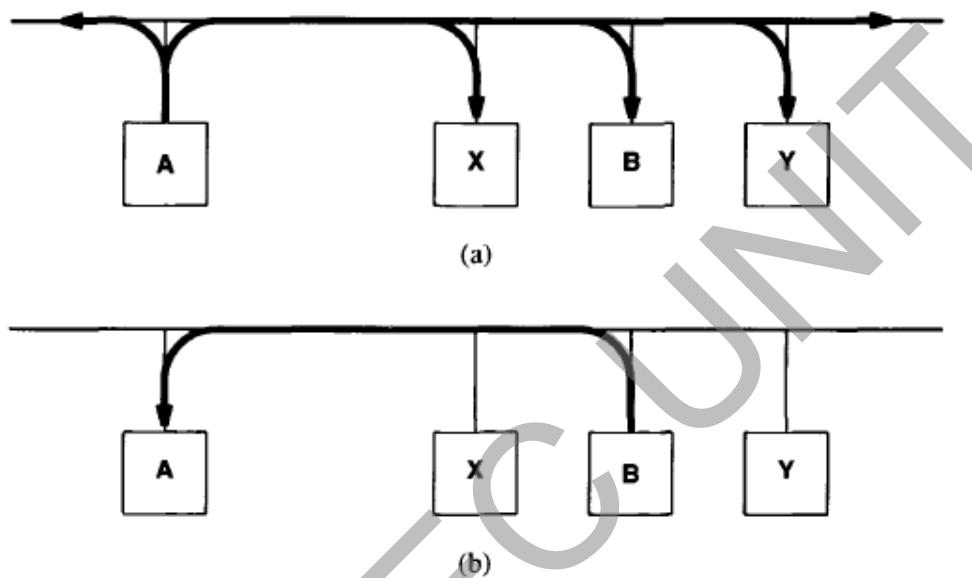


Figure 5.1 The ARP protocol. To determine P_B , B 's physical address, from I_B , its IP address, (a) host A broadcasts an ARP request containing I_B to all machines on the net, and (b) host B responds with an ARP reply that contains the pair (I_B, P_B) .

As Figure shows, the idea behind dynamic resolution with **ARP** is simple: when host **A** wants to resolve **IP** address **ZB**, it broadcasts a special packet that asks the host with IP address **Ie** to respond with its physical address, **PB**. All hosts, including **B**, receive the request, but only host **B** recognizes its **IP** address and sends a reply that contains its physical address. When **A** receives the reply, it uses the physical address to send the internet packet directly to **B**.

To reduce communication costs, computers that use ARP maintain a cache of recently acquired IP-to-physical address bindings. That is, whenever a computer sends an **ARP** request and receives an **ARP** reply, it saves the **IP** address and corresponding hardware address information in its cache for successive lookups. When transmitting a packet, a computer always looks in its cache for a binding before sending an AFW request. If it finds the desired binding in its **ARP** cache, the computer need not broadcast on the network. Thus, when two computers on a network communicate, they begin with an **ARP** request and response, and then repeatedly transfer packets without using **ARP** for each one.

ARP is divided into two parts. The first part maps an IP address to a physical address when sending a packet, and the second part answers requests from other machines. Address resolution for outgoing packets seems straightforward, but small details complicate an implementation. Given a destination IP address the software consults its ARP cache to see if it knows the mapping from IP address to physical address. If it does, the software extracts the physical address, places the data in a frame using that address, and sends the frame. If it does not know the mapping, the software must broadcast an **ARP** request and wait for a reply.

Broadcasting an ARP request to find an address mapping can become complex. The target machine can be down or just too busy to accept the request. If so, the sender may not receive a reply or the reply may be delayed. Because the Ethernet is a best-effort delivery system, the initial ARP broadcast request can also be. Meanwhile, the host must store the original outgoing packet so it can be sent once the address has been resolved. In fact, the host must decide whether to allow other application programs to proceed while it processes an AFW request (most do). If so, the software must handle the case where an application generates additional **ARP** requests for the same address without broadcasting multiple requests for a given target. When **ARP** messages travel from one machine to another, they must be carried in physical frames. To identify the frame as carrying an ARP message, the sender assigns a special value to the type field in the frame header, and places the ARP message in the frame's data field. When a frame arrives at a computer, the network software uses the frame type to determine its contents.

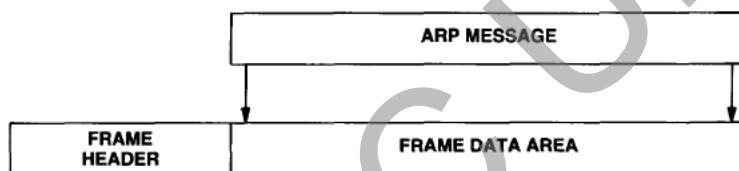


Figure 5.2 An ARP message encapsulated in a physical network frame.

ARP Protocol Format

ARP packets does not have a fixed-format header. Instead, to make ARP useful for a variety of network technologies, the length of fields that contain addresses depend on the type of network. Field **HARDWARE TYPE** specifies a hardware interface type for which the sender seeks an answer; it contains the value **1** for Ethernet. Similarly, field **PROTOCOL TYPE** specifies the type of high-level protocol address the sender has supplied; it contains **0800**, for IP addresses. Field **OPERATION** specifies an **ARP** request (1), **ARP** response (2), **RARP** request (3), or **RARP** response (4). Fields **HLEN** and **PLEN** allow **ARP** to be used with arbitrary networks because they specify the length of the hardware address and the length of the high-level protocol address. The sender supplies its hardware address and **IP'** address, if known, in fields **SENDER HA** and **SENDER IP**. When making a request, the sender also supplies the target hardware address (**RARP**) or target IP address (**ARP**), using fields **TARGET HA** or **TARGET IP**. Before the target machine responds, it fills in the missing addresses, swaps the target and sender pairs, and changes the operation to a reply. Thus, a reply carries the **IP** and hardware addresses of the original requester, as well as the **IP** and hardware addresses of the machine for which a binding was sought.

HARDWARE TYPE		PROTOCOL TYPE				
HLEN	PLEN	OPERATION				
SENDER HA (octets 0-3)						
SENDER HA (octets 4-5)		SENDER IP (octets 0-1)				
SENDER IP (octets 2-3)		TARGET HA (octets 0-1)				
TARGET HA (octets 2-5)						
TARGET IP (octets 0-3)						

Figure 5.3 An example of the ARP/RARP message format when used for IP-to-Ethernet address resolution. The length of fields depends on the hardware and protocol address lengths, which are 6 octets for an Ethernet address and 4 octets for an IP address.

TCP/IP protocol that allows a computer to obtain its IP address from a server is known as the **Reverse Address Resolution Protocol (RARP)**. Like an ARP message, a RARP message is sent from one machine to another encapsulated in the data portion of a network frame. For example, an Ethernet frame carrying a RARP request has the usual preamble, Ethernet source and destination addresses, and packet **type** fields in front of the frame. The frame **type** contains the value 8035,, to identify the contents of the frame as a RARP message. The data portion of the frame contains the 28-octet RARP message.

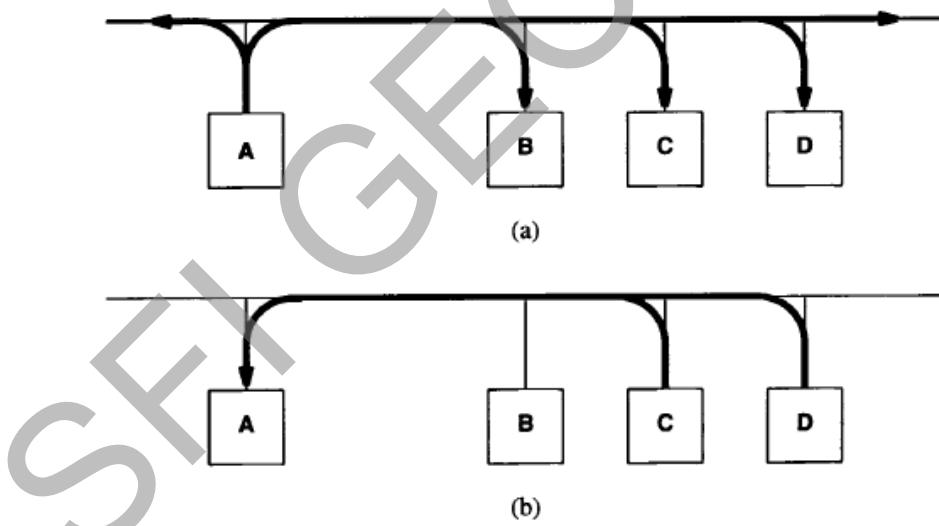


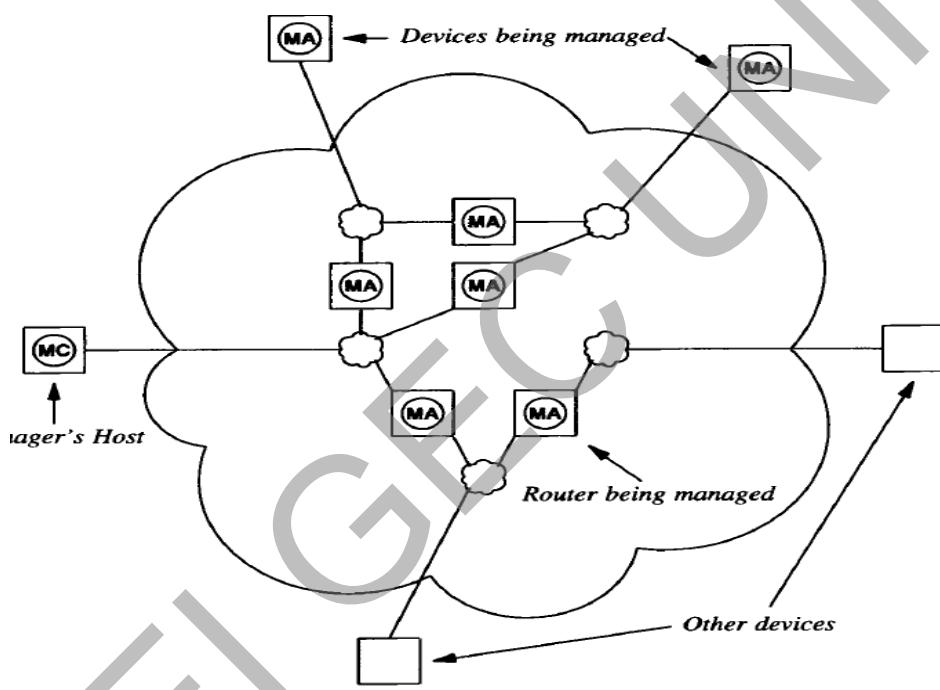
Figure 6.1 Example exchange using the RARP protocol. (a) Machine A broadcasts a RARP request specifying itself as a target, and (b) those machines authorized to supply the RARP service (C and D) reply directly to A.

Figure illustrates how a host uses RARP. The sender broadcasts a RARP request that specifies itself as both the sender and target machine, and supplies its physical network address in the target hardware address field. All computers on the network receive the request, but only those authorized to supply the RARP service process the request and send a reply; such computers are known informally as **RARP servers**. For RARP to succeed, the network must contain at least one RARP server. Servers answer requests by filling in the target protocol address field,

changing the message **type** from **request** to **reply**, and sending the reply back directly to the machine making the request. The original machine receives replies from all RARP servers, even though only the first is needed.

20. Explain how network is managed by using SNMP protocol.

Although the designers provided a range of values to be used as the delay between successive router advertisements, they chose the default of 10 minutes. The value was selected as a compromise between rapid failure detection and low overhead. A smaller value would allow more rapid detection of router failure, but would increase network traffic; a larger value would decrease traffic, but would delay failure detection. One of the issues the designers considered was how to accommodate a large number of routers on the same network. From the point of view of a host, the default delay has a severe disadvantage: a host cannot afford to wait many minutes for an advertisement when it first boots. To avoid such delays, the designers included an **ICMP router solicitation message** that allows host to request an immediate advertisement



- Client software usually runs on the manager's workstation.
- Each participating router or host runs a server program.
- Server software is called a management agent or merely an agent.
- A manager invokes client software on the local host computer and specifies an agent with which it communicates.
- After the client contacts the agent, it sends queries to obtain information or it sends commands to change conditions in the router.
- Internet management software uses an authentication mechanism to ensure only authorized managers can access or control a particular device.

- TCP/IP network management protocols divide the management problem into two parts and specify separate standards for each part.
- The first part concerns communication of information.
 - A protocol specifies how client software running on a manager's host **communicates with an agent**.
 - The protocol defines the **format and meaning** of messages clients and servers exchange as well as the form of names and addresses.
- The second part concerns the data being managed.
 - A protocol specifies which **data items** a managed device must keep as well as the **name of each data item and the syntax** used to express the name.
- Management Information Base (MIB), the standard specifies the data items a managed device must keep, the operations allowed on each, and the meanings.
 - **Router** keeps statistics on the status of its network interfaces, incoming and outgoing packet traffic, dropped datagrams, and error messages generated.
 - **Modem** keeps statistics about the number of characters sent and received, baud rate, and calls accepted.

MIB category	Includes Information About
system	The host or router operating system
interfaces	Individual network interfaces
at	Address translation (e.g., ARP mappings)
ip	Internet Protocol software
icmp	Internet Control Message Protocol software
tcp	Transmission Control Protocol software
udp	User Datagram Protocol software
ospf	Open Shortest Path First software
bgp	Border Gateway Protocol software
rmon	Remote network monitoring
rip-2	Routing Information Protocol software
dns	Domain Name System software

Figure 30.2 Example categories of MIB information. The category is encoded in the identifier used to specify an object.

MIB Variable	Category	Meaning
sysUpTime	system	Time since last reboot
ifNumber	interfaces	Number of network interfaces
ifMtu	interfaces	MTU for a particular interface
ipDefaultTTL	ip	Value IP uses in time-to-live field
ipInReceives	ip	Number of datagrams received
ipForwDatagrams	ip	Number of datagrams forwarded
ipOutNoRoutes	ip	Number of routing failures
ipReasmOKs	ip	Number of datagrams reassembled
ipFragOKs	ip	Number of datagrams fragmented
ipRoutingTable	ip	IP Routing table
icmplnEchos	icmp	Number of ICMP Echo Requests received
tcpRtoMin	tcp	Minimum retransmission time TCP allows
tcpMaxConn	tcp	Maximum TCP connections allowed
tcpInSegs	tcp	Number of segments TCP has received
udpInDatagrams	udp	Number of UDP datagrams received

Figure 30.3 Examples of MIB variables along with their categories.

- SMI standard specifies a set of rules used to define and identify MIB variables.
- To keep network management protocols simple, the SMI places restrictions on the **types of variables allowed** in the MIB, specifies the **rules for naming** those variables, and creates **rules for defining variable types**.
- The SMI standard specifies that all MIB variables must be defined and referenced using ISO's ASN.1.
- ASN.1 is a formal language that has two main features:
 - a notation used in documents that humans read and
 - a compact encoded representation of the same information used in communication protocols.
- In both cases, the precise, formal notation **removes any possible ambiguities** from both the representation and meaning.
- ASN.1 also helps **simplify** the implementation of network management protocols and **guarantees interoperability**.
- It defines precisely how to encode both names and data items in a message.
- Names used for MIB variables are taken from the object identifier namespace administered by ISO and ITU.
- The object identifier namespace is absolute (global), meaning that names are structured to make them globally unique.
- The object identifier namespace is hierarchical.
- The root of the object identifier hierarchy is unnamed, but has three direct descendants managed by: ISO, ITU, and jointly by ISO and ITU.
- The descendants are assigned both short text strings and integers that identify them.

- ISO has allocated one subtree for use by other national or international standards organizations (including U.S. standards organizations)
- The U.S. National Institute for Standards and Technology has allocated a subtree for the U.S. **Department of Defense**.
- the names of all MIB variables corresponding to IP have an identifier that begins with the prefix 1.3.6.1.2.1.4
- Textual label representation instead of the numeric representation can be:

iso . org . dod. internet. mgmt . mib . Ip

- Suffix 0 refers to the instance of the variable with that name. So, when it appears in a message sent to a router, the numeric representation is: 1.3.6.1.2.1.4.3.0
which refers to the instance of on that router.
- Network management protocols specify communication between the network management client program a manager invokes and a network management server program executing on a host or router.
- In addition to defining the form and meaning of messages exchanged and the representation of names and values in those messages, network management protocols also define administrative relationships among routers being managed.
- They provide authentication of managers.

Command	Meaning
get-request	Fetch a value from a specific variable
get-next-request	Fetch a value without knowing its exact name
get-bulk-request	Fetch a large volume of data (e.g., a table)
response	A response to any of the above requests
set-request	Store a value in a specific variable
inform-request	Reference to third-part data (e.g., for a proxy)
snmpv2-trap	Reply triggered by an event
report	Undefined at present

Figure 30.6 The set of possible SNMP operations. *Get-next-request* allows the manager to iterate through a table of items.

- Operations get-request and set-request provide the basic fetch and store operations response provides the reply.
- SNMP specifies that operations must be atomic, meaning that if a single SNMP message specifies operations on multiple variables, the server either performs all operations or none of them.
- The get-next-request operation allows a client to iterate through a table without knowing how many items the table contains.
- When sending a get-next-request, the client supplies a prefix of a valid object identifier, P.

- The agent examines the set of object identifiers for all variables it controls, and sends a response for the variable that occurs next in lexicographic order.
- The agent must know the ASN.1 names of all variables and be able to select the first variable with object identifier greater than P.
- SNMP messages do not have fixed fields.
- They use the standard ASN.1 encoding.
- Message can be difficult for humans to decode and understand.
- Each item in the grammar consists of a descriptive name followed by a declaration of the item's type.
- For example, an item such as

```
msgversion INTEGER (0..2147483647)
```

declares the name msgversion to be a nonnegative integer less than or equal to 2147483647.

```
SNMPv3Message ::=  
SEQUENCE {  
    msgVersion INTEGER (0..2147483647),  
    -- note: version number 3 is used for SNMPv3  
    msgGlobalData HeaderData,  
    msgSecurityParameters OCTET STRING,  
    msgData ScopedPduData  
}
```

Figure 30.7 The SNMP message format in ASN.1-style notation. Text following two consecutive dashes is a comment.

21. Explain the architecture of World Wide Web.

The Web, as the World Wide Web is popularly known, is an architectural framework for accessing linked content spread out over millions of machines all over the Internet. From the users' point of view, the Web consists of a vast, worldwide collection of content in the form of **Web pages**, often just called **pages** for short. Each page may contain links to other pages anywhere in the world. Users can follow a link by clicking on it, which then takes them to the page pointed to. This process can be repeated indefinitely. The idea of having one page point to another, now called **hypertext**. Pages are generally viewed with a program called a **browser**. A piece of text, icon, image, and so on associated with another page is called a **hyperlink**. To follow a link, the user places the mouse cursor on the linked portion of the page area (which causes the cursor to change shape) and clicks. Following a link is simply a way of telling the browser to fetch another page. In the early days of the Web, links were highlighted with underlining and colored text so that they would stand out. Nowadays, the creators of Web pages have ways to control the look of linked regions, so a link might appear as an icon or change its appearance when the mouse passes over it. It is up to the creators of the page to make the links

visually distinct, to provide a usable interface. The basic model behind the display of pages is also shown in Fig. The browser is displaying a Web page on the client machine. Each page is fetched by sending a request to one or more servers, which respond with the contents of the page. The request-response protocol for fetching pages is a simple text-based protocol that runs over TCP, just as was the case for SMTP. It is called **HTTP (HyperText Transfer Protocol)**. The content may simply be a document that is read off a disk, or the result of a database query and program execution. The page is a **static page** if it is a document that is the same every time it is displayed. In contrast, if it was generated on demand by a program or contains a program it is a **dynamic page**. A dynamic page may present itself differently each time it is displayed.

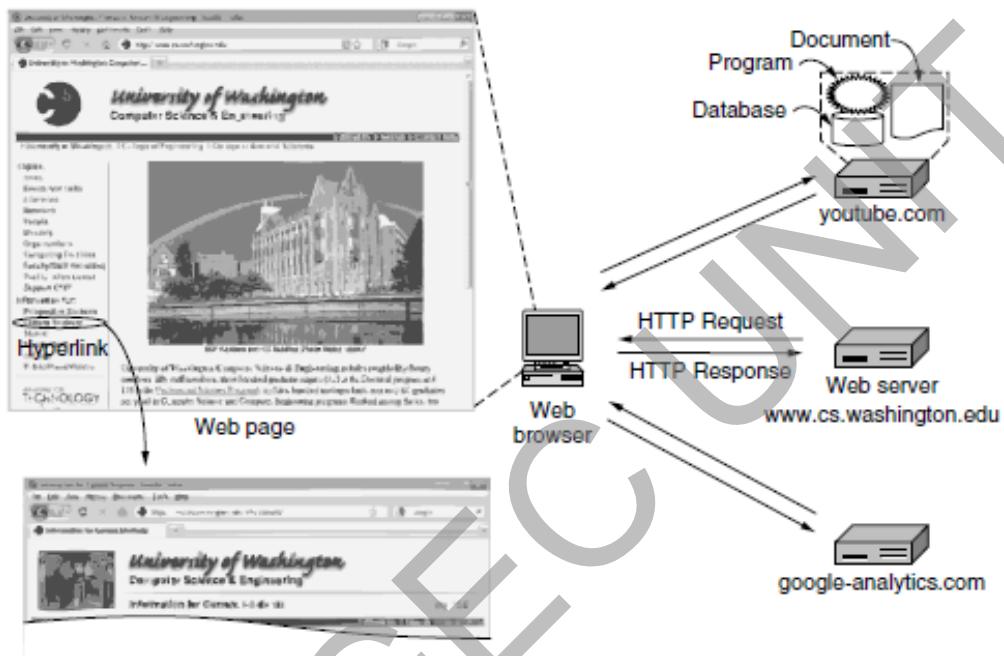


Figure 7-18. Architecture of the Web.

The Client Side

When the Web was first created, it was immediately apparent that having one page point to another Web page required mechanisms for naming and locating pages. In particular, three questions had to be answered before a selected page could be displayed:

1. What is the page called?
2. Where is the page located?
3. How can the page be accessed?

If every page were somehow assigned a unique name, there would not be any ambiguity in identifying pages. The solution chosen identifies pages in a way that solves all three problems at once. Each page is assigned a **URL (Uniform Resource Locator)** that effectively serves as the page's worldwide name. URLs have three parts: the protocol (also known as the **scheme**), the DNS name of the machine on which the page is located, and the path uniquely indicating the specific page (a file to read or program to run on the machine). In the general case, the path has a hierarchical name that models a file directory structure. This URL consists of three parts: the protocol (http), the DNS name of the host (www.cs.washington.edu), and the path name (index.html).

When a user clicks on a hyperlink, the browser carries out a series of steps in order to fetch the page pointed to. Let us trace the steps that occur when our example link is selected:

1. The browser determines the URL (by seeing what was selected).
2. The browser asks DNS for the IP address of the server www.cs.washington.edu.
3. DNS replies with 128.208.3.88.
4. The browser makes a TCP connection to 128.208.3.88 on port 80, the well-known port for the HTTP protocol.
5. It sends over an HTTP request asking for the page /index.html.
6. The www.cs.washington.edu server sends the page as an HTTP response, for example, by sending the file /index.html.
7. If the page includes URLs that are needed for display, the browser fetches the other URLs using the same process. In this case, the URLs include multiple embedded images also fetched from www.cs.washington.edu, an embedded video from youtube.com, and a script from google-analytics.com.
8. The browser displays the page /index.html as it appears in Fig.
9. The TCP connections are released if there are no other requests to the same servers for a short period. The URL design is open-ended in the sense that it is straightforward to have browsers use multiple protocols to get at different kinds of resources. The http protocol is the Web's native language, the one spoken by Web servers. **HTTP** stands for **HyperText Transfer Protocol**. We will examine it in more detail later in this section. The ftp protocol is used to access files by FTP, the Internet's file transfer protocol. It is possible to access a local file as a Web page by using the file protocol, or more simply, by just naming it. This approach does not require having a server. Of course, it works only for local files, not remote ones. The mailto protocol does not really have the flavor of fetching Web pages, but is useful anyway. It allows users to send email from a Web browser. Most browsers will respond when a mailto link is followed by starting the user's mail agent to compose a message with the address field already filled in. The rtsp and sip protocols are for establishing streaming media sessions and audio and video calls. Finally, the about protocol is a convention that provides information about the browser. URLs have been generalized into **URIs (Uniform Resource Identifiers)**. Some URIs tell how to locate a resource. These are the URLs. Other URIs tell the name of a resource but not where to find it. These URIs are called **URNs (Uniform Resource Names)**. The rules for writing URIs are given in RFC 3986, while the different URI schemes in use are tracked by IANA.

MIME Types

To allow all browsers to understand all Web pages, Web pages are written in a standardized language called **HTML**. Although a browser is basically an HTML interpreter, most browsers have numerous buttons and features to make it easier to navigate the Web. For a rapidly growing collection of file types, most browsers have chosen a more general solution. When a server returns a page, it also returns some additional information about the page. This information includes the MIME type of the page. Pages of type text/html are just displayed directly, as are pages in a few other built-in types. If the MIME type is not one of the built-in ones, the browser consults its table of MIME types to determine how to display the page.

This table associates MIME types with viewers. There are two possibilities: plug-ins and helper applications. A **plug-in** is a third-party code module that is installed as an extension to the browser, as illustrated in Fig. Because plug-ins run inside the browser, they have access to the current page and can modify its appearance. Each browser has a set of procedures that all plug-ins must implement so the browser can call the plug-ins. For example, there is typically a procedure the browser's base code calls to supply the plug-in with data to display. This set of

procedures is the plug-in's interface and is browser specific. In addition, the browser makes a set of its own procedures available to the plug-in, to provide services to plug-ins. Typical procedures in the browser interface are for allocating and freeing memory, displaying a message on the browser's status line, and querying the browser about parameters.

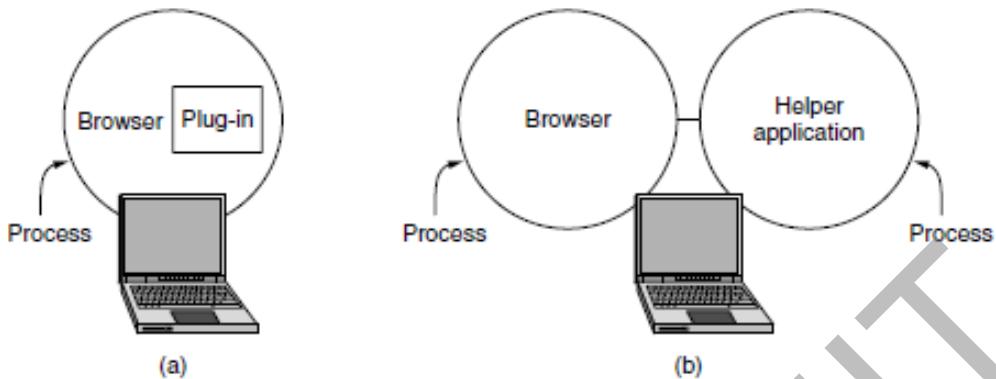


Figure 7-20. (a) A browser plug-in. (b) A helper application.

Before a plug-in can be used, it must be installed. The usual installation procedure is for the user to go to the plug-in's Web site and download an installation file. Executing the installation file unpacks the plug-in and makes the appropriate calls to register the plug-in's MIME type with the browser and associate the plug-in with it. Browsers usually come preloaded with popular plug-ins. The other way to extend a browser is make use of a **helper application**. This is a complete program, running as a separate process.

The Server Side

the steps that the server performs in its main loop are:

1. Accept a TCP connection from a client (a browser).
2. Get the path to the page, which is the name of the file requested.
3. Get the file (from disk).
4. Send the contents of the file to the client.
5. Release the TCP connection.

Web servers are implemented with a different design to serve many requests per second. One problem with the simple design is that accessing files is often the bottleneck. Disk reads are very slow compared to program execution, and the same files may be read repeatedly from disk using operating system calls. Another problem is that only one request is processed at a time. The file may be large, and other requests will be blocked while it is transferred. One obvious improvement is to maintain a cache in memory of the n most recently read files or a certain number of gigabytes of content. Before going to disk to get a file, the server checks the cache. If the file is there, it can be served directly from memory, thus eliminating the disk access. Although effective caching requires a large amount of main memory and some extra processing time to check the cache and manage its contents, the savings in time are nearly always worth the overhead and expense. To tackle the problem of serving a single request at a time, one strategy is to make the server **multithreaded**. In one design, the server consists of a front-end module that accepts all incoming requests and k processing modules, as shown in Fig. The threads all belong to the same process, so the processing modules all have access to the cache

within the process' address space. When a request comes in, the front end accepts it and builds a short record describing it. It then hands the record to one of the processing modules.

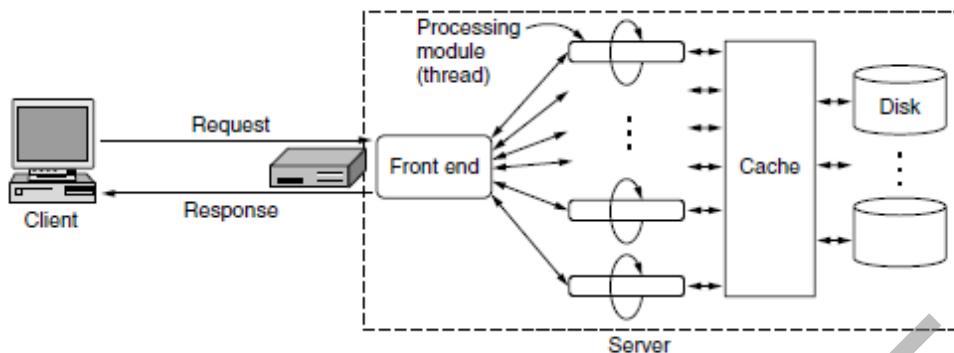


Figure 7-21. A multithreaded Web server with a front end and processing modules.

The processing module first checks the cache to see if the file needed is there. If so, it updates the record to include a pointer to the file in the record. If it is not there, the processing module starts a disk operation to read it into the cache (possibly discarding some other cached file(s) to make room for it). When the file comes in from the disk, it is put in the cache and also sent back to the client. The advantage of this scheme is that while one or more processing modules are blocked waiting for a disk or network operation to complete (and thus consuming no CPU time), other modules can be actively working on other requests. With k processing modules, the throughput can be as much as k times higher than with a single-threaded server. In fact, the actual processing of each request can get quite complicated. For this reason, in many servers each processing module performs a series of steps. The front end passes each incoming request to the first available module, which then carries it out using some subset of the following steps, depending on which ones are needed for that particular request. These steps occur after the TCP connection and any secure transport mechanism have been established.

1. Resolve the name of the Web page requested.
2. Perform access control on the Web page.
3. Check the cache.
4. Fetch the requested page from disk or run a program to build it.
5. Determine the rest of the response (e.g., the MIME type to send).
6. Return the response to the client.
7. Make an entry in the server log.

Cookies

Navigating the Web as we have described it so far involves a series of independent page fetches. There is no concept of a login session. The browser sends a request to a server and gets back a file. Then the server forgets that it has ever seen that particular client. This model is perfectly adequate for retrieving publicly available documents, and it worked well when the Web was first created. However, it is not suited for returning different pages to different users depending on what they have already done with the server. This behavior is needed for many ongoing interactions with Web sites. The name cookies derives from ancient programmer slang in which a program calls a procedure and gets something back that it may need to present later to get some work done in 1994 and are now specified in RFC 2109.

When a client requests a Web page, the server can supply additional information in the form of a cookie along with the requested page. The cookie is a rather small, named string (of at

most 4 KB) that the server can associate with a browser. This association is not the same thing as a user, but it is much closer and more useful than an IP address. Browsers store the offered cookies for an interval, usually in a cookie directory on the client's disk so that the cookies persist across browser invocations, unless the user has disabled cookies. Cookies are just strings, not executable programs. In principle, a cookie could contain a virus, but since cookies are treated as data, there is no official way for the virus to actually run and do damage. However, it is always possible for some hacker to exploit a browser bug to cause activation. A cookie may contain up to five fields, as shown in Fig. The Domain tells where the cookie came from. Browsers are supposed to check that servers are not lying about their domain. Each domain should store no more than 20 cookies per client. The Path is a path in the server's directory structure that identifies which parts of the server's file tree may use the cookie. The Content field takes the form name = value. Both name and value can be anything the server wants. This field is where the cookie's content is stored. The Expires field specifies when the cookie expires. If this field is absent, the browser discards the cookie when it exits. Such a cookie is called a **nonpersistent cookie**. If a time and date are supplied, the cookie is said to be a **persistent cookie** and is kept until it expires. Expiration times are given in Greenwich Mean Time. To remove a cookie from a client's hard disk, a server just sends it again, but with an expiration time in the past. Finally, the Secure field can be set to indicate that the browser may only return the cookie to a server using a secure transport, namel

Reg. No:

Total Pages :
Name :

6TH SEMESTER B.TECH DEGREE MODEL EXAMINATION MARCH 2018

COURSE CODE : CS 308

COURSE NAME : SOFTWARE ENGINEERING AND PROJECT MANAGEMENT

MAX MARKS : 100

DURATION : 3 HRS

PART A

(Answer ALL questions, each carry 3 marks)

1. Describe process and methods
2. Write in down any two software elicitation methods in detail.
3. What are the necessary characteristics for an SRS.
4. Explain the different project planning phases.

Part B

(Answer Any TWO questions. Each carry 9 marks)

5. Explain water fall model. List its advantages and disadvantages.
6. Write and explain incremental and prototype model. List their significance.
7. Explain CMM

PART C

(Answer ALL questions, each carry 3 marks)

8. What are the objectives of project planning?
9. Explain single variable cost estimation model.
10. Describe Software scope.
11. What are the design principles to adopted in software engineering?

Part D

(Answer Any TWO questions. Each carry 9 marks)

12. Explain COCOMO model in Detail.
13. Describe the modular design concepts Cohesion and Coupling.
14. What are the white box testing techniques.

Part E

(Answer Any FOUR questions. Each carry 10 marks)

15. Write down about Project Management tools.
16. What is CASE? Explain its tools.
17. Describe Software Configuration management.
18. Explain the FOUR Ps in Project management.
19. What are the different steps in risk managements?
20. Explain task set selection in software projects.

**6TH SEMESTER B.TECH DEGREE MODEL EXAMINATION
MARCH 2018**

COURSE CODE : CS 308

COURSE NAME : SOFTWARE ENGINEERING AND PROJECT MANAGEMENT

MAX MARKS : 100

DURATION : 3 HRS

PART A

1. Describe process and methods.

Ans: A *software process* is defined as a framework for the tasks that are required to build high-quality software. The foundation for software engineering is the *process* layer. Software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software. Process defines a framework for a set of *key process areas* (KPAs) that must be established for effective delivery of software engineering technology. The key process areas form the basis for management control of software projects and establish the context in which technical methods are applied, work products (models, documents, data, reports, forms, etc.) are produced, milestones are established, quality is ensured, and change is properly managed.(2.5 marks)

Software engineering *methods* provide the technical how-to's for building software. Methods encompass a broad array of tasks that include requirements analysis, design, program construction, testing, and support. Software engineering methods rely on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.(2.5 marks)

2. Write down any two software elicitation methods in detail.

Ans : Definition(1 mark): Requirements Elicitation is the process to find out the requirements for an intended software system by communicating with client, end users, system users and others who have a stake in the software system development.

1. Interview
2. Surveys
3. Questionnaires
4. Prototyping
5. Observation
6. Brainstorming (Explain any two each carry 2 marks) Refer "Software Requirement Engineering.pdf"

3. What are the necessary characteristics for an SRS.

Ans : Gathering software requirements is the foundation of the entire software development project. Hence they must be clear, correct and well-defined.

A complete Software Requirement Specifications must be:

- Clear
 - Correct
 - Consistent
 - Coherent
 - Comprehensible
 - Modifiable
 - Verifiable
 - Prioritized
 - Unambiguous
 - Traceable
 - Credible source
4. Explain the different project planning phases.

Ans : Project planning takes place at three stages in a project life cycle:

1. At the proposal stage, when we are bidding for a contract to develop or provide a software system. We need a plan at this stage to help us to decide if we have the resources to complete the work and to work out the price that we should quote to a customer.
2. During the project startup phase, when you have to plan who will work on the project, how the project will be broken down into increments, how resources will be allocated across your company, etc.
3. Periodically throughout the project, when we modify our plan in light of experience gained and information from monitoring the progress of the work. We learn more about the system being implemented and capabilities of our development team. This information allows us to make more accurate estimates of how long the work will take. Furthermore, the software requirements are likely to change, means that the work breakdown has to be altered and the schedule extended.

4. Part B

5. Explain water fall model. List its advantages and disadvantages.

Ans : Explain with the help of neat diagram (1 mark)

1. Software requirement analysis
2. Design
3. code generation
4. testing

5. Support

Advantages of waterfall model:

1. This model is simple and easy to understand and use.
2. It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
3. In this model phases are processed and completed one at a time. Phases do not overlap.
4. Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model:

1. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
2. No working software is produced until late during the life cycle.
3. High amounts of risk and uncertainty.
4. Not a good model for complex and object-oriented projects.
5. Poor model for long and ongoing projects.
6. Not suitable for the projects where requirements are at a moderate to high risk of changing.

6. Write and explain incremental and prototype model. List their significance.

Ans :

Explain incremental with neat diagram

Diagram 1 mark

Explain 4 marks

In an Iterative Incremental model, initially, a partial implementation of a total system is constructed so that it will be in a deliverable state. Increased functionality is added. Defects, if any, from the prior delivery are fixed and the working product is delivered. The process is repeated until the entire product development is completed. The repetitions of these processes are called iterations. At the end of every iteration, a product increment is delivered.

Advantages of Incremental model:

Generates working software quickly and early during the software life cycle.

This model is more flexible – less costly to change scope and requirements.

It is easier to test and debug during a smaller iteration.

In this model customer can respond to each built.

Lowers initial delivery cost.

Easier to manage risk because risky pieces are identified and handled during it'd iteration.

Disadvantages of Incremental model:

Needs good planning and design.

Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.

Total cost is higher than **waterfall**.

When to use the Incremental model:

This model can be used when the requirements of the complete system are clearly defined and understood.

Major requirements must be defined; however, some details can evolve with time.

There is a need to get a product to the market early.

A new technology is being used

Resources with needed skill set are not available

There are some high risk features and goals.

Explain **prototype** with neat diagram

Diagram 1 mark

Explain 4 marks

Prototype Model :- Prototype is a working model of software with some limited functionality. The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.

Prototyping is used to allow the users evaluate developer proposals and try them out before implementation. It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.

Following is a stepwise approach explained to design a software prototype.

1. Basic Requirement Identification

This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.

2. Developing the initial Prototype

The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These features may not exactly work in the same manner internally in the actual software developed. While, the workarounds are used to give the same look and feel to the customer in the prototype developed.

3. Review of the Prototype

The prototype developed is then presented to the customer and the other important stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.

4. Revise and Enhance the Prototype

The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like – time and budget constraints and technical feasibility of the actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until the customer expectations are met.

The advantages of the Prototyping Model are as follows –

- Increased user involvement in the product even before its implementation.
- Since a working model of the system is displayed, the users get a better understanding of the system being developed.
- Reduces time and cost as the defects can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily.
- Confusing or difficult functions can be identified.

The Disadvantages of the Prototyping Model are as follows –

- Risk of insufficient requirement analysis owing to too much dependency on the prototype.
- Users may get confused in the prototypes and actual systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Developers may try to reuse the existing prototypes to build the actual system, even when it is not technically feasible.
- The effort invested in building prototypes may be too much if it is not monitored properly.

7. Explain CMM

Ans : Explain need of CMM 1 mark

Explain five levels 7 marks

Explain KPAs 1 marks

The Software Engineering Institute (SEI) Capability Maturity Model (CMM) is an increasing series of levels of a software development organization. The higher the level, the better the software development process, hence reaching each level is an expensive and time-consuming process.

Level One :Initial - The software process is characterized as inconsistent, and occasionally even chaotic. Defined processes and standard practices that exist are abandoned during a crisis. Success of the organization majorly depends on an individual effort, talent, and heroics. The heroes eventually move on to other organizations taking their wealth of knowledge or lessons learnt with them.

•Level Two: Repeatable - This level of Software Development Organization has a basic and consistent project management processes to track cost, schedule, and functionality. The process is in place to repeat the earlier successes on projects with similar applications. Program management is a key characteristic of a level two organization.

•Level Three: Defined - The software process for both management and engineering activities are documented, standardized, and integrated into a standard software process for the entire organization and all projects across the organization use an approved, tailored version of the organization's standard

software process for developing, testing and maintaining the application.

•Level Four: Managed - Management can effectively control the software development effort using precise measurements. At this level, organization set a quantitative quality goal for both software process and software maintenance. At this maturity level, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable.

•Level Five: Optimizing - The Key characteristic of this level is focusing on continually improving process performance through both incremental and innovative technological improvements. At this level, changes to the process are to improve the process performance and at the same time maintaining statistical probability to achieve the established quantitative process-improvement objectives.

Part C

8. What are the objectives of project planning

Ans : Project planning is concerned with identifying the activities.
Milestones and deliverables produced by a project.

- A plan is drawn up to guide the development towards the project goals.
- Cost estimation is a related activity that is concerned with estimating the resources required to accomplish the project plan.

Planning Objectives

- To provide a framework that allows a software manager to make an estimate of resources, cost, and schedule.
- Project outcomes should be bounded by 'best case' and 'worst case' scenarios.
- Estimates should be updated as the project progresses.

9. Single variable model

- used an equation to estimate the desired values such as cost, time , effort etc
- depend on the same variable used as predictor

$$C = aL^b$$

where c is cost, L size in LOC, a and b depend on local development environment.
SEL model.

$$\text{Effort} = 1.4 L^{0.93}$$

$$\text{Documentation} = 30.4 L^{0.9}$$

$$\text{Duration} = 4.6 L^{0.26}$$

10. Scope Management

- want to establish a project scope that is unambiguous and understandable at management and technical levels
- It defines the scope of project; this includes all the activities, process need to be done in order to make a deliverable software product.
- creates boundaries of the project by clearly defining what would be done

in the project and what would not be done.

- makes project to contain limited and quantifiable tasks, which can easily be documented and in turn avoids cost and time overrun.

Software Scope

Scope is defined by answering the following questions:

- **Context** :- How does the software to be built fit into a larger system, product, or business context and what constraints are imposed as a result of the context?
- **Information objectives** :-What customer-visible data objects are produced as output from the software? What data objects are required for input?
- **Function and performance.** What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed?

During Project Scope management, it is necessary to :-

- Define the scope
- Decide its verification and control
- Divide the project into various smaller parts for ease of management.
- Verify the scope
- Control the scope by incorporating changes to the scope

Scoping Techniques:

- FAST (Facilitated Application Specification Technique)
- QFD (Quality Function Deployment)
- Use-Cases

Scope is affected by:

- Customers' needs
- Business context
- Project boundaries
- Customers' motivation
- Likely paths for change

11. A set1 of principles for software design, which have been adapted and extended in the following list:

- **The design process should not suffer from “tunnel vision.”** A good designer should consider alternative approaches, judging each based on the requirements of the problem, the resources available to do the job.
- **The design should be traceable to the analysis model.** Because a single element of the design model often traces to multiple requirements, it is necessary to have a means for tracking how requirements have been satisfied by the design model.
- **The design should not reinvent the wheel.** Systems are constructed using a set of design patterns, many of which have likely been encountered before. These patterns should always be chosen as an alternative to reinvention. Time is short and resources

are limited! Design time should be invested in representing truly new ideas and integrating those patterns that already exist.

- **The design should “minimize the intellectual distance”**

between the software and the problem as it exists in the real world. That is, the structure of the software design should (whenever possible) mimic the structure of the problem domain.

- **The design should exhibit uniformity and integration.** A design is uniform if it appears that one person developed the entire thing. Rules of style and format should be defined for a design team before design work begins. A design is integrated if care is taken in defining interfaces between design components.

- **The design should be structured to accommodate change.** The design concepts discussed in the next section enable a design to achieve this principle.

- **The design should be structured to degrade gently, even when aberrant data, events, or operating conditions are encountered.** Welldesigned software should never “bomb.” It should be designed to accommodate unusual circumstances, and if it must terminate processing, do so in a graceful manner.

- **Design is not coding, coding is not design.** Even when detailed procedural designs are created for program components, the level of abstraction of the design model is higher than source code. The only design decisions made at the coding level address the small implementation details that enable the procedural design to be coded.

- **The design should be assessed for quality as it is being created, not after the fact.** A variety of design concepts and design measuresare available to assist the designer in assessing quality.

- **The design should be reviewed to minimize conceptual (semantic) errors.** There is sometimes a tendency to focus on minutiae when the design is reviewed, missing the forest for the trees. A design team should ensure that major conceptual elements of the design (omissions, ambiguity, inconsistency) have been addressed before worrying about the syntax of the design model.

Part D

11. COCOMO Model

- COCOMO (COnstructive COst MOdel) proposed by Boehm.

- Divides software product developments into 3 categories:

- Organic
- Semidetached
- Embedded

COCOMO Product classes

- Roughly correspond to:

- Application, utility and system programs respectively.
 - Data processing and scientific programs are considered to be application programs.
 - Compilers, linkers, editors, etc., are utility programs.
 - Operating systems and real-time system programs, etc. are system

programs.

Elaboration of Product classes

- Organic:
 - Relatively small groups
 - Working to develop well-understood applications.
- Semidetached:
 - Project team consists of a mixture of experienced and inexperienced staff.
- Embedded:
 - The software is strongly coupled to complex hardware, or real-time systems.

COCOMO Model

- For each of the three product categories:
 - From size estimation (in KLOC), Boehm provides equations to predict:
 - project duration in months
 - effort in programmer-months
- Boehm obtained these equations:
 - Examined historical data collected from a large number of actual projects.
- Software cost estimation is done through three stages:
 - Basic COCOMO,
 - Intermediate COCOMO,
 - Complete COCOMO.

Basic COCOMO Model

- Gives only an approximate estimation:
 - Effort = $a_1(KLOC)a_2$
 - $D=b_1\text{Effort}b_2$
 - KLOC is the estimated kilolines of source code,
 - $a_1a_2b_1b_2$ are constants for different categories of software products,
 - D is the estimated time to develop the software in months,
 - Effort estimation is obtained in terms of person months (PMs).

Development Effort Estimation

- Organic :
 - Effort = 2.4 (KLOC)1.05 PM
- Semi-detached:
 - Effort = 3.0(KLOC)1.12 PM
- Embedded:
 - Effort = 3.6 (KLOC)1.20PM

Development Time Estimation

- Organic:
 - $D = 2.5 (\text{Effort})0.38 \text{ Months}$
- Semi-detached:
 - $D = 2.5 (\text{Effort})0.35 \text{ Months}$
- Embedded:

- $D = 2.5 \text{ (Effort)} \times 0.32 \text{ Months}$

Intermediate COCOMO

- Basic COCOMO model assumes
 - Effort and development time depend on product size alone.
- However, several parameters affect effort and development time:
 - Reliability requirements
 - Availability of CASE tools and modern facilities to the developers
 - Size of data to be handled
- For accurate estimation,
 - the effect of all relevant parameters must be considered:
 - Intermediate COCOMO model recognizes this fact:
 - Refines the initial estimate obtained by the basic COCOMO by using a set of 15 cost drivers(multipliers).
- If modern programming practices are used,
 - Initial estimates are scaled downwards.
- If there are stringent reliability requirements on the product :
 - Initial estimate is scaled upwards.
- Rate different parameters on a scale of one to three:
 - Depending on these ratings,
 - Multiply cost driver values with the estimate obtained using the basic COCOMO.
- Cost driver classes:
 - Inherent complexity of the product, reliability requirements of the product, etc.
 - Execution time, storage requirements, etc.
 - Experience of personnel, etc.
 - Development Environment: Sophistication of the tools used for software development.

Shortcoming of basic and intermediate COCOMO models

- Both models:
 - consider a software product as a single homogeneous entity:
 - However, most large systems are made up of several smaller sub-systems.
 - Some sub-systems may be considered as organic type, some may be considered embedded, etc.
 - For some the reliability requirements may be high, and so on.

12. Coupling and Cohesion

When a software program is modularized, its tasks are divided into several modules based on some characteristics. As we know, modules are set of instructions put together in order to achieve some tasks. They are though, considered as single entity but may refer to each other to work together. There are measures by which the quality of a design of modules and their interaction among them can be measured. These measures are called coupling and cohesion.

Cohesion

Cohesion is a measure that defines the degree of intra-dependability within elements of a module. The greater the cohesion, the better is the program design.

There are seven types of cohesion, namely –

- **Co-incidental cohesion** - It is unplanned and random cohesion, which might be the result of breaking the program into smaller modules for the sake of modularization. Because it is unplanned, it may serve confusion to the programmers and is generally not-accepted.
- **Logical cohesion** - When logically categorized elements are put together into a module, it is called logical cohesion.
- **Temporal Cohesion** - When elements of module are organized such that they are processed at a similar point in time, it is called temporal cohesion.
- **Procedural cohesion** - When elements of module are grouped together, which are executed sequentially in order to perform a task, it is called procedural cohesion.
- **Communicational cohesion** - When elements of module are grouped together, which are executed sequentially and work on same data (information), it is called communicational cohesion.
- **Sequential cohesion** - When elements of module are grouped because the output of one element serves as input to another and so on, it is called sequential cohesion.
- **Functional cohesion** - It is considered to be the highest degree of cohesion, and it is highly expected. Elements of module in functional cohesion are grouped because they all contribute to a single well-defined function. It can also be reused.

Coupling

Coupling is a measure that defines the level of inter-dependability among modules of a program. It tells at what level the modules interfere and interact with each other. The lower the coupling, the better the program.

There are five levels of coupling, namely -

- **Content coupling** - When a module can directly access or modify or refer to the content of another module, it is called content level coupling.
- **Common coupling** - When multiple modules have read and write access to some global data, it is called common or global coupling.
- **Control coupling** - Two modules are called control-coupled if one of them decides the function of the other module or changes its flow of execution.
- **Stamp coupling** - When multiple modules share common data structure and work on different part of it, it is called stamp coupling.
- **Data coupling** - Data coupling is when two modules interact with each other by means of passing data (as parameter). If a module passes data structure as parameter, then the receiving module should use all its components.

Ideally, no coupling is considered to be the best.

•13. White Box Testing Techniques:

•**Statement Coverage** - This technique is aimed at exercising all programming statements with minimal tests.

•**Branch Coverage** - This technique is running a series of tests to ensure that all branches are tested at least once.

•**Path Coverage** - This technique corresponds to testing all possible paths which means that each statement and branch is covered.

Calculating Structural Testing Effectiveness:

•Statement Testing = (Number of Statements Exercised / Total Number of Statements) x 100 %

•

•Branch Testing = (Number of decisions outcomes tested / Total Number of decision Outcomes) x 100 %

•

•Path Coverage = (Number paths exercised / Total Number of paths in the program) x 100 %

•Advantages of White Box Testing:

•Forces test developer to reason carefully about implementation.

•Reveals errors in "hidden" code.

•Spots the Dead Code or other issues with respect to best programming practices.

Disadvantages of White Box Testing:

•Expensive as one has to spend both time and money to perform white box testing.

•Every possibility that few lines of code are missed accidentally.

•In-depth knowledge about the programming language is necessary to perform white box testing.

Part E

15. Project Management Tools

The risk and uncertainty rises multifold with respect to the size of the project, even when the project is developed according to set methodologies.

There are tools available, which aid for effective project management. A few are described -

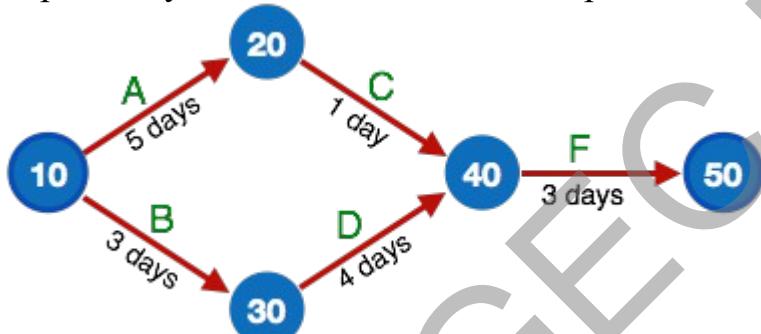
Gantt Chart

Gantt charts was devised by Henry Gantt (1917). It represents project schedule with respect to time periods. It is a horizontal bar chart with bars representing activities and time scheduled for the project activities.

Weeks	1	2	3	4	5	6	7	8	9	10
Project Activities										
Planning										
Design										
Coding										
Testing										
Delivery										

PERT Chart

PERT (Program Evaluation & Review Technique) chart is a tool that depicts project as network diagram. It is capable of graphically representing main events of project in both parallel and consecutive way. Events, which occur one after another, show dependency of the later event over the previous one.



Events are shown as numbered nodes. They are connected by labeled arrows depicting sequence of tasks in the project.

Resource Histogram

This is a graphical tool that contains bar or chart representing number of resources (usually skilled staff) required over time for a project event (or phase). Resource Histogram is an effective tool for staff planning and coordination.

Staff	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Designer	4	4	3	3	2	2	1
Developer	0	0	1	2	4	4	3
Tester	0	0	0	0	2	2	2
Total	4	4	4	5	8	8	6

16. CASE stands for Computer Aided Software Engineering. It means, development and maintenance of software projects with help of various automated software tools.
CASE Tools

CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.

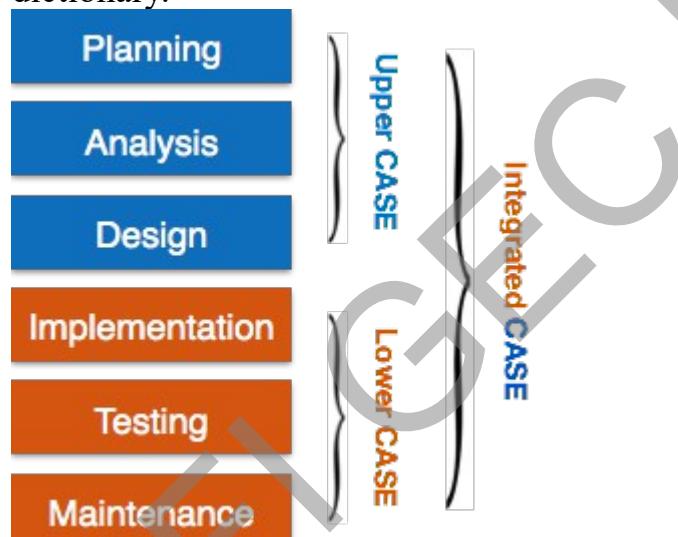
There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.

Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

Components of CASE Tools

CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:

- **Central Repository** - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.



- **Upper CASE Tools** - Upper CASE tools are used in planning, analysis and design stages of SDLC.
- **Lower CASE Tools** - Lower CASE tools are used in implementation, testing and maintenance.
- **Integrated Case Tools** - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

CASE tools can be grouped together if they have similar functionality, process activities and capability of getting integrated with other tools.

Scope of Case Tools

The scope of CASE tools goes throughout the SDLC.

Case Tools Types

Now we briefly go through various CASE tools

Diagram tools

These tools are used to represent system components, data and control flow among

various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.

Process Modeling Tools

Process modeling is method to create software process model, which is used to develop the software. Process modeling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF Composer

Project Management Tools

These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

Documentation Tools

Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project.

Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.

Analysis Tools

These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.

Design Tools

These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design

Configuration Management Tools

An instance of software is released under one version. Configuration Management tools deal with –

- Version and revision management
- Baseline configuration management
- Change control management

CASE tools help in this by automatic tracking, version management and release management. For example, Fossil, Git, Accu REV.

Change Control Tools

These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released. CASE tools automate change tracking, file management, code management and more. It also helps in enforcing change policy of the organization.

Programming Tools

These tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse.

Prototyping Tools

Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product.

Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information. In addition, they provide simulation of software prototype. For example, Serena prototype composer, Mockup Builder.

Web Development Tools

These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect, Foundation 3, Brackets.

Quality Assurance Tools

Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. For example, SoapTest, AppsWatch, JMeter.

Maintenance Tools

Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP Quality Center.

17. Software Configuration Management

- manages evolving software systems
- controls the costs involved in making changes to a system

SCM identifies some procedures that must be defined for each software project to ensure that a good SCM process is implemented. Those are following:

•Identification

•Control

•Status Accounting/Monitoring

•Authentication

Most of this section will cover traditional SCM theory. Do not consider this as boring subject since this section defines and explains the terms that will be used throughout this document.

Identification

Software is usually made up of several programs. Each program has its related

documentation and data can be called as a “configurable item”(CI). The number of CI in any software project that make up a CI is a decision made of the project. The end product is made up of a bunches of CIs.

The status of the CIs at a given point in time is called as a baseline. The baseline serves as a reference point in the software development life cycle. Each new baseline is the sum total of an older baseline plus a series of approved changes made on the CIs.

Control

The process of deciding, coordinating the approved changes for the proposed CIs and implementing all the changes on the appropriate baseline is called Configuration control. It should be kept in mind that configuration controls only address the process after changes are approved. The act of evaluating and approving changes to the software comes under the purview of an entirely different process is called change control.

Status Accounting/Monitoring

Configuration status accounting is the keeping process of each release. This procedure involves tracking what all is in each version of software and the changes that lead to this version. Configuration status accounting keeps a record of all the changes made to the previous baseline to reach of the new baseline.

Authentication

The reviews and audits that verify the physical existence of CIs and checks that they are correctly recorded and parts list.

Configuration authentication (CA) is a process of assuring that a new baseline has all the planned and approved changes incorporated. The process involves verifying that all the functional aspects of the software is completed and also the completeness of the delivery in terms of the right programs, documentation and data are being delivered.

The configuration authentication is an audit performed on the delivery before it is opened to the entire world.

18. Four P's of Software Project Management

- The People
- The Product
- The Process
- The Project

Effective software project management focuses on these items (in this order)

- ***The people***

- ✓ Deals with the cultivation of motivated, highly skilled people.
- ✓ Consists of the stakeholders, the team leaders, and the software team

[The Software Engineering Institute has developed a people management capability maturity model (PM-CMM), “to enhance the readiness of software organizations to undertake increasingly complex applications by helping to attract, grow, motivate, deploy, and retain the talent needed to

improve their software development capability". The people management maturity model defines the following key practice areas for software people: recruiting, selection, performance management, training, compensation, career development, organization and work design, and team/culture development.]

➤ ***The product***

- ✓ Product objectives and scope should be established before a project can be planned.
- ✓ Objectives identify the overall goals for the product (from the customer's point of view) without considering how these goals will be achieved.
- ✓ Scope identifies the primary data, functions and behaviors that characterize the product, and more important, attempts to *bound* these characteristics in a quantitative manner.

[Once the product objectives and scope are understood, alternative solutions are considered. The alternatives enable managers and practitioners to select a "best" approach, given the constraints imposed by delivery deadlines, budgetary restrictions, personnel availability, technical interfaces, and myriad other factors.]

- ✓ Alternative solutions should be considered, and technical and management constraints should be identified.

➤ ***The process***

- ✓ The software process provides the framework from which a comprehensive plan for software development can be established

➤ ***The project***

- ✓ Planning and controlling a software project is done for one primary reason...it is the only known way to manage complexity

In a 1998 survey, 26% of software projects failed outright, 46% experienced cost and schedule overruns

19. Project Risk Management

Managers can plan their strategy based on four steps of risk management which prevails in an organization. Following are the steps to manage risks effectively in an organization:

- Risk Identification
- Risk Quantification
- Risk Response
- Risk Monitoring and Control

The step in project risk management:

Risk Identification

Managers face many difficulties when it comes to identifying and naming the risks that occur when undertaking projects. These risks could be resolved through structured or unstructured brainstorming or strategies. It's important to understand that risks pertaining to the project can only be handled by the project manager and

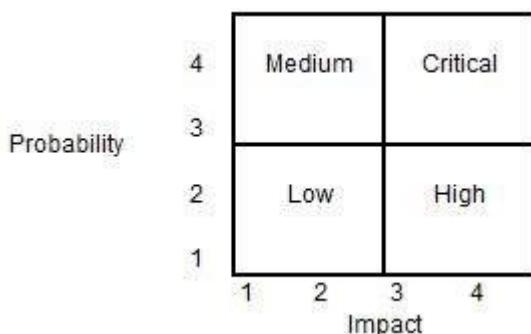
other stakeholders of the project.

Risks, such as operational or business risks will be handled by the relevant teams. The risks that often impact a project are supplier risk, resource risk and budget risk. Supplier risk would refer to risks that can occur in case the supplier is not meeting the timeline to supply the resources required.

Resource risk occurs when the human resource used in the project is not enough or not skilled enough. Budget risk would refer to risks that can occur if the costs are more than what was budgeted.

Risk Quantification

Risks can be evaluated based on quantity. Project managers need to analyze the likely chances of a risk occurring with the help of a matrix.



Using the matrix, the project manager can categorize the risk into four categories as Low, Medium, High and Critical. The probability of occurrence and the impact on the project are the two parameters used for placing the risk in the matrix categories. As an example, if a risk occurrence is low (probability = 2) and it has the highest impact (impact = 4), the risk can be categorized as 'High'.

Risk Response

When it comes to risk management, it depends on the project manager to choose strategies that will reduce the risk to minimal. Project managers can choose between the four risk response strategies, which are outlined below.

- Risks can be avoided
- Pass on the risk
- Take corrective measures to reduce the impact of risks
- Acknowledge the risk

Risk Monitoring and Control

Risks can be monitored on a continuous basis to check if any change is made. New risks can be identified through the constant monitoring and assessing mechanisms.

Risk Management Process

Following are the considerations when it comes to risk management process:

- Each person involved in the process of planning needs to identify and understand the risks pertaining to the project.
- Once the team members have given their list of risks, the risks should be consolidated to a single list in order to remove the duplications.
- Assessing the probability and impact of the risks involved with the help of a matrix.
- Split the team into subgroups where each group will identify the triggers that

lead to project risks.

- The teams need to come up with a contingency plan whereby to strategically eliminate the risks involved or identified.
- Plan the risk management process. Each person involved in the project is assigned a risk in which he/she looks out for any triggers and then finds a suitable solution for it.

Risk Register

Often project managers will compile a document, which outlines the risks involved and the strategies in place. This document is vital as it provides a huge deal of information.

Risk register will often consist of diagrams to aid the reader as to the types of risks that are dealt by the organization and the course of action taken. The risk register should be freely accessible for all the members of the project team.

20. Concept development projects are approached by applying the following major tasks:

1.1 Concept scoping determines the overall scope of the project.

1.2 Preliminary concept planning establishes the organization's ability to undertake the work implied by the project scope.

1.3 Technology risk assessment evaluates the risk associated with the technology to be implemented as part of the project scope.

1.4 Proof of concept demonstrates the viability of a new technology in the software context.

1.5 Concept implementation implements the concept representation in a manner that can be reviewed by a customer and is used for "marketing" purposes when a concept must be sold to other customers or management.

1.6 Customer reaction to the concept solicits feedback on a new technology concept and targets specific customer applications.

A *task network*, also called an *activity network*, is a graphic representation of the task flow for a project. It is sometimes used as the mechanism through which task sequence and dependencies are input to an automated project scheduling tool. In its simplest form (used when creating a macroscopic schedule), the task network depicts major software engineering tasks.

Reg No _____

Name _____

**FIFTH SEMESTER B.TECH DEGREE EXAMINATION MODEL TEST
QUESTION PAPER, March 2018**

**CS 308: SOFTWARE ENGINEERING AND PROJECT
MANAGEMENT**

Max. Marks: 100

Duration: 3 Hours

PART A

Answer all questions. Each carries 3 marks

1. What is software engineering? Discuss the main objectives of software engineering in brief.
2. List the scope of software engineering?
3. Explain the phases of software development?
4. Explain prescriptive model and waterfall model?

PART B

Answer any Two questions. Each carries 9 marks

5.

- a. Explain incremental model and evolutionary model in detail?
- b. Discuss CMMI in detail

6. Discuss about the Software Development Life Cycle? Why is it important to adhere to cycle model while developing a large software product

7. Explain Requirement Elicitation for software.

PART C

Answer all questions. Each carries 3 marks

8. List the objectives of Project Planning

9 Explain the concepts of design process

10 What is the difference between White box and Blackbox testing

11 Describe the walkthrough and inspection in brief

PART D
Answer any Two questions. Each carries 9 marks

- 12 Explain the COCOMO model in detail.
- 13 Discuss the difference between top down and bottom up strategies
- 14 Explain Basis path testing in detail

PART E

Answer any four questions. Each carries 10 marks

- 15 What is software Maintenance .Describe various categories of software maintainence.Which category consume more effort and why
- 16 Discuss Risk monitoring management
- 17 Explain Software Configuration management in detail.
- 18 Explain CASE tools
- 19 Discuss Project Scheduling and tracking
- 20 Explain Product Process Project measures

Reg No _____

Name _____

**FIFTH SEMESTER B.TECH DEGREE EXAMINATION MODEL TEST
QUESTION PAPER, March 2018**

**CS 308: SOFTWARE ENGINEERING AND PROJECT
MANAGEMENT**

Max. Marks: 100

Duration: 3 Hours

PART A

Answer all questions. Each carries 3 marks

1. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software".Software Engineers create, test, maintain, research, and design all kinds of software, from individual applications to operating systems. It is possible to work as a Software Engineer in military, government, medical, industrial, scientific, or business organizations. In addition to the computer science side, engineers also need to serve clients through troubleshooting. A clear resume objective that outlines your mastery over these areas significantly benefits those seeking employment as a Software Engineer.
2. Historical,Economical,Maintainence ,Team Programming.
3. In software engineering, a software development process is the process of dividing software development work into distinct phases to improve design, product management, and project management. It is also known as a software development life cycle. The methodology may include the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application.[1]

Most modern development processes can be vaguely described as agile. Other methodologies include waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, and extreme programming. Some people consider a life-cycle "model" a more general term for a category of methodologies and a software development "process" a more specific term to refer to a specific process chosen by a specific organization. For example, there are many specific software development processes that fit the spiral life-cycle model. The field is often considered a subset of the systems development life cycle

4. The following framework activities are carried out irrespective of the process model chosen by the organization.

1. Communication
2. Planning
3. Modeling
4. Construction
5. Deployment

The name 'prescriptive' is given because the model prescribes a set of activities, actions, tasks, quality assurance and change the mechanism for every project.

There are three types of prescriptive process models. They are:

1. The Waterfall Model
 2. Incremental Process model
 3. RAD model
1. The Waterfall Model

The waterfall model is also called as 'Linear sequential model' or 'Classic life cycle model'.

In this model, each phase is fully completed before the beginning of the next phase.

This model is used for the small projects.

In this model, feedback is taken after each phase to ensure that the project is on the right path.

Testing part starts only after the development is complete.

waterfall model

An alternative design for 'linear sequential model' is as follows:

linear sequential model

Advantages of waterfall model

The waterfall model is simple and easy to understand, implement, and use.

All the requirements are known at the beginning of the project, hence it is easy to manage.

It avoids overlapping of phases because each phase is completed at once.

This model works for small projects because the requirements are understood very well.

This model is preferred for those projects where the quality is more important as compared to the cost of the project.

Disadvantages of the waterfall model

This model is not good for complex and object oriented projects.

It is a poor model for long projects.

The problems with this model are uncovered, until the software testing.

The amount of risk is high.

Incremental Process model

The incremental model combines the elements of waterfall model and they are applied in an iterative fashion.

The first increment in this model is generally a core product.

Each increment builds the product and submits it to the customer for any suggested modifications.

The next increment implements on the customer's suggestions and add additional requirements in the previous increment.

This process is repeated until the product is finished.

For example, the word-processing software is developed using the incremental model.

incremental model

Advantages of incremental model

This model is flexible because the cost of development is low and initial product delivery is faster.

It is easier to test and debug during the smaller iteration.

The working software generates quickly and early during the software life cycle.

The customers can respond to its functionalities after every increment.

Disadvantages of the incremental model

The cost of the final product may cross the cost estimated initially.

This model requires a very clear and complete planning.

The planning of design is required before the whole system is broken into small increments.

The demands of customer for the additional functionalities after every increment causes problem during the system architecture.

3. RAD model

RAD is a Rapid Application Development model.

Using the RAD model, software product is developed in a short period of time.

The initial activity starts with the communication between customer and developer.

Planning depends upon the initial requirements and then the requirements are divided into groups.

Planning is more important to work together on different modules.

The RAD model consist of following phases:

1. Business Modeling

Business modeling consist of the flow of information between various functions in the project.

For example what type of information is produced by every function and which are the functions to handle that information.

A complete business analysis should be performed to get the essential business information.

2. Data modeling

The information in the business modeling phase is refined into the set of objects and it is essential for the business.

The attributes of each object are identified and define the relationship between objects.

3. Process modeling

The data objects defined in the data modeling phase are changed to fulfil the information flow to implement the business model.

The process description is created for adding, modifying, deleting or retrieving a data object.

4. Application generation

In the application generation phase, the actual system is built.

To construct the software the automated tools are used.

5. Testing and turnover

The prototypes are independently tested after each iteration so that the overall testing time is reduced.

The data flow and the interfaces between all the components are are fully tested. Hence, most of the programming components are already tested.

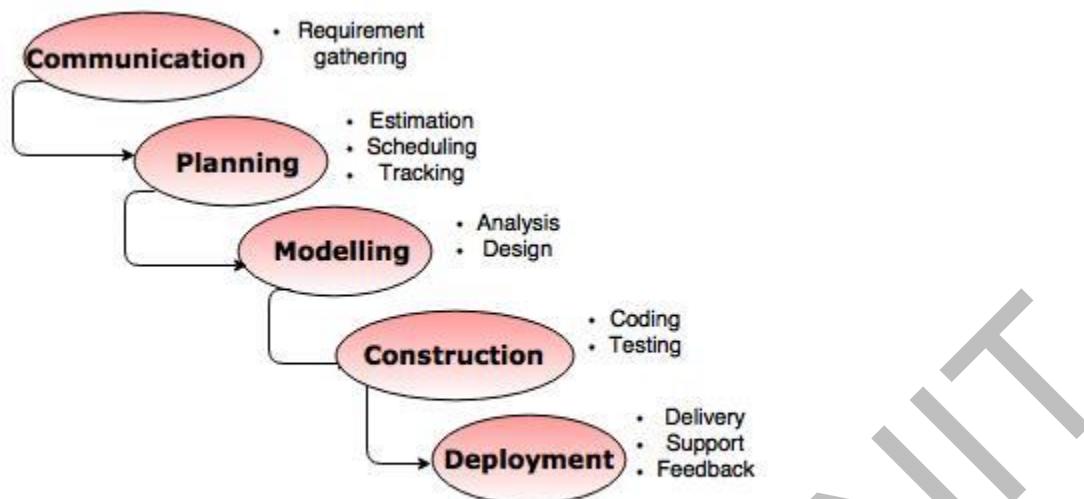


Fig. - The Waterfall model

PART B

Answer any Two questions. Each carries 9 marks

5a.. Evolutionary, Incremental, and High-Risk are software process models for systems engineering ‘in the large’. In the Evolutionary model, the complete cycle of activities is repeated for each version. In the Incremental model, increments are individually designed, tested, and delivered at successive points in time. In the high-risk model, the project is divided into phases and each phase helps constrain risk. In Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle, Ian F. Alexander and Neil Maiden write about the Evolutionary, Incremental, and High-Risk software process models for systems engineering ‘in the large.’

Evolutionary Model

According to Alexander and Maiden, the Evolutionary model for software engineering is structured as follows:

Version 1: User Requirements Definition, System Requirements Definition, System Design/Architecture, Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification.

Version 2: User Requirements Definition, System Requirements Definition, System Design/Architecture, Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification.

Version 3: User Requirements Definition, System Requirements Definition, System Design/Architecture, Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification.

Incremental Model

According to Alexander and Maiden, the Incremental model for software engineering is structured as follows:

User Requirements Definition, System Requirements Definition, System Design/Architecture occur only once up front – User Requirements Definition, System Requirements Definition, System Design/Architecture are factored out of the sequence of incremental deliveries and they occur only once.

Individual increments: The physical increments are individually designed, tested, and delivered at successive points in time.

Increment 1: Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification

Increment 1 and 2: Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification

Increment 1, 2, and 3: Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification

High-Risk Model

According to Alexander and Maiden, the High-Risk model for software engineering is structured as follows:

The project is factored into life-cycle stages: Explore Concept, Proof of Concept, Full Development, and Operations.

Explore Concept – The dominant activities during Explore Concept are User Requirements and System Requirements.

Proof of Concept – The dominant activities during Proof of Concept are System Design/Architecture and Preliminary Design.

Full Development – The dominant activities during Full Development are Detailed Design, Implementation, Test and Integration, and Acceptance and Certification.

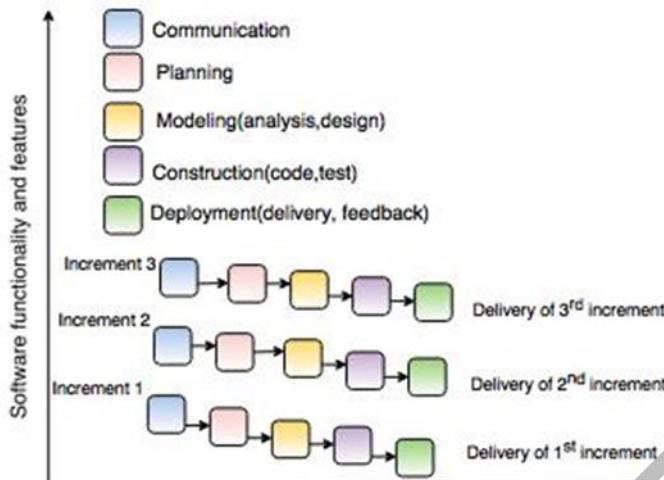


Fig. - Incremental Process Model

5b The Capability Maturity Model (CMM) is a methodology used to develop and refine an organization's software development process. The model describes a five-level evolutionary path of increasingly organized and systematically more mature processes.

The CMM is similar to ISO 9001, one of the ISO 9000 series of standards specified by the International Organization for Standardization (ISO). The ISO 9000 standards specify an effective quality system for manufacturing and service industries; ISO 9001 deals specifically with software development and maintenance. The main difference between the two systems lies in their respective purposes: ISO 9001 specifies a minimal acceptable quality level for software processes, while the CMM establishes a framework for continuous process improvement and is more explicit than the ISO standard in defining the means to be employed to that end.

CMM's Five Maturity Levels of Software Processes

At the initial level, processes are disorganized, even chaotic. Success is likely to depend on individual efforts, and is not considered to be repeatable, because processes would not be sufficiently defined and documented to allow them to be replicated.

At the repeatable level, basic project management techniques are established, and successes could be repeated, because the requisite processes would have been made established, defined, and documented.

At the defined level, an organization has developed its own standard software process through greater attention to documentation, standardization, and integration.

At the managed level, an organization monitors and controls its own processes through data collection and analysis.

At the optimizing level, processes are constantly being improved through monitoring feedback from current processes and introducing innovative processes to better serve the organization's particular needs.

6 various SDLC methodologies have been developed to guide the processes involved, including the waterfall model (which was the original SDLC method); rapid application development (RAD); joint application development (JAD); the fountain model; the spiral model; build and fix; and synchronize-and-stabilize. Frequently, several models are combined into some sort of hybrid methodology. Documentation is crucial regardless of the type of model chosen or devised for any application, and is usually done in parallel with the development process. Some methods work better for specific types of projects, but in the final analysis, the most important factor for the success of a project may be how closely the particular plan was followed.

In general, an SDLC methodology follows the following steps:

The existing system is evaluated. Deficiencies are identified. This can be done by interviewing users of the system and consulting with support personnel.

The new system requirements are defined. In particular, the deficiencies in the existing system must be addressed with specific proposals for improvement.

The proposed system is designed. Plans are laid out concerning the physical construction, hardware, operating systems, programming, communications, and security issues.

The new system is developed. The new components and programs must be obtained and installed. Users of the system must be trained in its use, and all aspects of performance must be tested. If necessary, adjustments must be made at this stage.

The system is put into use. This can be done in various ways. The new system can phased in, according to application or location, and the old system gradually replaced. In some cases, it may be more cost-effective to shut down the old system and implement the new system all at once.

Once the new system is up and running for a while, it should be exhaustively evaluated. Maintenance must be kept up rigorously at all times. Users of the system should be kept up-to-date concerning the latest modifications and procedures.

7 In requirements engineering, requirements elicitation is the practice of collecting the requirements of a system from users, customers and other stakeholders.[1] The practice is also sometimes referred to as "requirement gathering".

The term elicitation is used in books and research to raise the fact that good requirements cannot just be collected from the customer, as would be indicated by the name requirements gathering. Requirements elicitation is non-trivial because you can never be sure you get all requirements from the user and customer by just asking them what the system should do OR NOT do (for Safety and Reliability). Requirements elicitation practices include interviews, questionnaires, user observation, workshops, brainstorming, use cases, role playing and prototyping.

Before requirements can be analyzed, modeled, or specified they must be gathered through an elicitation process. Requirements elicitation is a part of the requirements engineering process, usually followed by analysis and specification of the requirements.

Commonly used elicitation processes are the stakeholder meetings or interviews[2]. For example, an important first meeting could be between software engineers and customers where they discuss their perspective of the requirements.

. The requirements elicitation process may appear simple: ask the customer, the users and others what the objectives for the system or product are, what is to be accomplished, how the system or product fits into the needs of business, and finally, how the system or product is to be used on a day-to-day basis. However, issues may arise that complicate the process.

In 1992, Christel and Kang identified problems that indicate the challenges for requirements elicitation:

'Problems of scope'. The boundary of the system is ill-defined or the customers/users specify unnecessary technical details that may confuse, rather than clarify, overall system objectives.

Problems of understanding. The customers/users are not completely sure of what is needed, have a poor understanding of the capabilities and limitations of their computing environment, don't have a full understanding of the problem domain, have trouble communicating needs to the system engineer, omit information that is believed to be "obvious," specify requirements that conflict with the needs of other customers/users, or specify requirements that are ambiguous or untestable.

Problems of volatility. The requirements change over time. The rate of change is sometimes referred to as the level of requirement volatility

Requirements quality can be improved through these approaches:[4]

Visualization. Using tools that promote better understanding of the desired end-product such as visualization and simulation.

Consistent language. Using simple, consistent definitions for requirements described in natural language and use the business terminology that is prevalent in the enterprise.

Guidelines. Following organizational guidelines that describe the collection techniques and the types of requirements to be collected. These guidelines are then used consistently across projects.

Consistent use of templates. Producing a consistent set of models and templates to document the requirements.

Documenting dependencies. Documenting dependencies and interrelationships among requirements.

Analysis of changes. Performing root cause analysis of changes to requirements and project Management has developed in order to plan, co-ordinate and control the complex and diverse activities of modern industrial and commercial projects. All projects share one common characteristic - the projection of ideas and activities into new endeavours.

PART C

Answer all questions. Each carries 3 marks

8. The purpose of project management is to foresee or predict as many dangers and problems as possible; and to plan, organise and control activities so that the project is completed as successfully as possible in spite of all the risks. The ever-present element of risk and uncertainty means that events and tasks leading to completion can never be foretold with absolute accuracy. For some complex or advanced projects, even the possibility of successful completion might be of serious doubt.

Project management can involve the following activities: planning - deciding what is to be done; organising - making arrangements; staffing - selecting the right people for the job; directing - giving instructions; monitoring - checking on progress; controlling - taking action to remedy hold ups; innovation - coming up with new solutions; representing - liaising with users.

Setting Objectives

Effective objectives in project management are specific. A specific objective increases the chances of leading to a specific outcome. Therefore objectives shouldn't be vague, such as "to improve customer relations," because they are not measurable. Objectives should show how successful a project has been, for example "to reduce customer complaints by 50%" would be a good objective. The measure can be, in some cases, a simple yes or no answer, for example, "did we reduce the number of customer complaints by 50%?"

While there may be one major project objective, in pursuing it there may be interim project objectives. In lots of instances, project teams are tasked with achieving a series of objectives in pursuit of the final objective. In many cases, teams can only proceed in a stair step fashion to achieve the desired outcome. If they were to proceed in any other manner, they may not be

able to develop the skills or insights along the way that will enable them to progress in a productive manner.

Objectives can often be set under three headings:

1. Performance and Quality

The end result of a project must fit the purpose for which it was intended. At one time, quality was seen as the responsibility of the quality control department. In more recent years the concept of total quality management has come to the fore, with the responsibility for quality shared by all staff from top management downwards.

2. Budget

The project must be completed without exceeding the authorised expenditure. Financial sources are not always inexhaustible and a project might be abandoned altogether if funds run out before completion. If that was to happen, the money and effort invested in the project would be forfeited and written off. In extreme cases the project contractor could face ruin. There are many projects where there is no direct profit motive, however it is still important to pay proper attention to the cost budgets, and financial management remains essential.

3. Time to Completion

Actual progress has to match or beat planned progress. All significant stages of the project must take place no later than their specified dates, to result in total completion on or before the planned finish date. The timescale objective is extremely important because late completion of a project is not very likely to please the project purchaser or the sponsor. making corrective actions

9. One framing of the engineering design process delineates the following stages: research, conceptualization, feasibility assessment, establishing design requirements, preliminary design, detailed design, production planning and tool design, and production.[2] Others, noting that "different authors (in both research literature and in textbooks) define different phases of the design process with varying activities occurring within them," have suggested more simplified/generalized models - such as problem definition, conceptual design, preliminary design, detailed design, and design communication.[3] A standard summary of the process in European engineering design literature is that of clarification of the task, conceptual design, embodiment design, detail design.[4] In these examples, other key aspects - such as concept evaluation and prototyping - are subsets and/or extensions of one or more of the listed steps. It's also important to understand that in these as well as other articulations of the process, different terminology employed may have varying degrees of overlap, which affects what steps get stated explicitly or deemed "high level" versus subordinate in any given model.

The source of information should be relevant, including existing solutions. Reverse engineering can be an effective technique if other solutions are available on the market.[5] Other sources of information include the Internet, local libraries, available government documents, personal organizations, trade journals, vendor catalogs and individual experts available.

Design requirements[edit]

Establishing design requirements and conducting requirement analysis, sometimes termed problem definition (or deemed a related activity), is one of the most important elements in the design process,[6] and this task is often performed at the same time as a feasibility analysis. The design requirements control the design of the product or process being developed, throughout the engineering design process. These include basic things like the functions, attributes, and specifications - determined after assessing user needs. Some design requirements include hardware and software parameters, maintainability, availability, and testability.

Feasibility

In some cases, a feasibility study is carried out after which schedules, resource plans and estimates for the next phase are developed. The feasibility study is an evaluation and analysis of the potential of a proposed project to support the process of decision making. It outlines and analyses alternatives or methods of achieving the desired outcome. The feasibility study helps to narrow the scope of the project to identify the best scenario. A feasibility report is generated following which Post Feasibility Review is performed.

The purpose of a feasibility assessment is to determine whether the engineer's project can proceed into the design phase. This is based on two criteria: the project needs to be based on an achievable idea, and it needs to be within cost constraints. It is important to have engineers with experience and good judgment to be involved in this portion of the feasibility study.[2]

Conceptualization[edit]

A concept study (conceptualization, conceptual design) is often a phase of project planning that includes producing ideas and taking into account the pros and cons of implementing those ideas. This stage of a project is done to minimize the likelihood of error, manage costs, assess risks, and evaluate the potential success of the intended project. In any event, once an engineering issue or problem is defined, potential solutions must be identified. These solutions can be found by using ideation, the mental process by which ideas are generated. In fact, this step is often termed Ideation or "Concept Generation." The following are widely used techniques

trigger word - a word or phrase associated with the issue at hand is stated, and subsequent words and phrases are evoked.

morphological analysis - independent design characteristics are listed in a chart, and different engineering solutions are proposed for each solution. Normally, a preliminary sketch and short report accompany the morphological chart.

synectics - the engineer imagines him or herself as the item and asks, "What would I do if I were the system?" This unconventional method of thinking may find a solution to the problem at hand. The vital aspects of the conceptualization step is synthesis. Synthesis is the process of taking the element of the concept and arranging them in the proper way. Synthesis creative process is present in every design.

brainstorming - this popular method involves thinking of different ideas, typically as part of a small group, and adopting these ideas in some form as a solution to the problem

Various generated ideas must then undergo a concept evaluation step, which utilizes various tools to compare and contrast the relative strengths and weakness of possible alternatives.

Preliminary design[edit]

The preliminary design, or high-level design includes (also called FEED), often bridges a gap between design conception and detailed design, particularly in cases where the level of conceptualization achieved during ideation is not sufficient for full evaluation. So in this task, the overall system configuration is defined, and schematics, diagrams, and layouts of the project may provide early project configuration. (This notably varies a lot by field, industry, and product.) During detailed design and optimization, the parameters of the part being created will change, but the preliminary design focuses on creating the general framework to build the project on.[2]

S. Blanchard and J. Fabrycky describe it as: "The 'whats' initiating conceptual design produce 'hows' from the conceptual design evaluation effort applied to feasible conceptual design concepts. Next, the 'hows' are taken into preliminary design through the means of allocated requirements. There they become 'whats' and drive preliminary design to address 'hows' at this lower level."

Detailed design[edit]

Following FEED is the Detailed Design (Detailed Engineering) phase, which may consist of procurement of materials as well. This phase further elaborates each aspect of the project/product by complete description through solid modeling, drawings as well as specifications.

Design for manufacturability[edit]

Design for manufacturability (DFM) is the general engineering art of designing products in such a way that they are easy to manufacture.

Operating parameters

Operating and nonoperating environmental stimuli

Test requirements

External dimensions

Maintenance and testability provisions

Materials requirements

Reliability requirements

External surface treatment

Design life

Packaging requirements

External marking

Computer-aided design (CAD) programs have made detailed design phase more efficient. For example, a CAD program can provide optimization to reduce volume without hindering a part's quality. It can also calculate stress and displacement using the finite element method to determine stresses throughout the part. [7]

Production planning[edit]

The production planning and tool design consists of planning how to mass-produce the product and which tools should be used in the manufacturing process. Tasks to complete in this step include selecting materials, selection of the production processes, determination of the sequence of operations, and selection of tools such as jigs, fixtures, metal cutting and metal or plastic

Software testing is one of the best means to affirm the quality of software and deliver an error-free application. Over the years, software testing has matured into a separate discipline giving way to several different testing techniques that have been introduced, analyzed and studied in this area. Black box testing and white box testing are two such testing approaches that are quite commonly used by software testers.

10.What is Black Box Testing?

Crudely put, when the tester has no idea of the internal working of the system which he is testing, that approach is called black box testing.

In this case, the system under test is viewed as a “black box”.

Requirements Document or Functional Specification Document forms the basis of this testing, which requires the user to understand the processes within the software.

black box testing

How to write Test Cases for Black Box Testing? The tester examines requirements and specifications of the system.

The tester explores the system's UI and functionality to understand how the processes on the system are expected to work.

Tester designs test cases with valid inputs and the corresponding expected outputs.

Tester also includes some negative test cases with invalid inputs and expected outputs (error messages/program termination) as applicable.

Techniques of Black Box Testing

In case of black box testing, inputs to the test cases are the driving factor. Any one of the three techniques discussed below can be used to choose the inputs during the black box testing process

Boundary Value Analysis: This approach is focused on testing the boundary values associated with the system. This approach aims at testing the boundaries of the input domain that have the highest probability of giving erroneous outputs.

Equivalence Class Partitioning: In this approach, a limited set of functions is identified along with its corresponding valid and invalid inputs and expected outputs. This approach aims at identifying classes of errors and therefore reducing the number of test cases required.

Error Guessing: An experienced tester most often uses this approach to first identify the defects and then develop corresponding test cases.

What is White Box Testing?

In white box testing methodology, the tester has the knowledge of the internals of a system and knows how the system is implemented. The tester uses this knowledge to develop test cases that will examine the control flow, information flow, data flow, exception and error handling as well as coding practices of the system.

white box testing

How to write Test Cases for White Box Testing?

The tester analyzes and understands the structure of the system by examining its code.

The tester understands the weak spots within the code that is most prone to defects.

The tester develops test cases to cover individual data/information/ control flows and branches within the code.

The tester also develops test cases to test proper working of all the functionalities and error handing of the system.

Techniques of White Box Testing When it comes to white box testing, the knowledge that the tester possesses about the system is the driving factor, which helps the tester to devise test cases aimed at discovering defects with the internal working of the system.

Statement Tests: All the statements within the code must have a test case associated with its such that each statement must be executed at least once during the testing cycle.

Decision Tests: All the decision directions must be executed at least once during the testing life cycle.

Branch Condition Tests: All the conditions in a specific decision must be tested for proper working at least once.

Decision/Condition Tests: All the combination of the possible conditions within a specific decision for all the decisions is to be tested.

Data Flow Tests: This will ensure that all the variables and data that are used within the system are tested by passing the specific variables through each possible calculation.

Multiple Condition Tests: This will ensure that each point of entry within the code is tested at least once during the testing life cycle.

11. Review is "A process or meeting during which artifacts of software product are examined by project stockholders, user representatives, or other interested parties for feedback or approval". Software Review can be on Technical specifications, designs, source code, user documentation, support and maintenance documentation, test plans, test specifications, standards, and any other type of specific to work product, it can be conducted at any stage of the software development life cycle.

Purpose of conducting review is to minimize the defect ratio as early as possible in Software Development life cycle. As a general principle, the earlier a document is reviewed, the greater will be the impact of its defects on any downstream activities and their work products. Magnitude cost of defect fixing after the release of the product is around 60-100x. Review can be formal or informal. Informal reviews are referred as walkthrough and formal as Inspection.

Walkthrough: Method of conducting informal group/individual review is called walkthrough, in which a designer or programmer leads members of the development team and other interested parties through a software product, and the participants ask questions and make comments about possible errors, violation of development standards, and other problems or may suggest improvement on the article, walkthrough can be pre planned or can be conducted at need basis and generally people working on the work product are involved in the walkthrough process.

The Purpose of walkthrough is to:

- Find problems
- Discuss alternative solutions
- Focusing on demonstrating how work product meets all requirements.IEEE 1028 recommends three specialist roles in a walkthrough:

Leader: who conducts the walkthrough, handles administrative tasks, and ensures orderly conduct (and who is often the Author)

Recorder: who notes all anomalies (potential defects), decisions, and action items identified during the walkthrough meeting, normally generate minutes of meeting at the end of walkthrough session.

Author: who presents the software product in step-by-step manner at the walk-through meeting, and is probably responsible for completing most action items.

Walkthrough Process: Author describes the artifact to be reviewed to reviewers during the meeting. Reviewers present comments, possible defects, and improvement suggestions to the author. Recorder records all defect, suggestion during walkthrough meeting. Based on reviewer comments, author performs any necessary rework of the work product if required. Recorder prepares minutes of meeting and sends the relevant stakeholders and leader is normally to monitor overall walkthrough meeting activities as per the defined company process or responsibilities for conducting the reviews, generally performs monitoring activities, commitment against action items etc.

Inspection: An inspection is a formal, rigorous, in-depth group review designed to identify problems as close to their point of origin as possible., Inspection is a recognized industry best practice to improve the quality of a product and to improve productivity, Inspections is a formal review and generally need is predefined at the start of the product planning, The objectives of the inspection process are to

- Find problems at the earliest possible point in the software development process
- Verify that the work product meets its requirement
- Ensure that work product has been presented according to predefined standards
- Provide data on product quality and process effectiveness
- Inspection advantages are to build technical knowledge and skill among team members by reviewing the output of other people
- Increase the effectiveness of software testing.

IEEE 1028 recommends three following roles in an Inspection:

Inspector Leader: The inspection leader shall be responsible for administrative tasks pertaining to the inspection, shall be responsible for planning and preparation, shall ensure

that the inspection is conducted in an orderly manner and meets its objectives, should be responsible for collecting inspection data

Recorder: The recorder should record inspection data required for process analysis. The inspection leader may be the recorder.

Reader: The reader shall lead the inspection team through the software product in a comprehensive and logical fashion, interpreting sections of the work product and highlighting important aspects

Author: The author shall be responsible for the software product meeting its inspection entry criteria, for contributing to the inspection based on special understanding of the software product, and for performing any rework required to make the software product meet its inspection exit criteria.

Inspector: Inspectors shall identify and describe anomalies in the software product. Inspectors shall be chosen to represent different viewpoints at the meeting (for example, sponsor, requirements, design, code, safety, test, independent test, project management, quality management, and hardware engineering). Only those viewpoints pertinent to the inspection of the product should be present. Some inspectors should be assigned specific review topics to ensure effective coverage. For example, one inspector may focus on conformance with a specific standard or standards, another on syntax, and another for overall coherence. These roles should be assigned by the inspection leader when planning the inspection.

All participants in the review are inspectors. The author shall not act as inspection leader and should not act as reader or recorder. Other roles may be shared among the team members. Individual participants may act in more than one role. Individuals holding management positions over any member of the inspection team shall not participate in the inspection

Inspection Process: Following are review phases:

- Planning
- Overview
- Preparation
- Examination meeting

Planning:

- Inspection Leader perform following task in planning phase
- Determine which work products need to be inspected
- Determine if a work product that needs to be inspected is ready to be inspected
- Identify the inspection team
- Determine if an overview meeting is needed.

The moderator ensures that all inspection team members have had inspection process training. The moderator obtains a commitment from each team member to participate. This commitment means the person agrees to spend the time required to perform his or her assigned role on the team. Identify the review materials required for the inspection, and distribute materials to relevant stakeholders.

Overview: Purpose of the overview meeting is to educate inspectors; meeting is lead by Inspector lead and is presented by author, overview is presented for the inspection, this meeting normally acts as optional meeting, purpose to sync the entire participant and the area to be inspected.

Preparation: Objective of the preparation phase is to prepare for the inspection meeting by critically reviewing the review materials and the work product, participant drill down on the document distributed by the lead inspector and identify the defect before the meeting.

Examination meeting: The objective of the inspection meeting is to identify final defect list in the work product being inspected, based on the initial list of defects prepared by the inspectors [identified at preparation phase and the new one found during the inspection meeting. The Lead Auditor opens the meeting and describes the review objectives and area to be inspected. Identify that all participants are well familiar with the content material, Reader reads the meeting material and inspector finds out any inconsistency, possible defects, and improvement suggestions to the author. Recorder records all the discussion during the inspection meeting, and mark actions against the relevant stakeholders. Lead Inspector may take decision that if there is need of follow up meeting. Author updates the relevant document if required on the basis of the inspection meeting discussion.

Rework and Follow-up: Objective is to ensure that corrective action has been taken to correct problems found during an inspection.

PART D

Answer any Two questions. Each carries 9 marks

12. The constructive cost model was developed by Barry W. Boehm in the late 1970s[1] and published in Boehm's 1981 book Software Engineering Economics[2] as a model for estimating effort, cost, and schedule for software projects. It drew on a study of 63 projects at TRW Aerospace where Boehm was Director of Software Research and Technology. The study examined projects ranging in size from 2,000 to 100,000 lines of code, and programming languages ranging from assembly to PL/I. These projects were based on the waterfall model of software development which was the prevalent software development process in 1981.

References to this model typically call it COCOMO 81. In 1995 COCOMO II was developed and finally published in 2000 in the book Software Cost Estimation with COCOMO II.[3]

COCOMO II is the successor of COCOMO 81 and is claimed to be better suited for estimating modern software development projects; providing support for more recent software development processes and was tuned using a larger database of 161 projects. The need for the new model came as software development technology moved from mainframe and overnight batch processing to desktop development, code reusability, and the use of off-the-shelf software components. This article refers to COCOMO 81.

COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. The first level, Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors to account for difference in project attributes (Cost Drivers). Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases.

Basic COCOMO[edit]

Basic COCOMO compute software development effort (and cost) as a function of program size. Program size is expressed in estimated thousands of source lines of code (SLOC, KLOC).

COCOMO applies to three classes of software projects:

Organic projects - "small" teams with "good" experience working with "less than rigid" requirements

Semi-detached projects - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements

Embedded projects - developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects.(hardware, software, operational, ...)

The basic COCOMO equations take the form

$$\text{Effort Applied (E)} = ab(\text{KLOC})bb \text{ [man-months]}$$

$$\text{Development Time (D)} = cb(\text{Effort Applied})db \text{ [months]}$$

$$\text{People required (P)} = \text{Effort Applied} / \text{Development Time} \text{ [count]}$$

where, KLOC is the estimated number of delivered lines (expressed in thousands) of code for project. The coefficients ab, bb, cb and db are given in the following table (note: the values listed below are from the original analysis, with a modern reanalysis[4] producing different values):

Software project	ab	bb	cb	db
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO is good for quick estimate of software costs. However it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and so on.

Intermediate COCOMOs[edit]

Intermediate COCOMO computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes. This extension considers a set of four "cost drivers", each with a number of subsidiary attributes:-

Product attributes

Required software reliability extent

Size of application database

Complexity of the product

Hardware attributes

Run-time performance constraints

Memory constraints

Volatility of the virtual machine environment

Required turnabout time

Personnel attributes

Analyst capability

Software engineering capability
 Applications experience
 Virtual machine experience
 Programming language experience
 Project attributes
 Use of software tools
 Application of software engineering methods

Required development schedule

Each of the 15 attributes receives a rating on a six-point scale that ranges from "very low" to "extra high" (in importance or value). An effort multiplier from the table below applies to the rating. The product of all effort multipliers results in an effort adjustment factor (EAF). Typical values for EAF range from 0.9 to 1.4.

Cost Drivers	Ratings	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes							
Required software reliability	0.75	0.88	1.00	1.15	1.40		
Size of application database		0.94	1.00	1.08	1.16		
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65	
Hardware attributes							
Run-time performance constraints				1.00	1.11	1.30	1.66
Memory constraints		1.00	1.06	1.21	1.56		
Volatility of the virtual machine environment				0.87	1.00	1.15	1.30
Required turnabout time		0.87	1.00	1.07	1.15		
Personnel attributes							
Analyst capability	1.46	1.19	1.00	0.86	0.71		
Applications experience	1.29	1.13	1.00	0.91	0.82		
Software engineer capability	1.42	1.17	1.00	0.86	0.70		

Virtual machine experience	1.21	1.10	1.00	0.90
Programming language experience	1.14	1.07	1.00	0.95

Project attributes

Application of software engineering methods		1.24	1.10	1.00	0.91	0.82
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule		1.23	1.08	1.00	1.04	1.10

The Intermediate Cocomo formula now takes the form:

$$E = ai(KLoC)(bi)(EAF)$$

where E is the effort applied in person-months, KLoC is the estimated number of thousands of delivered lines of code for the project, and EAF is the factor calculated above. The coefficient ai and the exponent bi are given in the next table.

Software project	ai	bi
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

The Development time D calculation uses E in the same way as in the Basic COCOMO.

Detailed COCOMO[edit]

Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process.

The detailed model uses different effort multipliers for each cost driver attribute. These Phase Sensitive effort multipliers are each to determine the amount of effort required to complete each phase. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software life cycle.

A Detailed project schedule is never static.

The Six phases of detailed COCOMO are:-

planning and requirements
system design
detailed design
module code and test
integration and test

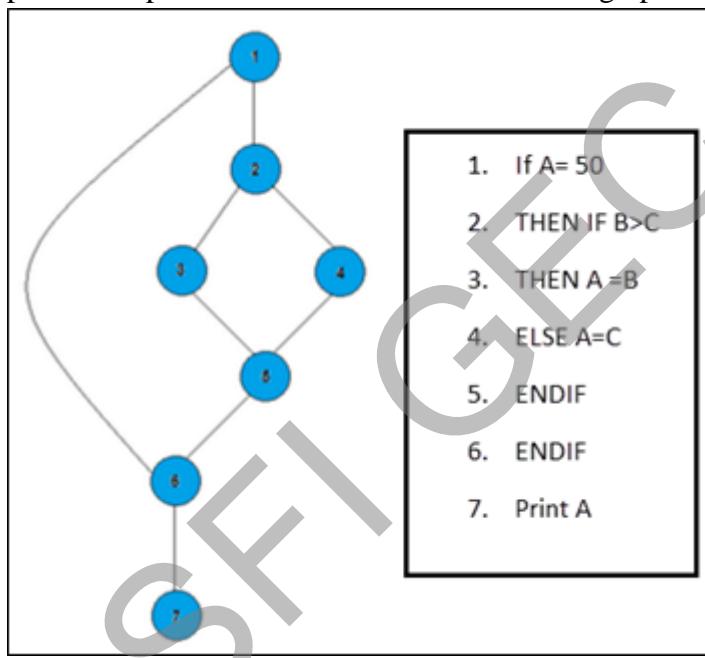
13. Top-down and bottom-up are both strategies of information processing and knowledge ordering, used in a variety of fields including software, humanistic and scientific theories (see systemics), and management and organization. In practice, they can be seen as a style of thinking, teaching, or leadership.

A top-down approach (also known as stepwise design and in some cases used as a synonym of decomposition) is essentially the breaking down of a system to gain insight into its compositional sub-systems in a reverse engineering fashion. In a top-down approach an overview of the system is formulated, specifying, but not detailing, any first-level subsystems. Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements. A top-down model is often specified with the assistance of "black boxes", which makes it easier to manipulate. However, black boxes may fail to clarify elementary mechanisms or be detailed enough to realistically validate the model. Top down approach starts with the big picture. It breaks down from there into smaller segments.[1]

A bottom-up approach is the piecing together of systems to give rise to more complex systems, thus making the original systems sub-systems of the emergent system. Bottom-up processing is a type of information processing based on incoming data from the environment to form a perception. From a cognitive psychology perspective, information enters the eyes in one direction (sensory input, or the "bottom"), and is then turned into an image by the brain

that can be interpreted and recognized as a perception (output that is "built up" from processing to final cognition). In a bottom-up approach the individual base elements of the system are first specified in great detail. These elements are then linked together to form larger subsystems, which then in turn are linked, sometimes in many levels, until a complete top-level system is formed. This strategy often resembles a "seed" model, by which the beginnings are small but eventually grow in complexity and completeness. However, "organic strategies" may result in a tangle of elements and subsystems, developed in isolation and subject to local optimization as opposed to meeting a global purpose.

14. In software engineering, basis path testing, or structured testing,[1] is a white box method for designing test cases. The method analyzes the control flow graph of a program to find a set of linearly independent paths of execution. The method normally uses McCabe's cyclomatic complexity to determine the number of linearly independent paths and then generates test cases for each path thus obtained.[2] Basis path testing guarantees complete branch coverage (all edges of the control flow graph), but achieves that without covering all possible paths of the control flow graph—the latter is usually too costly



PART E

Answer any four questions. Each carries 10 marks

15 Definition: Software maintenance is a part of Software Development Life Cycle. Its main purpose is to modify and update software application after delivery to correct faults and to

improve performance. Software is a model of the real world. When the real world changes, the software requires alteration wherever possible.

Description: Software maintenance is a vast activity which includes optimization, error correction, deletion of discarded features and enhancement of existing features. Since these changes are necessary, a mechanism must be created for estimation, controlling and making modifications. The essential part of software maintenance requires preparation of an accurate plan during the development cycle. Typically, maintenance takes up about 40-80% of the project cost, usually closer to the higher pole. Hence, a focus on maintenance definitely helps keep costs down.

Software Maintenance Processes are:

- The SM process includes a maintenance plan which contains software preparation, problem identification and find out about product configuration management.
- The problem analysis process includes checking validity, examining it and coming up with a solution and finally getting all the required support to apply for modification.
- The process acceptance by confirming the changes with the individual who raised the request.
- The platform migration process, which is used if software is needed to be ported to another platform without any change in functionality.

Some software points that affect maintenance cost include:

- Structure of Software Program
- Programming Language
- Dependence on external environment

16. Software development is activity that uses a variety of technological advancements and requires high levels of knowledge. Because of these and other factors, every software development project contains elements of uncertainty. This is known as project risk. The success of a software development project depends quite heavily on the amount of risk that corresponds to each project activity. As a project manager, it's not enough to merely be aware of the risks. To achieve a successful outcome, project leadership must identify, assess, prioritize, and manage all of the major risks.

The goal of most software development and software engineering projects is to be distinctive—often through new features, more efficiency, or exploiting advancements in software engineering. Any software project executive will agree that the pursuit of such opportunities cannot move forward without risk.

Because risks are painfully real and quite prevalent on all software projects, it's critically necessary that stakeholders work hard to identify, understand, and mitigate any risks that might threaten the success of a project. For projects that have time and cost constraints, our experience shows most clearly that successful software development efforts are those in which risk mitigation is a central management

What Is Risk In Software Engineering?

Very simply, a risk is a potential problem. It's an activity or event that may compromise the success of a software development project. Risk is the possibility of suffering loss, and total risk exposure to a specific project will account for both the probability and the size of the potential loss.

Guesswork and crisis-management are never effective. Identifying and aggregating risks is the only predictive method for capturing the probability that a software development project will experience unplanned or inadmissible events. These include terminations, discontinuities, schedule delays, cost underestimation, and overrun of project resources

What Is Risk Management In Software Engineering?

Risk management means risk containment and mitigation. First, you've got to identify and plan. Then be ready to act when a risk arises, drawing upon the experience and knowledge of the entire team to minimize the impact to the project.

Risk management includes the following tasks:

Identify risks and their triggers

Classify and prioritize all risks

Craft a plan that links each risk to a mitigation

Monitor for risk triggers during the project

Implement the mitigating action if any risk materializes

Communicate risk status throughout project

Identify and Classify Risks

Most software engineering projects are inherently risky because of the variety potential problems that might arise. Experience from other software engineering projects can help managers classify risk. The importance here is not the elegance or range of classification, but rather to precisely identify and describe all of the real threats to project success. A simple but effective classification scheme is to arrange risks according to the areas of impact

Five Types of Risk In Software Project Management

For most software development projects, we can define five main risk impact areas:

New, unproven technologies

User and functional requirements

Application and system architecture

Performance

Organizational

New, unproven technologies. The majority of software projects entail the use of new technologies. Ever-changing tools, techniques, protocols, standards, and development systems increase the probability that technology risks will arise in virtually any substantial software engineering effort. Training and knowledge are of critical importance, and the improper use of new technology most often leads directly to project failure.

User and functional requirements. Software requirements capture all user needs with respect to the software system features, functions, and quality of service. Too often, the process of requirements definition is lengthy, tedious, and complex. Moreover, requirements usually change with discovery, prototyping, and integration activities. Change in elemental requirements will likely propagate throughout the entire project, and modifications to user requirements might not translate to functional requirements. These disruptions often lead to one or more critical failures of a poorly-planned software development project.

Application and system architecture. Taking the wrong direction with a platform, component, or architecture can have disastrous consequences. As with the technological risks, it is vital that the team includes experts who understand the architecture and have the capability to make sound design choices.

Performance. It's important to ensure that any risk management plan encompasses user and partner expectations on performance. Consideration must be given to benchmarks and threshold testing throughout the project to ensure that the work products are moving in the right direction.

Organizational. Organizational problems may have adverse effects on project outcomes. Project management must plan for efficient execution of the project, and find a balance between the needs of the development team and the expectations of the customers. Of course, adequate staffing includes choosing team members with skill sets that are a good match with the project.

Risk Management Plan

After cataloging all of the risks according to type, the software development project manager should craft a risk management plan. As part of a larger, comprehensive project plan, the risk management plan outlines the response that will be taken for each risk—if it materializes.

Monitor and Mitigate

To be effective, software risk monitoring has to be integral with most project activities. Essentially, this means frequent checking during project meetings and critical events

Monitoring includes:

Publish project status reports and include risk management issues

Revise risk plans according to any major changes in project schedule

Review and reprioritize risks, eliminating those with lowest probability

Brainstorm on potentially new risks after changes to project schedule or scope

When a risk occurs, the corresponding mitigation response should be taken from the risk management plan.

Mitigating options include:

Accept: Acknowledge that a risk is impacting the project. Make an explicit decision to accept the risk without any changes to the project. Project management approval is mandatory here.

Avoid: Adjust project scope, schedule, or constraints to minimize the effects of the risk.

Control: Take action to minimize the impact or reduce the intensification of the risk.

Transfer: Implement an organizational shift in accountability, responsibility, or authority to other stakeholders that will accept the risk.

Continue Monitoring: Often suitable for low-impact risks, monitor the project environment for potentially increasing impact of the risk.

17. In software engineering, software configuration management (SCM or S/W CM)[1] is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management.[2] SCM practices include revision control and the establishment of baselines. If something goes wrong, SCM can determine what was changed and who changed it. If a configuration is working well, SCM can determine how to replicate it across many hosts.

The acronym "SCM" is also expanded as source configuration management process and software change and configuration management

The goals of SCM are generally:[citation needed]

Configuration identification - Identifying configurations, configuration items and baselines.

Configuration control - Implementing a controlled change process. This is usually achieved by setting up a change control board whose primary function is to approve or reject all change requests that are sent against any baseline.

Configuration status accounting - Recording and reporting all the necessary information on the status of the development process.

Configuration auditing - Ensuring that configurations contain all their intended parts and are sound with respect to their specifying documents, including requirements, architectural specifications and user manuals.

Build management - Managing the process and tools used for builds.

Process management - Ensuring adherence to the organization's development process.

Environment management - Managing the software and hardware that host the system.

Teamwork - Facilitate team interactions related to the process.

Defect tracking - Making sure every defect has traceability back to the source.

With the introduction of cloud computing the purposes of SCM tools have become merged in some cases. The SCM tools themselves have become virtual appliances that can be instantiated as virtual machines and saved with state and version. The tools can model and manage cloud-based virtual resources, including virtual appliances, storage units, and software bundles. The roles and responsibilities of the actors have become merged as well with developers now being able to dynamically instantiate virtual servers and related resources.

18. Computer-aided software engineering (CASE) is the domain of software tools used to design and implement applications. CASE tools are similar to and were partly inspired by computer-aided design (CAD) tools used for designing hardware products. CASE tools are used for developing high-quality, defect-free, and maintainable software.[1] CASE software is often associated with methods for the development of information systems together with automated tools that can be used in the software Tools support specific tasks in the software life-cycle.

Workbenches combine two or more tools focused on a specific part of the software life-cycle.

Environments combine two or more tools or workbenches and support the complete software life-cycle.

Tools

CASE tools supports specific tasks in the software development life-cycle. They can be divided into the following categories:

Business and Analysis modeling. Graphical modeling tools. E.g., E/R modeling, object modeling, etc.

Development. Design and construction phases of the life-cycle. Debugging environments. E.g., GNU Debugger.

Verification and validation. Analyze code and specifications for correctness, performance, etc.

Configuration management. Control the check-in and check-out of repository objects and files. E.g., SCCS, CMS.

Metrics and measurement. Analyze code for complexity, modularity (e.g., no "go to's"), performance, etc.

Project management. Manage project plans, task assignments, scheduling.

Another common way to distinguish CASE tools is the distinction between Upper CASE and Lower CASE. Upper CASE Tools support business and analysis modeling. They support traditional diagrammatic languages such as ER diagrams, Data flow diagram, Structure charts, Decision Trees, Decision tables, etc. Lower CASE Tools support development activities, such as physical design, debugging, construction, testing, component integration, maintenance, and reverse engineering. All other activities span the entire life-cycle and apply equally to upper and lower CASE.[9]

Workbenches[edit]

Workbenches integrate two or more CASE tools and support specific software-process activities. Hence they achieve:

a homogeneous and consistent interface (presentation integration).

seamless integration of tools and tool chains (control and data integration).

An example workbench is Microsoft's Visual Basic programming environment. It incorporates several development tools: a GUI builder, smart code editor, debugger, etc. Most commercial CASE products tended to be such workbenches that seamlessly integrated two or more tools. Workbenches also can be classified in the same manner as tools; as focusing on Analysis, Development, Verification, etc. as well as being focused on upper case, lower case or processes such as configuration management that span the complete life-cycle.

Environments[edit]

An environment is a collection of CASE tools or workbenches that attempts to support the complete software process. This contrasts with tools that focus on one specific task or a specific part of the life-cycle. CASE environments are classified by Fuggetta as follows:[8]

Toolkits. Loosely coupled collections of tools. These typically build on operating system workbenches such as the Unix Programmer's Workbench or the VMS VAX set. They

typically perform integration via piping or some other basic mechanism to share data and pass control. The strength of easy integration is also one of the drawbacks. Simple passing of parameters via technologies such as shell scripting can't provide the kind of sophisticated integration that a common repository database can.

Fourth generation. These environments are also known as 4GL standing for fourth generation language environments due to the fact that the early environments were designed around specific languages such as Visual Basic. They were the first environments to provide deep integration of multiple tools. Typically these environments were focused on specific types of applications. For example, user-interface driven applications that did standard atomic transactions to a relational database. Examples are Informix 4GL, and Focus.

Language-centered. Environments based on a single often object-oriented language such as the Symbolics Lisp Genera environment or VisualWorks Smalltalk from Parcplace. In these environments all the operating system resources were objects in the object-oriented language. This provides powerful debugging and graphical opportunities but the code developed is mostly limited to the specific language. For this reason, these environments were mostly a niche within CASE. Their use was mostly for prototyping and R&D projects. A common core idea for these environments was the model-view-controller user interface that facilitated keeping multiple presentations of the same design consistent with the underlying model. The MVC architecture was adopted by the other types of CASE environments as well as many of the applications that were built with them.

Integrated. These environments are an example of what most IT people tend to think of first when they think of CASE. Environments such as IBM's AD/Cycle, Andersen Consulting's FOUNDATION, the ICL CADES system, and DEC Cohesion. These environments attempt to cover the complete life-cycle from analysis to maintenance and provide an integrated database repository for storing all artifacts of the software process. The integrated software repository was the defining feature for these kinds of tools. They provided multiple different design models as well as support for code in heterogeneous languages. One of the main goals for these types of environments was "round trip engineering": being able to make changes at the design level and have those automatically be reflected in the code and vice versa. These environments were also typically associated with a particular methodology for software development. For example, the FOUNDATION CASE suite from Andersen was closely tied to the Andersen Method/1 methodology.

Process-centered. This is the most ambitious type of integration. These environments attempt to not just formally specify the analysis and design objects of the software process but the actual process itself and to use that formal process to control and guide software projects. Examples are East, Enterprise II, Process Wise, Process Weaver, and Arcadia.

19.

Project Scheduling and Tracking – Basic Concepts

Project Scheduling helps to establish a roadmap for project managers together with estimation methods and risk analysis. Project scheduling and Tracking begins with the identification of process models, identification of software tasks and activities, estimation of effort and work and ends with creation of network of tasks and making sure it gets done on time. This network is adapted on encountering of changes and risks.

At the project level, the Project Manager does project tracking and scheduling based on information received from Software Engineers. At an individual level the Software Engineer does it. It is important, as in a complex system many tasks may occur in parallel and have interdependencies that are understandable only with a schedule. A detailed schedule is a useful tool to assess progress on a moderate or large project.

The basic steps followed are, once the tasks dictated by the software process model is refined based on the functionality of the system , effort and duration are allocated for each task and an activity network is created that allows the project to meets its deadlines. The work product of this activity is the project schedule and in order that it is accurate it is required to check all tasks are covered in the activity network, effort and timing are appropriately allocated, interdependencies are correctly indicated, resources are allocated tasks in a right manner and closely spaced milestones are defined to track the project easily.

One of the major challenges in software project management is the difficulty to adhere to schedules. The common reasons for a late delivery of software project are an unrealistic deadline, changing customer requirements, honest underestimate of effort or resources, overlooked risks, unforeseen technical difficulties or human difficulties, miscommunication and failure by project manager to recognize the delay early and take appropriate measures.

Software project scheduling is an activity that distributes estimated effort across the duration of project cycle by allocating effort to each specific task that is associated with all process. The basic principles that guides software project scheduling is compartmentalization of the project into a number of manageable tasks, correct allocation of time, correct effort validation ,defining responsibility for each task to a team member, defining outcomes or work product for each task and defining milestones for a task or group of tasks as appropriate.

A Task set is a collection of software tasks, milestones and deliveries that must be completed for the project to be successfully accomplished. Task sets are defined for being applicable to different type of project and degree of rigor. The types of projects commonly encountered are Concept Development projects, New applications, Development projects, Application enhancement projects, Application maintenance projects and Re-engineering projects. The degree of rigor with which the software process is applied may be casual, structured, strict or quick reaction (used for emergency situation). For the project manager to develop a systematic approach for selecting degree of rigor for the type of project project adaptation criteria are defined and a task set selector value is computed based on the characteristics of the project.

Program evaluation and review technique (PERT) and critical path method (CPM) are two of the commonly used project scheduling methods. These techniques are driven by information

such as estimates of effort, decomposition of the product function, the selection of process model, task set and decomposition of tasks. The interdependencies among tasks are defined using a task network. A task network or activity network is a graphic representation of the task flows for a project. According to basic PERT, expected task duration is calculated as the weighted average of the most pessimistic, the most optimistic and most probable time estimates. The expected duration of any path on the network is found by summing the expected durations of tasks. PERT gives appropriate results when there is a single dominant path in the network. The time needed to complete the project is defined by the longest path in the network which is called critical path. CPM allows software planner to determine the critical path and establish most likely time estimates.

While creating schedule a timeline chart also called as Gantt chart can be generated. This can be developed for entire project or separately for each function or individual.

The information necessary for generation of this is Work Breakdown Structure (WBS – Making a complex project manageable by breaking it into individual components in a hierarchical structure that defines independent tasks), effort, duration and start date and details of assignment of tasks to resources. Along with this most software project scheduling tools produce project tables which is a tabular listing of project tasks, their planned and actual start, end dates and other information. This with timeline chart is valuable to project managers to track the progress.

Tracking the project schedule can be done by conducting periodic project status meeting, evaluating result of reviews conducted at all stages of development life cycle, determining the completion of defined project milestones, comparison of actual and planned dates, using earned value analysis technique for performing quantitative analysis of program.

Error tracking methods can also be used for assessing the status of current project. This is done by collecting error related measures and metrics from past project and using this as baseline for comparison against real time data.

20. Process and Project Metrics

- Metrics should be collected so that process and product indicators can be ascertained
- *Process metrics* used to provide indicators that lead to long term process improvement
- *Project metrics* enable project manager to
 - Assess status of ongoing project
 - Track potential risks
 - Uncover problems before they go critical
 - Adjust work flow or tasks
 - Evaluate the project team's ability to control quality of software products

Process Metrics

- Private process metrics (e.g. defect rates by individual or module) are only known to by the individual or team concerned.
- Public process metrics enable organizations to make strategic changes to improve the software process.
- Metrics should not be used to evaluate the performance of individuals.
- Statistical software process improvement helps and organization to discover where they are strong and where are week.

Statistical Process Control

1. Errors are categorized by their origin
2. Record cost to correct each error and defect
3. Count number of errors and defects in each category
4. Overall cost of errors and defects computed for each category
5. Identiy category with greatest cost to organization
6. Develop plans to eliminate the most costly class of errors and defects or at least reduce their frequency

Project Metrics

- A software team can use software project metrics to adapt project workflow and technical activities.
- Project metrics are used to avoid development schedule delays, to mitigate potential risks, and to assess product quality on an on-going basis.
- Every project should measure its inputs (resources), outputs (deliverables), and results (effectiveness of deliverables).

Software Measurement

- *Direct process measures* include cost and effort.
- *Direct process measures* include lines of code (LOC), execution speed, memory size, defects reported over some time period.
- *Indirect product measures* examine the quality of the software product itself (e.g. functionality, complexity, efficiency, reliability, maintainability).

Size-Oriented Metrics

- Derived by normalizing (dividing) any direct measure (e.g. defects or human effort) associated with the product or project by LOC.
- Size oriented metrics are widely used but their validity and applicability is widely debated.

Function-Oriented Metrics

- Function points are computed from direct measures of the information domain of a business software application and assessment of its complexity.
- Once computed function points are used like LOC to normalize measures for software productivity, quality, and other attributes.
- The relationship of LOC and function points depends on the language used to implement the software.

Reconciling LOC and FP Metrics

- The relationship between lines of code and function points depends upon the programming language that is used to implement the software and the quality of the design
- Function points and LOC-based metrics have been found to be relatively accurate predictors of software development effort and cost
- Using LOC and FP for estimation a historical baseline of information must be established.

Object-Oriented Metrics

- Number of scenario scripts (NSS)
- Number of key classes (NKC)
- Number of support classes (e.g. UI classes, database access classes, computations classes, etc.)
- Average number of support classes per key class
- Number of subsystems (NSUB)

Use Case-Oriented Metrics

- Describe (indirectly) user-visible functions and features in language independent manner
- Number of use case is directly proportional to LOC size of application and number of test cases needed
- However use cases do not come in standard sizes and use as a normalization measure is suspect
- Use case points have been suggested as a mechanism for estimating effort

WebApp Project Metrics

- Number of static Web pages (N_{sp})
- Number of dynamic Web pages (N_{dp})
- Customization index: $C = N_{sp} / (N_{dp} + N_{sp})$
- Number of internal page links
- Number of persistent data objects
- Number of external systems interfaced
- Number of static content objects
- Number of dynamic content objects
- Number of executable functions

Software Quality Metrics

- Factors assessing software quality come from three distinct points of view (product operation, product revision, product modification).
- Software quality factors requiring measures include
 - correctness (defects per KLOC)
 - maintainability (mean time to change)
 - integrity (threat and security)
 - usability (easy to learn, easy to use, productivity increase, user attitude)
- Defect removal efficiency (DRE) is a measure of the filtering ability of the quality assurance and control activities as they are applied through out the process framework

$$\text{DRE} = E / (E + D)$$

E = number of errors found before delivery of work product

D = number of defects found after work product delivery

Integrating Metrics with Software Process

- Many software developers do not collect measures.
- Without measurement it is impossible to determine whether a process is improving or not.
- Baseline metrics data should be collected from a large, representative sampling of past software projects.
- Getting this historic project data is very difficult, if the previous developers did not collect data in an on-going manner.

Arguments for Software Metrics

- If you don't measure you have no way of determining any improvement
- By requesting and evaluating productivity and quality measures software teams can establish meaningful goals for process improvement
- Software project managers are concerned with developing project estimates, producing high quality systems, and delivering product on time
- Using measurement to establish a project baseline helps to make project managers tasks possible

Baselines

- Establishing a metrics baseline can benefit portions of the process, project, and product levels
- Baseline data must often be collected by historical investigation of past project (better to collect while projects are on-going)
- To be effective the baseline data needs to have the following attributes:
 - data must be reasonably accurate, not guesstimates
 - data should be collected for as many projects as possible
 - measures must be consistent

- applications should be similar to work that is to be estimated

Metrics for Small Organizations

- Most software organizations have fewer than 20 software engineers.
- Best advice is to choose simple metrics that provide value to the organization and don't require a lot of effort to collect.
- Even small groups can expect a significant return on the investment required to collect metrics, if this activity leads to process improvement.

Establishing a Software Metrics Program

1. identify business goal
2. Identify what you want to know
3. Identify subgoals
4. Identify subgoal entities and attributes
5. Formalize measurement goals
6. Identify quantifiable questions and indicators related to subgoals
7. Identify data elements needed to be collected to construct the indicators
8. Define measures to be used and create operational definitions for them
9. Identify actions needed to implement the measures
10. Prepare a plan to implement the measures

Reg. No._____

Name:_____

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
SIXTH SEMESTER B.TECH DEGREE MODEL EXAMINATION, APRIL 2018
Department: Computer Science and Engineering
Subject: - CS308: Software Engineering And Project Management

Max. Marks: 100

Duration:3Hours

PART A

(Answer All Questions Each carries 3 Marks)

1. What is software engineering? Is it an art, craft or a science? Discuss.
2. Write a short note on ISO 9000.
3. What do you understand by the term Software Development Life Cycle (SDLC)?
4. What is software requirements specification (SRS) ?

PART B

(Answer any TWO, Each carries 9 Marks)

5. a) What do you understand with the term “requirements elicitation” ? What are the use case guidelines. (6 marks)
- b) Explain why, for large software systems development, is it recommended that prototypes should be “throw-away” prototype ? (3 marks)
6. a) Discuss the prototyping model. What is the effect of designing a prototype on the overall cost of the project? (6 marks)
- b) What are the disadvantages of developing the Linear Sequential Model? (3 Marks)
7. a) Explain different levels of Capability Maturity Model. (3 Marks)
- b) Explain the spiral model of software development. What are the limitations of such a model? (6 Marks)

PART C

(Answer All Questions Each Carries 3 Marks)

8. Write a short note on data design.
9. An application has the following: 10 low external inputs, 12 high external outputs, 20 low internal logical files, 15 high external interface files, 12 average external inquiries, and a value of complexity adjustment factor of 1.10. What are the unadjusted and adjusted function point counts ?
10. Write a short note on CODING Stanards?
11. What is the purpose of integration testing?

PART D

(Answer any TWO, Each carries 9 Marks)

12. (a) What are the objectives of testing? (3 Marks)
(b) States importance of modularity and explain coupling and cohesion. (6 Marks)
13. (a) Write a short note on Basic COCOMO Model. (3 Marks)
(b) Suppose that a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three modes i.e., organic, semidetached and embedded. (6 Marks)
14. (a) Discuss the role of software testing during software life cycle and why is it so difficult? (3 Marks)
(b). Discuss various white box testing techniques. (3 Marks)
(c). What are the advantages and disadvantages of white box testing? (3 marks)

PART E

(Answer any FOUR, Each carries 10 Marks)

15. What is software maintenance? Describe various categories of maintenance. Why software requires maintenance?
16. (a) Describe the difference between known risks and predictable risks. (2 Marks)
(b) What is software risks. Discuss different categories of software risk. (5 marks)
(c) How staff turnover problem affects software projects? (3 Marks)
17. (a) Describe about scope of software product and problem decomposition. (5 marks)
(b) Explain various roles and responsibility of people in project management. (5 marks)

18. (a) What do you understand by software configuration? (2 marks)
- (b) Discuss the elements of a configuration management system. (4 marks)
- (c) Why is software configuration management crucial to the success of large software product development projects? (4 marks)
19. (a) What are CASE Tools? Explain building blocks for CASE. (8 marks)
- (b) List out any four CASE Tools. (2 marks)
20. (a) List out and explain about the basic principles of project scheduling. (7 marks)
- (b) List out the factors Influencing Task Set Selection in Projects. (3 marks)

SIXTH SEMESTER B.TECH DEGREE MODEL EXAMINATION, APRIL 2018

Department: Computer Science and Engineering

Subject: - CS308: Software Engineering And Project Management

Max. Marks: 100

Duration:3Hours

ANSWER KEY

PART A

1. Software engineering is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product. The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software.

Definition-----1 mark

It is a **science** as we try hard to follow formal approaches where feasible. It is a **craft** because solving problems often requires experience and best practices. And it is an **art** as **software engineering** leaves enough freedom for intuitive and even artistic solutions.

Explanation-----2 marks

2. ISO 9000(International Standards Organization)

- Published in 1987.
- Contract between independent parties.
- Specifies the guide lines for maintaining a quality system.
- ISO 9000 Series of 3 stds: ISO 9001, ISO 9002, ISO 9003.

Goals.....

- 1.Meet stakeholder needs
2. Be usable by all sizes of organizations
3. Be usable by all sectors
4. Be simple and clearly understood
- 5.Connect quality management system to business processes

Stages for getting ISO 9000

- **Application**
- **Pre assessment**
- **Document Review**

- **Compliance Audit**
- **Registration**
- **Continued Surveillance**

Definition-----Explanation-----3 marks

3. There are various software development approaches defined and designed which are used-employed during development process of software, these approaches are also referred as “Software Development Process Models” (e.g. Waterfall model, incremental model, V-model, iterative model, RAD model, Agile model, Spiral model, Prototype model etc.). Each process model follows a particular life cycle in order to ensure success in process of software development.

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced according to the design which is called development phase. After coding and development the testing verifies the deliverable of the implementation phase against requirements. The testing team follows Software Testing Life Cycle (STLC) which is similar to the development cycle followed by the development team.

There are following six phases in every Software development life cycle model:

1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Deployment
6. Maintenance

Definition-----1.5 marks

Phases-----1.5 marks

4. A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

Qualities of SRS:

- Correct
- Unambiguous

- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable
- Traceable

The output of the requirements phase of the software development process is Software Requirements Specification (SRS) (also known as requirements document). This document lays a foundation for software engineering activities and is created when entire requirements are elicited and analyzed. SRS is a formal document, which acts as a representation of software that enables the users to review whether it (SRS) is according to their requirements. In addition, it includes user requirements for a system as well as detailed specifications of the system requirements.

Definition-----2 marks

Characteristics-----1 mark

PART B

5. (a) Requirements elicitation-----4 marks

- Requirements elicitation (also called requirements gathering) combines elements of problem solving, elaboration, negotiation, and specification.
- In order to encourage a collaborative, team-oriented approach to requirements gathering, stakeholders work together to identify the problem, propose elements of the solution, negotiate different approaches and specify a preliminary set of solution requirements.
- Many different approaches to collaborative requirements gathering have been proposed. Each makes use of a slightly different scenario, but all apply some variation on the following basic guidelines:
 - Meetings are conducted and attended by both software engineers and other stakeholders.
 - Rules for preparation and participation are established.
 - An agenda is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas.
 - A “facilitator” (can be a customer, a developer, or an outsider) controls the meeting.
 - A “definition mechanism” (can be work sheets, flip charts, or wall stickers or an electronic bulletin board, chat room, or virtual forum) is used.
 - Quality Function Deployment(QFD) is a quality management technique that translates the needs of the customer into technical requirements for software.

- QFD “concentrates on maximizing customer satisfaction from the software engineering process”.
- To accomplish this, QFD emphasizes an understanding of what is valuable to the customer and then deploys these values throughout the engineering process. QFD identifies three types of requirements: Normal Requirements, Expected Requirements and Exciting Requirements.

Use Case guidelines-----2 marks

A use case is initiated by a user with a particular goal in mind, and completes successfully when that goal is satisfied.

1. Identify all users
2. Create a user profile for each category of users including all roles of the users play that are relevant to the system.
3. Create a use case for each goal, following the use case template maintain the same level of abstraction throughout the use case. Steps in higher level use cases may be treated as goals for lower level (i.e. more detailed), sub use cases.
4. Structure the use case
5. Review and validate with users.

4+2=6marks

(b)

Throw-away prototyping is about creating, as fast as possible, a part of the future application to either ensure a feature is technically feasible or to show the feature to stakeholders or potential users in order to gather feedback from them.

Since the source code of this prototype is not *reused* later when developing the application itself, it makes it a throw-away prototype. Knowing that it's a throw-away code helps focusing on the actual feature, while leaving aside aspects such as maintainability of the code, style, design patterns or testing. This makes it possible to finish the prototype very fast, without affecting negatively the technical debt of the final product.

Throw-away prototyping is different from sketching. Sketching is more graphical and oriented towards user interfaces and user experience, and doesn't consist of writing programming code. Throw-away prototyping is usually used when sketching is not enough (for example when you need to show how a feature will perform on different smart phones or when you need to show the actual performance and responsiveness).

Throw-away prototyping can present a risk when dealing with stakeholders without technical background and in a context of tight deadlines and very limited resources: stakeholders may try to convince you to reuse in the final product the source code from the prototype

Throw away prototype-----advantages-----3 marks

6. (a)

- a customer defines a set of general objectives for software but does not identify detailed input, processing, or output requirements.
- In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human/machine interaction should take.
- In these, and many other situations, a prototyping paradigm may offer the best approach.
- The prototyping paradigm begins with requirements gathering.
- Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory.
- A "quick design" then occurs. The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user (e.g., input approaches and output formats).
- The quick design leads to the construction of a prototype.
- The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed.
- Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done
- Ideally, the prototype serves as a mechanism for identifying software requirements.
- If a working prototype is built, the developer attempts to use existing program fragments or applies tools (e.g., report generators, window managers) that enable working programs to be generated quickly
- The prototype can serve as "the first system."

If you prototype correctly, it should lower your costs quite a bit. It means that you build it fast and cheap. It shouldn't take you months to build a prototype and it shouldn't cost you thousands of dollars. You should make your prototypes a very basic version of your idea, and then test that version. Listen to what your customers say and iterate on that prototype. Once you have a pretty good idea of what they want and what does and does not work then it's time to build the "perfect" version.

By testing it out cheaply you're working out kinks early on in the process when it's the cheapest to solve the problems, and before they become too big to solve.

Prototyping Model-----4 marks

Effects on cost-----2 marks

(b) The linear sequential model is the oldest and the most widely used paradigm for software engineering. Among the problems that are sometimes encountered when the linear sequential model is applied are:

1. Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.

2. It is often difficult for the customer to state all requirements explicitly. The linear sequential model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.
3. The customer must have patience. A working version of the program(s) will not be available until late in the project time-span. A major blunder, if undetected until the working program is reviewed, can be disastrous.

3 marks

- 7. (a)** The Software Engineering Institute (SEI) Capability Maturity Model (CMM) specifies an increasing series of levels of a software development organization. The higher the level, the better the software development process, hence reaching each level is an expensive and time-consuming process.

Definition- 1/2 MARK

There are 5 levels-Initial, repeatable, defined, managed and optimizing. (5*1/2=2.5MARKS)

Initial: Success of the organization majorly depends on an individual effort, talent, and heroics.

Repeatable: The process is in place to repeat the earlier successes on projects with similar applications. Program management is a key characteristic of a level two organization.

Defined: Company has pulled together a standard set of processes and controls for the entire organization so that developers can move between projects more easily and customers can begin to get consistency from different groups.

Managed: In addition to implementing standard processes, company has installed systems to measure the quality of those processes across all projects.

Optimizing: Company has accomplished all of the above and can now begin to see patterns in performance over time, so it can tweak its processes in order to improve productivity and reduce defects in software development across the entire organization.

0.5+2.5=3 marks

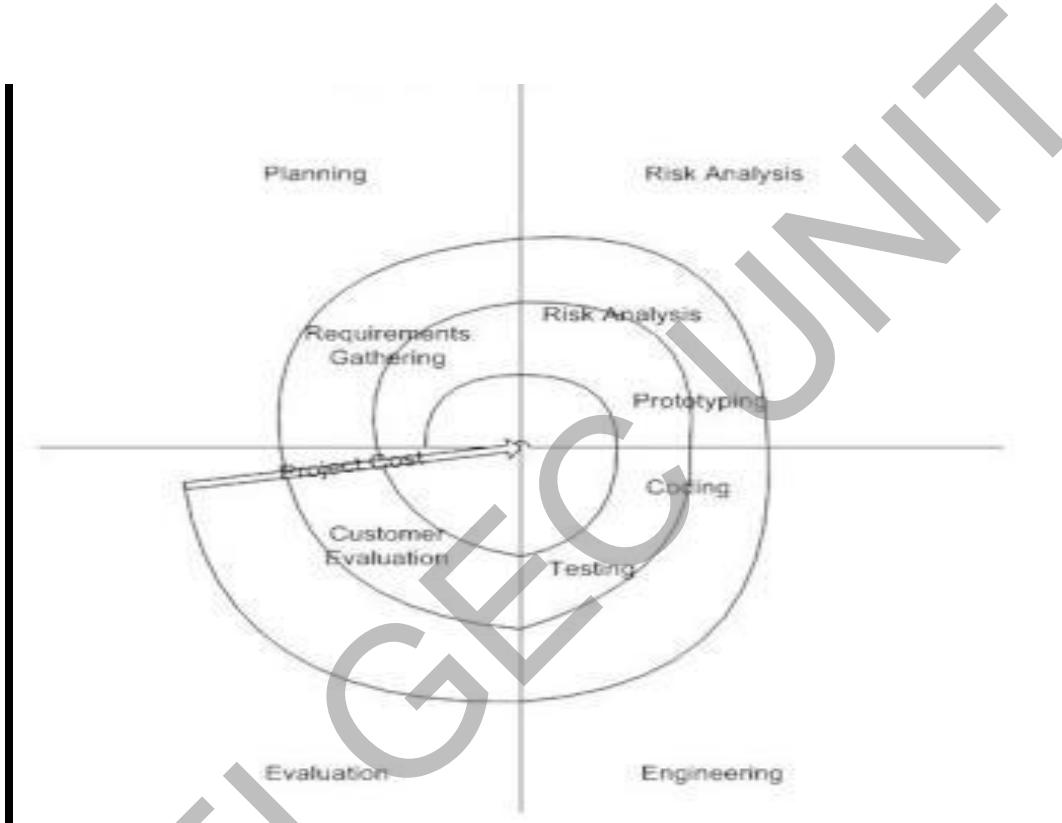
- (b)** The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spirals builds on the baseline spiral. Its one of the software development models like Waterfall, Agile, V-Model.

Planning Phase: Requirements are gathered during the planning phase. Requirements like ‘BRS’ that is ‘Business Requirement Specifications’ and ‘SRS’ that is ‘System Requirement specifications’.

Risk Analysis: In the **risk analysis phase**, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

Engineering Phase: In this phase software is **developed**, along with **testing** at the end of the phase. Hence in this phase the development and testing is done.

Evaluation phase: This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.



Limitations:

1. Can be a costly model to use.
2. Risk analysis requires highly specific expertise.
3. Project's success is highly dependent on the risk analysis phase.
4. Doesn't work well for smaller projects.

Definition and phases-----2.5 marks

Figure-----1.5 marks

Limitations-----2 marks

2.5+1.5+2+3=9marks

PART C

- 8.** The design task produces a data design, an architectural design, an interface design, and a component design. The importance of software design can be stated with a single word—quality. Design is the only way that we can accurately translate a customer's requirements into a finished software product or system.

Design-----Definition-----1 mark

Data Design-----2 marks

The data design transforms the information domain model created during analysis into the data structures that will be required to implement the software. The data objects and relationships defined in the entity relationship diagram and the detailed data content depicted in the data dictionary provide the basis for the data design activity

1+2=3marks

- 9. $FP = UFP * CAF$**

$$\begin{aligned}UFP &= 10 \times 3 + 12 \times 7 + 20 \times 7 + 15 + 10 + 12 \times 4 \\&= 30 + 84 + 140 + 150 + 48 \\&= 452\end{aligned}$$

CAF=1.10

$FP = 452 \times 1.10 = 497.2$ (3Marks)

- 10.** Good software development organizations usually develop their own coding standards and guidelines depending on what best suits their organization and the type of products they develop.

The following are some representative coding standards.

Rules for limiting the use of global: These rules list what types of data can be declared global and what cannot.

Naming conventions for global variables, local variables, and constant identifiers: A possible naming convention can be that global variable names always start with a capital letter, local variable names are made of small letters, and constant names are always capital letters.

Error return conventions and exception handling mechanisms: The way error conditions are reported by different functions in a program are handled should be standard within an organization. For example, different functions while encountering an error condition should either return a 0 or 1 consistently.

The following are some representative coding guidelines recommended by many software development organizations.

Do not use a coding style that is too clever or too difficult to understand: Code should be easy to understand. Many inexperienced engineers actually take pride in writing cryptic and incomprehensible code. Clever coding can obscure meaning of the code and hamper understanding. It also makes maintenance difficult.

Any 3 coding standard with explanation-----3*1=3 marks

11. Integration testing ("I&T") is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Definition-----1 mark

The purpose of integration testing is to determine that each independent module is correctly implemented. This gives little chance to determine that the interface between modules is also correct, and for this reason integration testing must be performed. One specific target of integration testing is the interface: whether parameters match on both sides as to type, permissible ranges, meaning and utilization. Integration testing is to verify the functional, performance, and reliability between the modules that are integrated. It is used to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

Purpose-----2 marks

PART D

12. (a) Testing Objectives:

- **Finding defects** which may get created by the programmer while developing the software.
- Gaining confidence in and providing information about the level of **quality**.
- To prevent defects.
- To make sure that the end result meets the business and user requirements.
- To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
- To gain the confidence of the customers by providing them a quality product.

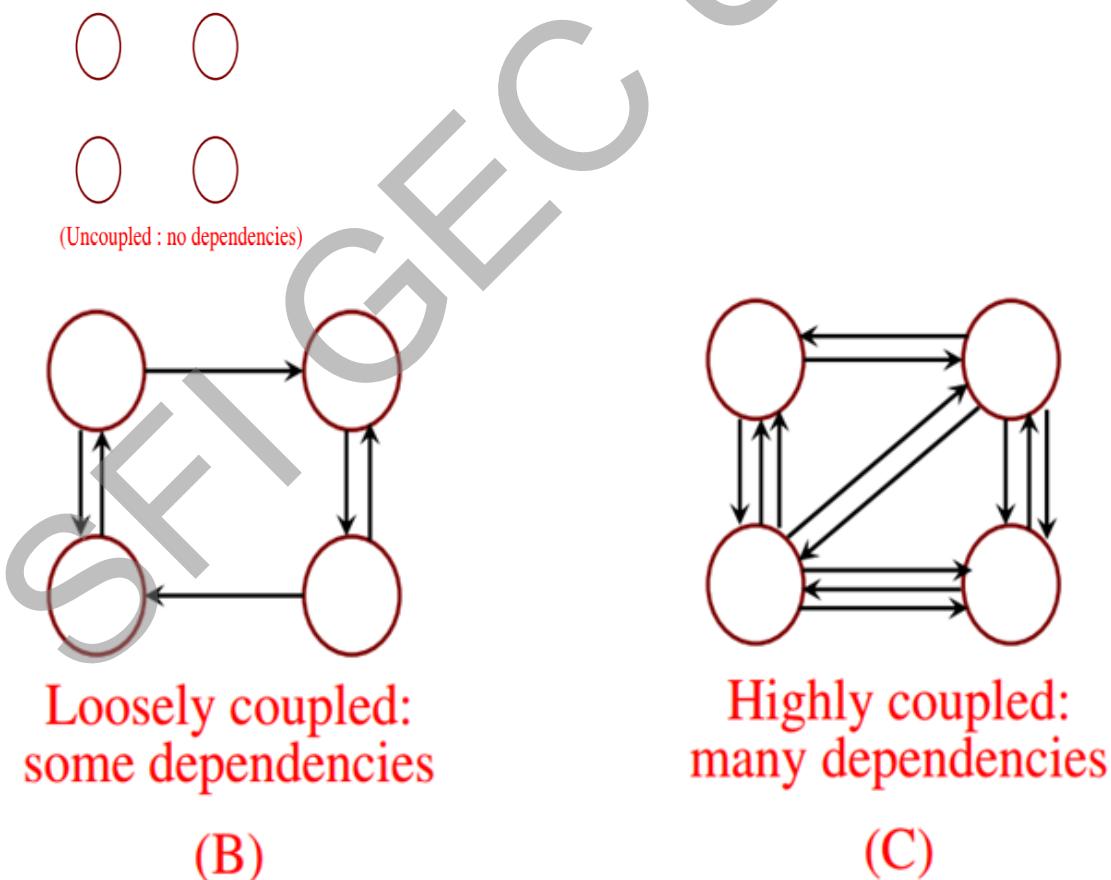
Any 6 objectives-----3 marks

(b) Modularity-----2 marks

- Software architecture embodies modularity; that is, software is divided into separately named and addressable components, often called modules, that are integrated to satisfy problem requirements.
- defines five criteria that enable us to evaluate a design method with respect to its ability to define an effective modular system:
 1. Modular decomposability.
 2. Modular composability.
 3. Modular understandability.
 4. Modular continuity
 5. Modular protection

Coupling-----2marks

Coupling is a measure of the relative interdependence among modules.

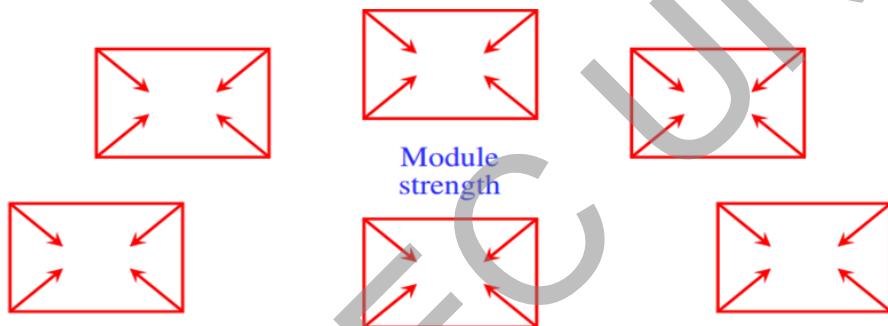


There are different types of module coupling:

1. Data coupling
2. Stamp coupling
3. control coupling
4. external coupling
5. common coupling
6. content coupling

Cohesion-----2 marks

Cohesion is a measure of the degree to which the elements of a module are functionally related.



Cohesion=Strength of relations within the module

Types of cohesion:

1. Functional cohesion
2. Sequential cohesion
3. Procedural cohesion
4. Temporal cohesion
5. Logical cohesion
6. Coincident cohesion

2+2+2=6 marks

13. (a) Basic COCOMO

- COCOMO model (Constructive cost model) was proposed by Boehm.
- This model estimates the total effort in terms of “person-months” of the technical project staff.
- Boehm introduces three forms of cocomo.
- It can be applied in three classes of software project: Organic mode, Semi detached mode and Embedded mode.
- 3 forms of COCOMO Model: Basic, Intermediate and Complete COCOMO
- Basic COCOMO: Computes software development effort and cost as a function of programme size expressed in terms of lines of code(LOC).
- **Effort $E = ab(KLOC)^{bb}$ persons-months**
- **Development Time $D = cb(E)^{db}$ months**
- a, b, c, d are the co-efficients for the three modes are given below:
- From E & D we can compute the no: of people required to accomplish the project as $N = E/D$
- Merits of Basic Cocomo model: Basic cocomo model is good for quick, early, rough order of magnitude estimates of software project.
- Limitations :
 - 1.The accuracy of this model is limited because it does not consider certain factors for cost estimation of software. These factors are hardware constraints, personal quality and experiences, modern techniques and tools.
 2. The estimates of Cocomo model are within a factor of 1.3 only 29% of the time and within the factor of 2 only 60% of time.

Definition-----2 marks

Merits and demerits-----1 mark

(b) The basic COCOMO equation take the form:

$$E = a (KLOC)^b$$

$$D = c (E)^d$$

Estimated size of the project = 400 KLOC

(i) Organic mode **(2 Marks)**

$$E = 2.4(400)^{1.05} = 1295.31 \text{ PM}$$

$$D = 2.5(1295.31)^{0.38} = 38.07 \text{ M}$$

(ii) Semidetached mode **(2 Marks)**

$$E = 3.0(400)^{1.12} = 2462.79 \text{ PM}$$

$$D = 2.5(2462.79)^{0.35} = 38.45 \text{ M}$$

(iii) Embedded mode **(2 Marks)**

$$E = 3.6(400)^{1.20} = 4772.81 \text{ PM}$$

$$D = 2.5(4772.8)^{0.32} = 38 \text{ M}$$

2+2=6 marks

3+6=9 marks

14. (a) software testing is considered as one phase of the software development life cycle. Software Testing is the process of finding bugs in the software & make the software bug free. In the software development life cycle(SDLC) the Testing is plays an importance role, which helps to improve the quality, reliability & performance of the system with all check what all functions software supposed to do & also check that Software is not doing what he not supposed to do.

Software Testing is essential for the betterment of the application and software. Software Testing is as difficult as developing software because now days it requires more human efforts, as everyone wants quality so doing Software Testing are become difficult because it takes lot of time and money.

Definition-----1 mark

Explanation---2 marks

1+2=3 marks

(b) White Box Testing-----1 mark

White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

White Box Testing Techniques:-----2 marks

- **Statement Coverage** - This technique is aimed at exercising all programming statements with minimal tests.
- **Branch Coverage** - This technique is running a series of tests to ensure that all branches are tested at least once.
- **Path Coverage** - This technique corresponds to testing all possible paths which means that each statement and branch is covered.

1+2=3 marks

(c)Advantages-----1.5 marks

Disadvantages----1.5 marks

Advantages of White Box Testing:

- Forces test developer to reason carefully about implementation.
- Reveals errors in "hidden" code.
- Spots the Dead Code or other issues with respect to best programming practices.

Disadvantages of White Box Testing:

- Expensive as one has to spend both time and money to perform white box testing.
- Every possibility that few lines of code are missed accidentally.
- In-depth knowledge about the programming language is necessary to perform white box testing.

1.5+1.5=3 marks

3+3+3=9 marks

PART E

15) Software Maintenance-----2 marks

Software maintenance is widely accepted part of SDLC now a days. It stands for all the modifications and updations done after the delivery of software product. There are number of reasons, why modifications are required, some of them are briefly mentioned below:

- Market Conditions - Policies, which changes over the time, such as taxation and newly introduced constraints like, how to maintain bookkeeping, may trigger need for modification.
- Client Requirements - Over the time, customer may ask for new features or functions in the software.
- Host Modifications - If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are needed to keep adaptability.
- Organization Changes - If there is any business level change at client end, such as reduction of organization strength, acquiring another company, organization venturing into new business, need to modify in the original software may arise.

Categories of maintenance-----4 marks

Software Maintenance is a very broad activity that includes error corrections, enhancements of capabilities, deletion of obsolete capabilities, and optimization. In a software lifetime, type of maintenance may vary based on its nature. It may be just a routine maintenance tasks as some bug discovered by some user or it may be a large event in itself based on maintenance size or nature. Following are some types of maintenance based on their characteristics:

- **Corrective Maintenance** - This includes modifications and updatations done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.
- **Adaptive Maintenance** - This includes modifications and updatations applied to keep the software product up-to date and tuned to the ever changing world of technology and business environment.
- **Perfective Maintenance** - This includes modifications and updates done in order to keep the software usable over long period of time. It includes new features, new user requirements for refining the software and improve its reliability and performance.
- **Preventive Maintenance** - This includes modifications and updatations to prevent future problems of the software. It aims to attend problems, which are not significant at this moment but may cause serious issues in future.

Why maintenance-----4 marks

1. Bug Fixing

In maintenance management, bug fixing comes at priority to run the software seamlessly. This process contains search out for errors in code and correct them. The issues can be occurred in hardware, operating systems or any part of software. This must be done without hurting rest of the functionalities of existing software.

2. Capability Enhancement

This comprises improvement in features and functions to make solution compatible with varying market environment. It enhances software platforms, work pattern, hardware upgrade, compilers and all other aspects that affect system workflow. Boost your business using a technically updated solution applying software maintenance services regularly.

3. Removal of Outdated Functions

The unwanted functionalities are useless. Moreover, by occupying space in solution, they hurt efficiency of the solution. Using software maintenance procedure, such elements of UI and coding are removed and replaced with new development using the latest tools and technologies. This elimination makes the system adaptive to cope with changing circumstances.

4. Performance Improvement

To improve system performance, developers detect issues through testing and resolve them. Data and coding restricting as well as reengineering are the part of software maintenance. It prevents the solution from vulnerabilities. This is not any functionality that performs in operations, but it develops to stop harmful activities like hacking.

Thus, software maintenance services keep the solution hale and hearty. The experienced developers offer reliable and authenticated maintenance management applying modern technologies.

16.(a) Known Risks :- • That can be uncovered after careful evaluation of the project plan, the business, and technical environment in which the product is being developed

Example : Unrealistic delivery rate

(1 Mark)

Predictable Risks :- • Extrapolated from past project experience

Example : Staff turnover

(1 Mark)

(b) Software Risk-----2 marks

Risk is an expectation of loss, a potential problem that may or may not occur in the future. It is generally caused due to lack of information, control or time. A possibility of suffering from loss in software development process is called a software risk. Risk always involves two characteristics

Uncertainty and loss: Loss can be anything, increase in production cost, development of poor quality software, not being able to complete the project on time. Software risk exists because the future is uncertain and there are many known and unknown things that cannot be incorporated in the project plan.

Categories of Risks----- 3marks

1. Project Risks: Risks which will affect the project schedule or resources. For example, staff turnover, that is an experience team member of a project left the organization. **1 mark**

2. Product Risks: Risks that affect the quality or performance of the software being developed. For example a component isn't performing as expected. **1 mark**

3. Business Risks: Risks that affect the organization developing or procuring the software. For example, a competitor is developing a similar product that will challenge the product being developed.

1 mark

(c) Staff turnover refers to the number or percentage of workers who leave an organization and are replaced by new employees. Measuring staff turnover can be helpful to employers that want to examine reasons for turnover or estimate the cost-to-hire for budget purposes.

Staff turnover may lead to affect the ,

Productivity Rate

Service Delivery

Spread of organizational knowledge

Maintenance of the project

Staff turnover----Definition----1 mark

Problems with explanation-----2 marks

$$2+2+3+3=10$$

17.(a) Scope-----2 marks

The first software project management activity is the determination of software scope. Scope is defined by answering the following questions:

Context. How does the software to be built fit into a larger system, product, or business context, and what constraints are imposed as a result of the context?

Information objectives. What customer-visible data objects are produced as output from the software? What data objects are required for input?

Function and performance. What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed? Software project scope must be unambiguous and understandable at the management and technical levels. A statement of software scope must be bounded. That is, quantitative data (e.g., number of simultaneous users, size of mailing list, maximum allowable response time) are stated explicitly, constraints and/or limitations (e.g., product cost restricts memory size) are noted, and mitigating factors (e.g., desired algorithms are well understood and available in Java) are described.

Problem Decomposition-----3 marks

Problem decomposition, sometimes called *partitioning* or *problem elaboration*, is an activity that sits at the core of software requirements analysis.

During the scoping activity no attempt is made to fully decompose the problem. Rather, decomposition is applied in two major areas: (1) the functionality and content (information) that must be delivered and (2) the process that will be used to deliver it.

Human beings tend to apply a divide-and-conquer strategy when they are confronted with a complex problem. Stated simply, a complex problem is partitioned into smaller problems that are more manageable. This is the strategy that applies as project planning begins. Software functions, described in the statement of scope, are evaluated and refined to provide more detail prior to the beginning of estimation. Because both cost and schedule estimates are functionally oriented, some degree of decomposition is often useful. Similarly, major content or data objects are decomposed into their constituent parts, providing a reasonable understanding of the information to be produced by the software.

2+3=5marks

(b) The Stakeholders-----2 marks

The software process (and every software project) is populated by stakeholders who can be categorized into one of five constituencies:

- 1.** Senior managers who define the business issues that often have a significant influence on the project.
- 2.** Project (technical) managers who must plan, motivate, organize, and control the practitioners who do software work.
- 3.** Practitioners who deliver the technical skills that are necessary to engineer a product or application.
- 4.** Customers who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
- 5.** End users who interact with the software once it is released for production use.

Team Leaders-----1mark

Project management is a people-intensive activity, and for this reason, competent practitioners often make poor team leaders.

Characteristics:

Motivation. The ability to encourage (by “push or pull”) technical people to produce to their best ability.

Organization. The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.

Ideas or innovation. The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

Software Team-----1 mark

The “best” team structure depends on the management style of your organization, the number of people who will populate the team and their skill levels, and the overall problem difficulty.

Agile Teams-----1 mark

Many software organizations advocate agile software development as an antidote to many of the problems that have plagued software project work. To review, the agile philosophy encourages

customer satisfaction and early incremental delivery of software, small highly motivated project teams, informal methods, minimal software engineering work products, and overall development simplicity.

18.(a) The results or deliverables of a large software development effort typically consist of a large number of objects, e.g. source code, design document, SRS document, test document, user's manual, etc. These objects are usually referred to and modified by a number of software engineers throughout the life cycle of the software. The state of all these objects at any point of time is called the configuration of the software product. The state of each deliverable object changes as development progresses and also as bugs are detected and fixed.

(2 marks)

(b) Component elements – set of tools coupled within a file management system to enable access to and management of each SCI(Software Configuration Items)

Process elements – collection of procedures and tasks that define an effective approach to change management for all stakeholders

Construction elements – set of tools that automate the construction of software by ensuring a set of validated components is assembled

Human elements – team uses a set of tools and process features encompassing other CM elements

Each Element with explanation carries 1 mark

4*1=4 marks

(c) Software project managers pay attention to the planning and execution of configuration management, an integral task, because it facilitates the ability to communicate status of documents and code as well as changes that have been made to them. High-quality released software has been tested and used, making it a reusable asset and saving development costs. Reused components aren't free, though—they require integration into new products, a difficult task without knowing exactly what they are and where they are.

CM enhances the ability to provide maintenance support necessary once the software is deployed. If software didn't change, maintenance wouldn't exist. Of course, changes do occur. The National Institute of Standards and Technology (NIST) says that software will be changed to adapt, perfect, or correct it. Pressman points out that new business, new customer needs, reorganizations, and budgetary or scheduling constraints may lead to software revision.

CM works for the project and the organization in other ways as well. It helps to eliminate confusion, chaos, double maintenance, the shared data problem, and the simultaneous update problem

Importance of SCM and its explanations-----4 Marks

2+4+4=10 Marks

19. (a)

CASE Tools

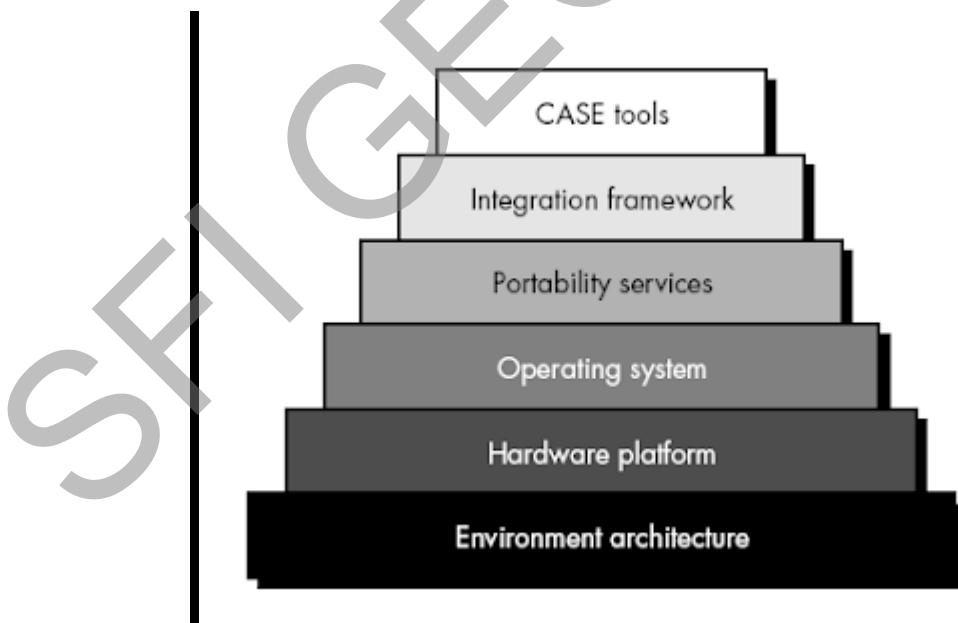
CASE stands for Computer Aided Software Engineering. It means, development and maintenance of software projects with help of various automated software tools.

CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.

There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few. Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

Definition-----2 marks

Building Blocks



Computer aided software engineering can be as simple as a single tool that supports a specific software engineering activity or as complex as a complete "environment" that encompasses tools, a database, people, hardware, a network, operating systems, standards, and myriad other

components. The building blocks for CASE are illustrated in figure. Each building block forms a foundation for the next, with tools sitting at the top of the heap. It is interesting to note that the foundation for effective CASE environments has relatively little to do with software engineering tools themselves. Rather, successful environments for software engineering are built on an environment architecture that encompasses appropriate hardware and systems software. In addition, the environment architecture must consider the human work patterns that are applied during the software engineering process.

The environment architecture, composed of the hardware platform and system support (including networking software, database management, and object management services), lays the ground work for CASE. But the CASE environment itself demands other building blocks. A set of portability services provides a bridge between CASE tools and their integration framework and the environment architecture. The integration framework is a collection of specialized programs that enables individual CASE tools to communicate with one another, to create a project database, and to exhibit the same look and feel to the end-user (the software engineer). Portability services allow CASE tools and their integration framework to migrate across different hardware platforms and operating systems without significant adaptive maintenance.

The building blocks depicted in figure above represent a comprehensive foundation for the integration of CASE tools. However, most CASE tools in use today have not been constructed using all these building blocks. In fact, some CASE tools remain "point solutions." That is, a tool is used to assist in a particular software engineering activity (e.g., analysis modeling) but does not directly communicate with other tools, is not tied into a project database, is not part of an integrated CASE environment (ICASE). Although this situation is not ideal, a CASE tool can be used quite effectively, even if it is a point solution.

Figure-----1 mark

Explanation-----5 marks

$$2+1+5=8$$

b) CASE TOOLS

- Business planning and modeling.
- Analysis and design.
- User-interface development.
- Programming.
- Verification and validation.
- Maintenance and reverse engineering.
- Configuration management.

- Project management.

Any four tools-----2 marks

20.(a) **Software project scheduling** can be defined as an activity that distributes the estimated effort across the planned project duration by allocating the effort to specific software engineering tasks. Simply one can say that project schedule is a tool which communicates

1. What works has to be performed.
2. Who will perform the work.
3. Time duration within which that work needs to be completed.

There are seven principles of software project scheduling :

Compartmentalization :

A given software project is compartmentalized into a number of manageable activities. The project is divided into a number of small tasks.

Interdependency :

Interdependent tasks are accomplished first. Certain tasks occur in sequence whereas other tasks occur in parallel. Therefore tasks which occur in sequence have to be performed in a sequential order since the output of one task will be the input of the next task. Other tasks can occur independently.

Time allocation :

Each and every task has to be assigned a specific time period i.e a start date and a completion date based on whether the work will be performed in a full time or part time basis.

Effort validation :

Every project is assigned to a software team. The project manager has to make sure that the effort allocated should not be more than the number of people available to do the work.

Defined responsibilities :

Each of the scheduled task is assigned to a specific member of the software team.

Defined outcomes :

Each task has a defined outcome. Work product is the outcome of a software project.

Defined milestones :

Every task is associated with a milestone. A milestone is an action or event marking a significant change in development process.

These are the seven basic principles that guide software project scheduling.

Each principle carries 1 mark

7*1=7marks

(b)

1. Size of project
2. Number of potential users
3. Mission criticality
4. Application longevity
5. Stability of requirements
6. Ease of customer/develop communication
7. Maturity of applicable technology
8. Performance constraints
9. Embedded, non embedded characteristics
10. Project staff
11. Reengineering factors

Any 10 points -----3 marks

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
SIXTH SEMESTER BTECH DEGREE EXAMINATION
CS 364 MOBILE COMPUTING

Time: 3 Hours

Max. Marks: 100

PART A (Answer all questions-3 marks each)

1. Give examples for six mobile computing applications.
2. What are the differences between middleware and gateways? Enunciate with examples in the context of Mobile Computing?
3. Compare the mechanisms of SDMA, TDMA, FDMA and CDMA.
4. What is a handoff? What are the reasons for a handoff to be conducted?

PART B (Answer any two questions- 9 marks each)

5. Explain the three-tier architecture for Mobile Computing with the help of a figure.
6. Explain Dynamic channel assignment and fixed channel assignment.
7. Explain the co-channel interference in cellular systems.

PART C (Answer all questions-3 marks each)

8. What are the advantages of wireless LANs?
9. What are the design goals for wireless LANs?
10. How can DHCP be used for mobility and support of mobile IP?
11. What are the main features of the WAP approach to transport connections?

PART D (Answer any two questions- 9 marks each)

12. Explain the architecture of an infrastructure -based IEEE 802.11 with the help of figure.
13. Explain snooping TCP. What are it's advantages and disadvantages?
14. Compare the different approaches for a “mobile” TCP?

PART E (Answer any four questions- 10 marks each)

15. Write short notes on
a) Bluetooth (5 marks)
b) XML (5 marks)
16. Write short notes on
a) J2ME (5 marks)
b) JavaCard (5marks)
17. Explain the features of
a) PalmOS (5marks)
b) Android (5marks)
18. a) What are the security issues involved in Mobile Computing (5 marks)
b) What are the components of Information Security? (5 marks)
19. a) What are the features of LTE? (5 marks)
b)What is MIMO? What are its categories? (5 marks)
20. a) What are the applications of 5G? (5 marks)
b) What is LiFi? What are its applications (5 marks)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

SIXTH SEMESTER BTECH DEGREE EXAMINATION

COURSE CODE: CS 364

COURSE NAME: MOBILE COMPUTING

ANSWER KEY

PART A

1. The six basic mobile computing applications are :

Personal (wallet, diary, etc.)

Perishable (news, sports, stock quotes, etc.)

Transaction oriented (bank transactions, mobile shopping,etc.)

Location specific (restaurant guide, map service, etc.)

Corporate (ERP, inventory, business alerts, etc.)

Entertainment (fun, games, etc.)

2. Any software layered between a user application and operating system can be termed as middleware. Middleware examples are communication middleware, object oriented middleware, message oriented middleware, transaction processing middleware, database middleware, behaviour management middleware, RPC middleware etc.

Between the device and the middleware there will be network of networks. Gateways are deployed when there are different transport bearers or networks with dissimilar protocols. For example, we need an IVR (Interactive Voice Response) gateway to interface voice with a computer, or an WAP gateway to access internet over a mobile phone.

3.	Approach	SDMA	TDMA	FDMA	CDMA
Idea	segment space into cells/sectors	segment sending time into disjoint time-slots, demand driven or fixed patterns	segment the frequency band into disjoint sub-bands	spread the spectrum using orthogonal codes	
Terminals	only one terminal can be active in one cell/one sector	all terminals are active for short periods of time on the same frequency	every terminal has its own frequency, uninterrupted	all terminals can be active at the same place at the same moment, uninterrupted	
Signal separation	cell structure, directed antennas	synchronization in the time domain	filtering in the frequency domain	code plus special receivers	
Advantages	very simple, increases capacity per km ²	established, fully digital, flexible	simple, established, robust	flexible, less frequency planning needed, soft handover	
Dis-advantages	inflexible, antennas typically fixed	guard space needed (multipath propagation), synchronization difficult	inflexible, frequencies are a scarce resource	complex receivers, needs more complicated power control for senders	
Comment	only in combination with TDMA, FDMA or CDMA useful	standard in fixed networks, together with FDMA/SDMA used in many mobile networks	typically combined with TDMA (frequency hopping patterns) and SDMA (frequency reuse)	still faces some problems, higher complexity, lowered expectations; will be integrated with TDMA/FDMA	

4. Handoff refers to a process of transferring an ongoing call or data session from one channel connected to the core network to another. Process of transferring a MS from one base station to another. Also called as 'Handover'.

Reasons for a Handoff to be conducted are:

To avoid call termination: call drops

When the capacity for connecting new calls of a given cell is used up.

Interference in the channels.

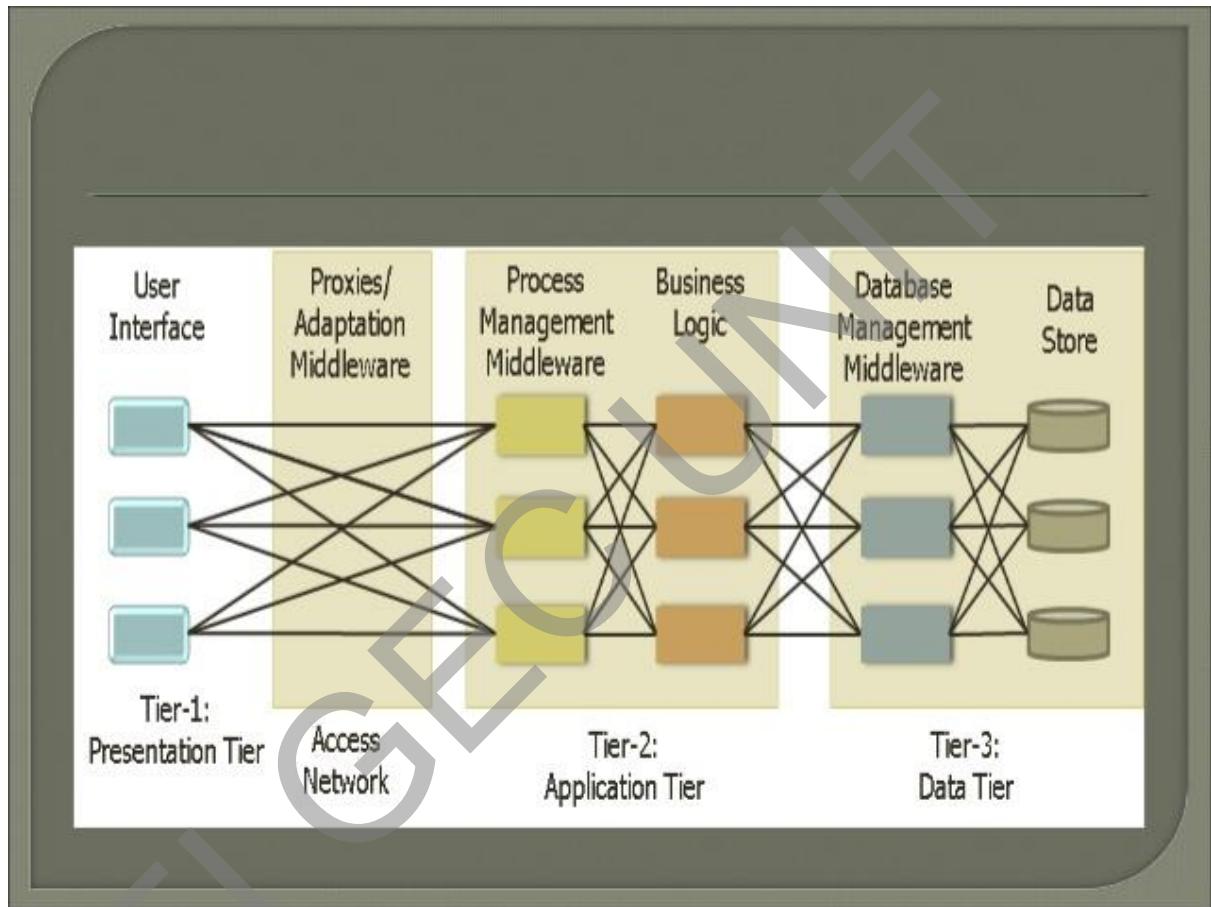
When the user behaviors change.

Speed and mobility.

PART B

5. The network-centric mobile computing architecture uses three-tier architecture

- 1) Presentation Tiers
- 2) Application Tiers
- 3) Data Tier



In three tier architecture, the first layer is User Interface or Presentation Tier. This layer deal with user facing device handling and rendering. This tier includes a user system interface where user service reside.

The second tier is the Process Management or Application Tier .This layer is capable of accommodating hundreds of users. The middle process management tier controls transaction and asynchronous queuing to ensure reliable completion of transaction.

The third final tier is the Database Management or Data Tier. This Layer is for database access and management. The three-tier architecture is better suited for an effective networked client/server design. It provide increased performance , flexibility , maintainability , reusability and scalability , while hiding the complexity of distributed processing from user. All these characteristic have made three-tier architecture a popular choice for Inter application and net-centric information system .

6. Objectives of channel assignment strategies are :

Increasing capacity

Minimizing interference

Classification of channel assignment strategies are :

Fixed channel assignment strategies

Dynamic channel assignment strategies

Fixed channel assignment

Each cell is allocated a predetermined set of channels. Any call attempt within the cell can only be served by the unused channels in that particular cell. If all the channels in that cell are occupied, the call is blocked and the subscriber does not receive service.

Dynamic channel assignment strategies

Channels are not allocated to different cells permanently. Each time a call request is made, the serving base station requests a channel from the MSC. The switch then allocates a channel to the requested cell following an algorithm that takes into account:

the likelihood of fixture blocking within the cell

the frequency of use of the candidate channel

the reuse distance of the channel

other cost functions.

7. Cells using the same set of frequencies are called cochannel cells, and the interference between signals from these cells is called co-channel interference. Unlike thermal noise which can be overcome by increasing the signal-to-noise ratio (SNR), co-channel interference cannot be combated by simply increasing the carrier power of a transmitter. This is because an increase in carrier transmit power increases the interference to neighboring co-channel cells. To reduce co-channel interference, co-channel cells must be physically separated by a minimum distance to provide sufficient isolation due to propagation.

The co-channel interference ratio is a function of the radius of the cell (R) and the distance between centers of the nearest cochannel cells (D). By increasing the ratio of D/R , the spatial separation between co-channel cells relative to the coverage distance of a cell is increased. Thus interference is reduced.

The parameter $Q = D/R$, called the co channel reuse ratio, is related to the cluster size N .

When the size of each cell is approximately the same, and the base stations transmit the same power, we have $Q = D/R = (3N)^{1/3}$

A small value of Q provides larger capacity since the cluster size N is small, whereas a large value of Q improves the transmission quality, due to a smaller level of co-channel interference.

A trade-off must be made between these two objectives in actual cellular design.

PART C

8. Very flexible within reception area
Ad-hoc networks do not need planning
(almost) no wiring difficulties (e.g. historic buildings, firewalls)
more robust against disasters like, e.g., earthquakes, fire

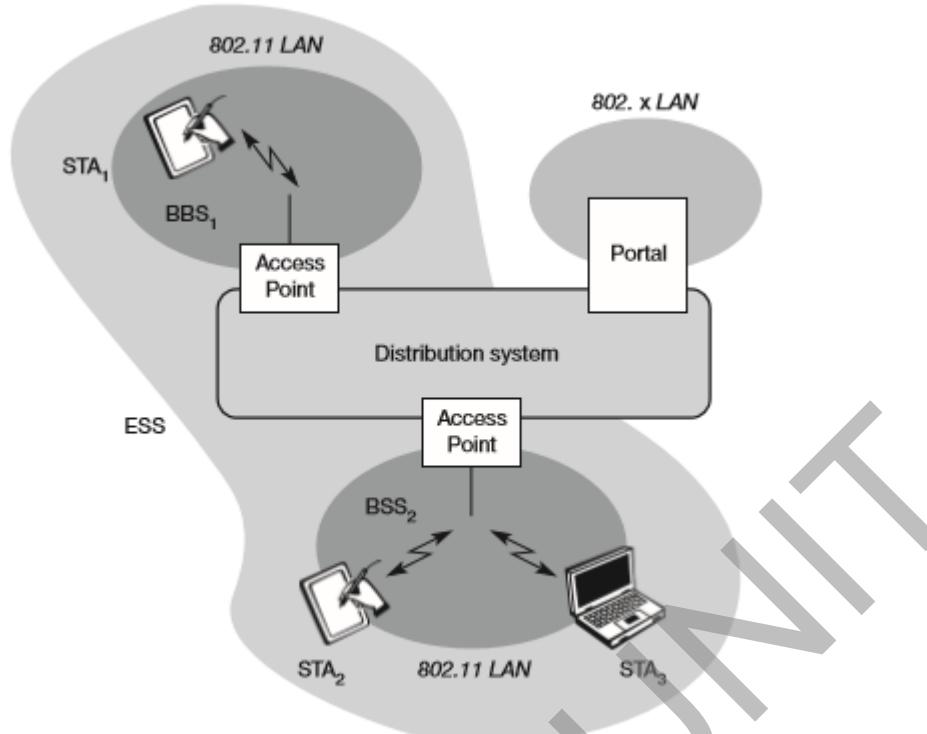
9. Global, seamless operation
Low power for battery use
No special permissions or licenses needed to use the LAN
Robust transmission technology
Simplified spontaneous cooperation at meetings
Easy to use for everyone, simple management
Protection of investment in wired networks
Security (no one should be able to read my data), privacy (no one should be able to collect user profiles), safety (low radiation)
Transparency concerning applications and higher layer protocols, but also location awareness if necessary

10. DHCP is a good candidate for support the acquisition of COA for mobile nodes. The same holds for all other parameters needed, such as address of the default router, DNS servers etc. A DHCP server should be located in the subnet of the access point of the mobile node, or at least a DHCP relay should provide forwarding of the messages.

11. WAP, in version 1 at least, is a replacement technology for TCP. WAP provides a full stack of protocols that operate on the wireless link (i.e. between base station and wireless terminal). In this way WAP is separate from the wider Internet, as the WAP gateway translates all requests into TCP/IP to be sent over the Internet. This approach allows WAP to be optimised for the terminal whilst not affecting the operation of the fixed Internet.

PART D

- 12.



Several nodes, called stations (STA_i), are connected to access points (AP). Stations are terminals with access mechanisms to the wireless medium and radio contact to the AP. The stations and the AP which are within the same radio coverage form a basic service set (BSS_i). The example shows two BSSs – BSS1 and BSS2 – which are connected via a distribution system. A distribution system connects several BSSs via the AP to form a single network and thereby extends the wireless coverage area. This network is now called an extended service set (ESS) and has its own identifier, the ESSID. The ESSID is the ‘name’ of a network and is used to separate different networks. Without knowing the ESSID (and assuming no hacking) it should not be possible to participate in the WLAN. The distribution system connects the wireless networks via the APs with a portal, which forms the interworking unit to other LANs.

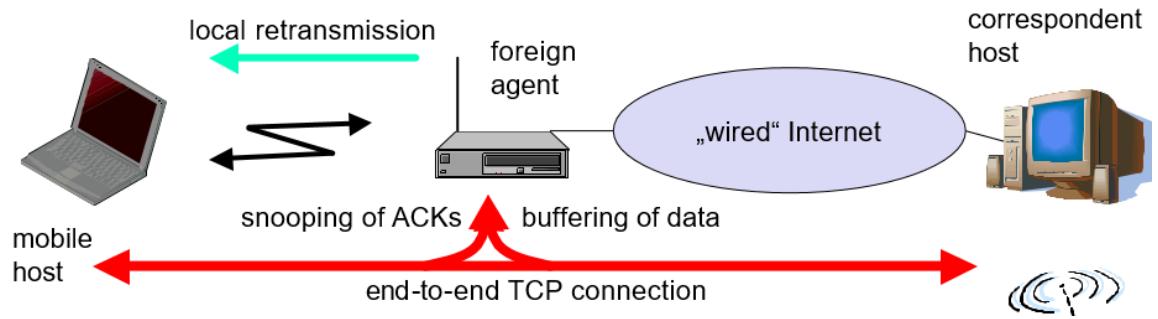
Stations can select an AP and associate with it. The APs support roaming (i.e., changing access points), the distribution system handles data transfer between the different APs. APs provide synchronization within a BSS, support power management, and can control medium access to support time-bounded service.

13. ‘Transparent’ extension of TCP within the foreign agent

Buffering of packets sent to the mobile host

Lost packets on the wireless link (both directions!) will be retransmitted immediately by the mobile host or foreign agent, respectively (so called “local” retransmission)
The foreign agent therefore “snoops” the packet flow and recognizes acknowledgements in both directions, it also filters ACKs

Changes of TCP only within the foreign agent



Data transfer to the mobile host

FA buffers data until it receives ACK of the MH, FA detects packet loss via duplicated ACKs or time-out

Fast retransmission possible, transparent for the fixed network

Data transfer from the mobile host

FA detects packet loss on the wireless link via sequence numbers, FA answers directly with a NACK to the MH.

MH can now retransmit data with only a very short delay.

Integration with MAC layer

MAC layer often has similar mechanisms to those of TCP.

Thus, the MAC layer can already detect duplicated packets due to retransmissions and discard them.

Problems

Snooping TCP does not isolate the wireless link as good as I-TCP .

Snooping might be tough if packets are encrypted.

14.

Approach	Mechanism	Advantages	Disadvantages
Indirect TCP	splits TCP connection into two connections	isolation of wireless link, simple	loss of TCP semantics, higher latency at handover
Snooping TCP	“snoops” data and acknowledgements, local retransmission	transparent for end-to-end connection, MAC integration possible	problematic with encryption, bad isolation of wireless link
M-TCP	splits TCP connection, chokes sender via window size	Maintains end-to-end semantics, handles long term and frequent disconnections	Bad isolation of wireless link, processing overhead due to bandwidth management
Fast retransmit/fast recovery	avoids slow-start after roaming	simple and efficient	mixed layers, not transparent
Transmission/time-out freezing	freezes TCP state at disconnect, resumes after reconnection	independent of content or encryption, works for longer interrupts	changes in TCP required, MAC dependant
Selective retransmission	retransmit only lost data	very efficient	slightly more complex receiver software, more buffer needed
Transaction oriented TCP	combine connection setup/release and data transmission	Efficient for certain applications	changes in TCP required, not transparent

PART E

15.a) Bluetooth is a wireless technology standard for exchanging data over short distances (using short-wavelength UHF radio waves in the ISM band from 2.4 to 2.485 GHz) from fixed and mobile devices, and building personal area networks (PANs). Invented by telecom vendor Ericsson in 1994, it was originally conceived as a wireless alternative to RS-232 data cables.

Bluetooth is managed by the Bluetooth Special Interest Group (SIG), which has more than 30,000 member companies in the areas of telecommunication, computing, networking, and consumer electronics. The IEEE standardized Bluetooth as IEEE 802.15.1, but no longer maintains the standard. The Bluetooth SIG oversees development of the specification, manages the qualification program, and protects the trademarks. A manufacturer must meet

Bluetooth SIG standards to market it as a Bluetooth device. A network of patents apply to the technology, which are licensed to individual qualifying devices.

Bluetooth operates at frequencies between 2402 and 2480 MHz, or 2400 and 2483.5 MHz including guard bands 2 MHz wide at the bottom end and 3.5 MHz wide at the top. This is in the globally unlicensed (but not unregulated) industrial, scientific and medical (ISM) 2.4 GHz short-range radio frequency band. Bluetooth uses a radio technology called frequency-hopping spread spectrum. Bluetooth divides transmitted data into packets, and transmits each packet on one of 79 designated Bluetooth channels. Each channel has a bandwidth of 1 MHz. It usually performs 800 hops per second, with Adaptive Frequency-Hopping (AFH) enabled. Bluetooth Low Energy uses 2 MHz spacing, which accommodates 40 channels.

15.b) In computing, Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The W3C's XML 1.0 Specification and several other related specifications—all of them free open standards—define XML.

The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

Several schema systems exist to aid in the definition of XML-based languages, while programmers have developed many application programming interfaces (APIs) to aid the processing of XML data.

Hundreds of document formats using XML syntax have been developed, including RSS, Atom, SOAP, SVG, and XHTML. XML-based formats have become the default for many office-productivity tools, including Microsoft Office (Office Open XML), OpenOffice.org and LibreOffice (OpenDocument), and Apple's iWork. XML has also provided the base language for communication protocols such as XMPP. Applications for the Microsoft .NET Framework use XML files for configuration. Apple has an implementation of a registry based on XML.

Most industry data standards, e.g. HL7, OTA, NDC, FpML, MISMO etc. are based on XML and the rich features of the XML schema specification. Many of these standards are quite complex and it is not uncommon for a specification to comprise several thousand pages.

In publishing, DITA is an XML industry data standard. XML is used extensively to underpin various publishing formats.

XML is widely used in a Services Oriented Architecture (SOA). Disparate systems communicate with each other by exchanging XML messages. The message exchange format is standardised as an XML schema (XSD). This is also referred to as the canonical schema.

XML has come into common use for the interchange of data over the Internet. IETF RFC:3023, now superseded by RFC:7303, gave rules for the construction of Internet Media Types for use when sending XML. It also defines the media types application/xml and text/xml, which say only that the data is in XML, and nothing about its semantics. The use of text/xml has been criticized as a potential source of encoding problems and it has been suggested that it should be deprecated.

RFC 7303 also recommends that XML-based languages be given media types ending in +xml; for example image/svg+xml for SVG.

Further guidelines for the use of XML in a networked context appear in RFC 3470, also known as IETF BCP 70, a document covering many aspects of designing and deploying an XML-based language.

16. a) Java 2 Micro Edition (J2ME) is Sun's version of Java aimed at machines with limited hardware resources such as PDAs, cell phones, and other consumer electronic and embedded devices. J2ME is aimed at machines with as little as 128KB of RAM and with processors a lot less powerful than those used on typical desktop and server machines. J2ME actually consists of a set of profiles. Each profile is defined for a particular type of device -- cell phones, PDAs, microwave ovens, etc. -- and consists of a minimum set of class libraries required for the particular type of device and a specification of a Java virtual machine required to support the device. The virtual machine specified in any profile is not necessarily the same as the virtual machine used in Java 2 Standard Edition (J2SE) and Java 2 Enterprise Edition (J2EE)

Sun has released the following profiles:

- The Foundation Profile -- A profile for next generation consumer electronic devices
- The Mobile Information Device Profile (MIDP) -- A profile for mobile information devices, such as cellular phones and two-way pagers, and PDAs

Some of the configurations currently available are

Connected Device Configuration (CDC) : An implementation of the Foundation Profile for next-generation, consumer electronic and embedded devices

Connected Limited Device Configuration (CLDC): An implementation of MIDP for small, resource-constrained devices such as Palm OS devices.

16.b) Java Card refers to a software technology that allows Java-based applications (applets) to be run securely on smart cards and similar small memory footprint devices. Java Card is the tiniest of Java platforms targeted for embedded

devices. Java Card gives the user the ability to program the devices and make them application specific. It is widely used in SIM cards (used in GSM mobile phones) and ATM cards. The first Java Card was introduced in 1996 by Schlumberger's card division which later merged with Gemplus to form Gemalto. Java Card products are based on the Java Card Platform specifications developed by Sun Microsystems (later a subsidiary of Oracle Corporation). Many Java card products also rely on the GlobalPlatform specifications for the secure management of applications on the card (download, installation, personalization, deletion).

The main design goals of the Java Card technology are portability and security.

Portability

Java Card aims at defining a standard smart card computing environment allowing the same Java Card applet to run on different smart cards, much like a Java applet runs on different computers. As in Java, this is accomplished using the combination of a virtual machine (the Java Card Virtual Machine), and a well-defined runtime library, which largely abstracts the applet from differences between smart cards. Portability remains mitigated by issues of memory size, performance, and runtime support (e.g. for communication protocols or cryptographic algorithms).

Security

Java Card technology was originally developed for the purpose of securing sensitive information stored on smart cards. Security is determined by various aspects of this technology:

Data encapsulation

Data is stored within the application, and Java Card applications are executed in an isolated environment (the Java Card VM), separate from the underlying operating system and hardware.

Applet Firewall

Unlike other Java VMs, a Java Card VM usually manages several applications, each one controlling sensitive data. Different applications are therefore separated from each other by an applet firewall which restricts and checks access of data elements of one applet to another.

Cryptography

Commonly used symmetric key algorithms like DES, Triple DES, AES, and asymmetric key algorithms such as RSA, elliptic curve cryptography are supported as well as other cryptographic services like signing, key generation and key exchange.

Applet

The applet is a state machine which processes only incoming command requests and responds by sending data or response status words back to the interface device.

17 a) Palm OS (also known as Garnet OS) is a discontinued mobile operating system initially developed by Palm, Inc., for personal digital assistants (PDAs) in 1996. Palm OS was designed for ease of use with a touchscreen-based graphical user interface. It is provided with a suite of basic applications for personal information management. Later versions of the OS have been extended to support smartphones. Several other licensees have manufactured devices powered by Palm OS.

Following Palm's purchase of the Palm trademark, the currently licensed version from ACCESS was renamed *Garnet OS*. In 2007, ACCESS introduced the successor to Garnet OS, called Access Linux Platform and in 2009, the main licensee of Palm OS, Palm, Inc., switched from Palm OS to webOS for their forthcoming devices.

Palm OS is a proprietary mobile operating system. Designed in 1996 for Palm Computing, Inc.'s new Pilot PDA, it has been implemented on a wide array of mobile devices, including smartphones, wrist watches, handheld gaming consoles, barcode readers and GPS devices.

Palm OS versions earlier than 5.0 run on Motorola/Freescale DragonBall processors. From version 5.0 onwards, Palm OS runs on ARM architecture-based processors.

The key features of the current Palm OS Garnet are:

- Simple, single-tasking environment to allow launching of full screen applications with a basic, common GUI set
- Monochrome or color screens with resolutions up to 480x320 pixel
- Handwriting recognition input system called Graffiti 2
- HotSync technology for data synchronization with desktop computers
- Sound playback and record capabilities
- Simple security model: Device can be locked by password, arbitrary application records can be made private
- TCP/IP network access
- Serial port/USB, infrared, Bluetooth and Wi-Fi connections
- Expansion memory card support
- Defined standard data format for personal information management applications to store calendar, address, task and note entries, accessible by third-party applications.

17. b) Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The operating system has since gone through multiple major releases, with the current version being 8.1 "Oreo", released in December 2017.

The main hardware platform for Android is the ARM (ARMv7 and ARMv8-A architectures), with x86, MIPS and MIPS64, and x86-64 architectures also officially supported in later versions of Android. The unofficial Android-x86project provided support for the x86 architectures ahead of the official support. MIPS architecture was also supported before Google did. Since 2012, Android devices with Intel processors began to appear, including phones and tablets. While gaining support for 64-bit platforms, Android was first made to run on 64-bit x86 and then on ARM64. Since Android 5.0 "Lollipop", 64-bit variants of all platforms are supported in addition to the 32-bit variants.

18 a) Mobile security or mobile phone security has become increasingly important in mobile computing. It is of particular concern as it relates to the security of personal information now stored on the smart phone. More and more users and businesses use smart phones as communication tools but also as a means of planning and organizing their work and private life. Within companies, these technologies are causing profound changes in the organization of information systems and therefore they have become the source of new risks. Indeed, smart phones collect and compile an increasing amount of sensitive information to which access must be controlled to protect the privacy of the user and the intellectual property of the company

All smart phones, as computers, are preferred targets of attacks. These attacks exploit weaknesses related to smart phones that can come from means of communication like SMS, MMS, WIFI NETWORKS. There are also attacks that exploit software vulnerabilities from both the web browser and operating system.

Different security counter-measures are being developed and applied to smart phones, from security in different layers of software to the dissemination of information to end users. There are good practices to be observed at all levels, from design to use, through the development of operating systems, software layers, and downloadable apps.

One of the key issues of these being, confidentiality and authentication, where the user must be protected from unauthorized eavesdropping. The goal of authentication protocol is to check the identity of other users or network centers before providing access to the confidential information on the user side. When designing any security protocol, there are certain conditions that

need to be considered. Firstly, the low computational power of the mobile users and secondly, the low bandwidth available. Therefore, it is important to design the security protocols so as to minimize, the number of message exchanges and the message size. a few authentication protocols that were proposed to provide security between the users and the network. These protocols are based on the use of certificates, which are built on the concept of security keys (cryptography). Another protocol is the Kilo Byte Secure Socket Layer (KSSL) protocol, which is an extension of Secure Socket Layer (SSL) protocol used for wired networks.

18.b) Computer security rests on confidentiality, integrity, and availability. The interpretations of these three aspects vary, as do the contexts in which they arise. The interpretation of an aspect in a given environment is dictated by the needs of the individuals, customs, and laws of the particular organization.

Confidentiality

Confidentiality is the concealment of information or resources. The need for keeping information secret arises from the use of computers in sensitive fields such as government and industry. For example, military and civilian institutions in the government often restrict access to information to those who need that information. The first formal work in computer security was motivated by the military's attempt to implement controls to enforce a "need to know" principle. This principle also applies to industrial firms, which keep their proprietary designs secure lest their competitors try to steal the designs. As a further example, all types of institutions keep personnel records secret.

Eg. Enciphering an income tax return will prevent anyone from reading it. If the owner needs to see the return, it must be deciphered. Only the possessor of the cryptographic key can enter it into a deciphering program. However, if someone else can read the key when it is entered into the program, the confidentiality of the tax return has been compromised.

Integrity

Integrity refers to the trustworthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized change. Integrity includes data integrity (the content of the information) and origin integrity (the source of the data, often called *authentication*). The source of the information may bear on its accuracy and credibility and on the trust that people place in the information. This dichotomy illustrates the principle that the aspect of

integrity known as credibility is central to the proper functioning of a system. We will return to this issue when discussing malicious logic.

Eg. A newspaper may print information obtained from a leak at the White House but attribute it to the wrong source. The information is printed as received (preserving data integrity), but its source is incorrect (corrupting origin integrity).

Availability

Availability refers to the ability to use the information or resource desired. Availability is an important aspect of reliability as well as of system design because an unavailable system is at least as bad as no system at all. The aspect of availability that is relevant to security is that someone may deliberately arrange to deny access to data or to a service by making it unavailable. System designs usually assume a statistical model to analyze expected patterns of use, and mechanisms ensure availability when that statistical model holds. Someone may be able to manipulate use (or parameters that control use, such as network traffic) so that the assumptions of the statistical model are no longer valid. This means that the mechanisms for keeping the resource or data available are working in an environment for which they were not designed. As a result, they will often fail.

Eg. Suppose Anne has compromised a bank's secondary system server, which supplies bank account balances. When anyone else asks that server for information, Anne can supply any information she desires. Merchants validate checks by contacting the bank's primary balance server. If a merchant gets no response, the secondary server will be asked to supply the data. Anne's colleague prevents merchants from contacting the primary balance server, so all merchant queries go to the secondary server. Anne will never have a check turned down, regardless of her actual account balance. Notice that if the bank had only one server (the primary one), this scheme would not work. The merchant would be unable to validate the check.

19. a) In telecommunication, Long-Term Evolution (LTE) is a standard for high-speed wireless communication for mobile devices and data terminals, based on theGSM/EDGE and UMTS/HSPA technologies. It increases the capacity and speed using a different radio interface together with core network improvements.^{[1][2]} The standard is developed by the 3GPP (3rd Generation Partnership Project) and is specified in its Release 8 document series, with minor enhancements described in Release 9. LTE is the upgrade path for

carriers with both GSM/UMTS networks and CDMA2000 networks.

The different LTE frequencies and bands used in different countries mean that only multi-band phones are able to use LTE in all countries where it is supported.

Its main features are:

- Peak download rates up to 299.6 Mbit/s and upload rates up to 75.4 Mbit/s depending on the user equipment category (with 4x4 antennas using 20 MHz of spectrum). Five different terminal classes have been defined from a voice centric class up to a high end terminal that supports the peak data rates. All terminals will be able to process 20 MHz bandwidth.
- Low data transfer latencies (sub-5 ms latency for small IP packets in optimal conditions), lower latencies for handover and connection setup time than with previous radio access technologies.
- Improved support for mobility, exemplified by support for terminals moving at up to 350 km/h (220 mph) or 500 km/h (310 mph) depending on the frequency band.
- Orthogonal frequency-division multiple access for the downlink, Single-carrier FDMA for the uplink to conserve power.
- Support for both FDD and TDD communication systems as well as half-duplex FDD with the same radio access technology.
- Support for all frequency bands currently used by IMT systems by ITU-R.
- Increased spectrum flexibility: 1.4 MHz, 3 MHz, 5 MHz, 10 MHz, 15 MHz and 20 MHz wide cells are standardized. (W-CDMA has no option for other than 5 MHz slices, leading to some problems rolling-out in countries where 5 MHz is a commonly allocated width of spectrum so would frequently already be in use with legacy standards such as 2G GSM and cdmaOne.)
- Support for cell sizes from tens of metres radius (femto and picocells) up to 100 km (62 miles) radius macrocells. In the lower frequency bands to be used in rural areas, 5 km (3.1 miles) is the optimal cell size, 30 km (19 miles) having reasonable performance, and up to 100 km cell sizes supported with acceptable performance. In city and urban areas, higher frequency bands (such as 2.6 GHz in EU) are used to support high speed mobile broadband. In this case, cell sizes may be 1 km (0.62 miles) or even less.
- Supports at least 200 active data clients in every 5 MHz cell.
- Simplified architecture: The network side of E-UTRAN is composed only of eNode Bs.
- Support for inter-operation and co-existence with legacy standards (e.g., GSM/EDGE, UMTS and CDMA2000). Users can start a call or transfer of data in an area using an LTE standard, and, should coverage be unavailable, continue the operation without any action on their part using GSM/GPRS or W-CDMA-based UMTS or even 3GPP2 networks such as cdmaOne or CDMA2000.

- Packet switched radio interface.
- Support for MBSFN (Multicast-broadcast single-frequency network). This feature can deliver services such as Mobile TV using the LTE infrastructure, and is a competitor for DVB-H-based TV broadcast only LTE compatible devices receives LTE signal.

19 b) In radio, multiple-input and multiple-output, or MIMO is a method for multiplying the capacity of a radio link using multiple transmit and receive antennas to exploit multipath propagation. MIMO has become an essential element of wireless communication standards including IEEE 802.11n (Wi-Fi), IEEE 802.11ac (Wi-Fi), HSPA+(3G), WiMAX (4G), and Long Term Evolution (LTE 4G). More recently, MIMO has been applied to power-line communication for 3-wire installations as part of ITU G.hn standard and HomePlug AV2 specification.

MIMO can be sub-divided into three main categories: precoding, spatial multiplexing (SM), and diversity coding.

Precoding is multi-stream beamforming, in the narrowest definition. In more general terms, it is considered to be all spatial processing that occurs at the transmitter. In (single-stream) beamforming, the same signal is emitted from each of the transmit antennas with appropriate phase and gain weighting such that the signal power is maximized at the receiver input. The benefits of beamforming are to increase the received signal gain – by making signals emitted from different antennas add up constructively – and to reduce the multipath fading effect. In line-of-sight propagation, beamforming results in a well-defined directional pattern. However, conventional beams are not a good analogy in cellular networks, which are mainly characterized by multipath propagation. When the receiver has multiple antennas, the transmit beamforming cannot simultaneously maximize the signal level at all of the receive antennas, and precoding with multiple streams is often beneficial. Note that precoding requires knowledge of channel state information (CSI) at the transmitter and the receiver.

Spatial multiplexing requires MIMO antenna configuration. In spatial multiplexing, a high-rate signal is split into multiple lower-rate streams and each stream is transmitted from a different transmit antenna in the same frequency channel. If these signals arrive at the receiver antenna array with sufficiently different spatial signatures and the receiver has accurate CSI, it can separate these streams into (almost) parallel channels. Spatial multiplexing is a very powerful technique for increasing channel capacity at higher signal-to-noise ratios (SNR). The maximum number of spatial streams is limited by the lesser of the number of antennas at the transmitter or receiver. Spatial multiplexing can be used without CSI at the transmitter, but can be combined with precoding if CSI is available. Spatial multiplexing can also be used for simultaneous transmission to multiple receivers, known as space-division multiple access or multi-user MIMO, in which case CSI is required at the transmitter. The scheduling of receivers with different spatial signatures allows good separability.

Diversity coding techniques are used when there is no channel knowledge at the transmitter. In diversity methods, a single stream (unlike multiple streams in spatial multiplexing) is transmitted, but the signal is coded using techniques called space-time coding. The signal is emitted from each of the transmit antennas with full or near orthogonal coding. Diversity coding exploits the independent fading in the multiple antenna links to enhance signal diversity. Because there is no channel knowledge, there is no beamforming or array gain from diversity coding. Diversity coding can be combined with spatial multiplexing when some channel knowledge is available at the transmitter.

20 .a) 5G technology is adorned with many as well as distinct features, which applicability is useful for a wide range people irrespective of their purposes

Applications of 5G

Some of the significant applications are –

- It will make unified global standard for all.
- Network availability will be everywhere and will facilitate people to use their computer and such kind of mobile devices anywhere anytime.
- Because of the IPv6 technology, visiting care of mobile IP address will be assigned as per the connected network and geographical position.
- Its application will make world real Wi Fi zone.
- Its cognitive radio technology will facilitate different version of radio technologies to share the same spectrum efficiently.
- Its application will facilitate people to avail radio signal at higher altitude as well.

20.b) Li-Fi is a technology for wireless communication between devices using light to transmit data. In its present state only LED lamps can be used for the transmission of visible light. The term was first introduced by Harald Haas during a 2011 TEDGlobal talk in Edinburgh. In technical terms, Li-Fi is a visible light communications system that is capable of transmitting data at high speeds over the visible light spectrum, ultraviolet and infrared radiation.

In terms of its end use the technology is similar to Wi-Fi. The key technical difference is that Wi-Fi uses radio frequency to transmit data. Using light to transmit data allows Li-Fi to offer several advantages like working across higher bandwidth, working in areas susceptible to electromagnetic interference (e.g. aircraft cabins, hospitals) and offering higher transmission speeds. The technology is actively being developed by several organisations across the globe.

Applications

Security

In contrast to radio frequency waves used by Wi-Fi, lights cannot penetrate through walls and doors. This makes it more secure and makes it easier to control who can connect to a network. As long as transparent materials like windows are covered, access to a Li-Fi channel is limited to devices inside the room.

Underwater application

Most remotely operated underwater vehicles (ROVs) use cables to transmit command, but the length of cables then limits the area ROVs can detect. However, as a light wave could travel through water, Li-Fi could be implemented on vehicles to receive and send back signals.

While it is theoretically possible for Li-Fi to be used in underwater applications, its utility is limited by the distance light can penetrate water. Significant amounts of light do not penetrate further than 200 meters. Past 1000 meters, no light penetrates.

Hospital

Many treatments now involve multiple individuals, Li-Fi system could be a better system to transmit communication about the information of patients. Besides providing a higher speed, light waves also have little effect on medical instruments and human bodies.

Vehicles

Vehicles could communicate with one another via front and back lights to increase road safety. Also street lights and traffic signals could also provide information about current road situations.

Industrial automation

Anywhere in industrial areas data has to be transmitted, Li-Fi is capable to replace slip rings, sliding contacts and short cables, such as Industrial Ethernet. Due to real time capability of Li-Fi, which is often required for automation processes, it is also an alternative to common industrial Wireless LAN standards.

SFI GEC UNIT

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
SIXTH SEMESTER B.TECH DEGREE MODEL EXAMINATION
APRIL 2018
Course Code: CS 364
Course Name: MOBILE COMPUTING

Max. Marks:100

Duration: 3 Hours

PART A(Answer all questions)

(4x3=12)

1. Write a short note on middleware and gateways
2. Explain any three application and services of mobile computing
3. Explain satellite systems and list out its types
4. Write a short note on FHSS

PART B(Answer any two questions)

(2x9=18)

1. What are the different tiers in mobile computing architecture ?- describe the functions of each tiers.
2. Describe GSM architecture with neat diagram
3. Explain Medium access control in detail.

PART C (Answer all questions)

(4x3=12)

1. Describe the advantages and disadvantages of WLAN
Explain
2. Give short notes on classification of routing algorithms and list out its types.
3. Discuss the design goals of Mobile IP.
4. What is basic purpose of DHCP ? Name the entities of DHCP.

PART D (Answer any two questions)

(2x9=18)

1. Explain Routing algorithms in detail
2. Explain snooping TCP.What are its advantages and disadvantage
3. Draw and discuss the protocol architecture of IEEE 802.11.

PART E (Answer any four questions)

(4x10=40)

1. Explain in detail,Security issues in mobile computing.
2. Describe mobile transport layer in detail
3. Explain how XML can be used for mobile computing.
4. Describe Networks LTE Architecture & Interface in detail

5. Explain in detail about protocol architecture of WAP.
6. Give a short note about
 - a) Bluetooth
 - b) J2ME
 - c) JavaCard
 - d) XML

SFI GEC UNIT

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
SIXTH SEMESTER B.TECH DEGREE MODEL EXAMINATION, APRIL 2018
Course Code: CS 364
Course Name: MOBILE COMPUTING
ANSWER KEY

Max. Marks:100

Duration: 3 Hours

PART A(Answer all questions)

(4x3=12)

1. Write a short note on middleware and gateways

- A software layered between a user application and operating system is called middleware
- Communication middleware
 - Transaction Processing middleware
 - Behavior management middleware
- Definition of gateways [1 Mark]

[2 Marks]

2. Explain any three application and services of mobile computing

- Any three application and services -[1 mark for each]

3. Explain satellite systems and list out its types

- Definition -[1 mark]
→ Types with explanation- [2 Marks]
- i. Low Earth Orbit
 - ii. Medium Earth Orbit
 - iii. Geo stationary earth orbit

4. Write a short note on FHSS

- Important encoding method for wireless communications
- Spread data over wide bandwidth
- Makes jamming and interception harder
- Frequency hopping —Signal broadcast over seemingly random series of frequencies
- Direct Sequence
—Each bit is represented by multiple bits in transmitted signal

— Chipping code

- Input fed into channel encoder — Produces narrow bandwidth analog signal around central frequency

- Signal modulated using sequence of digits

— Spreading code/sequence

— Typically generated by pseudonoise/pseudorandom number generator

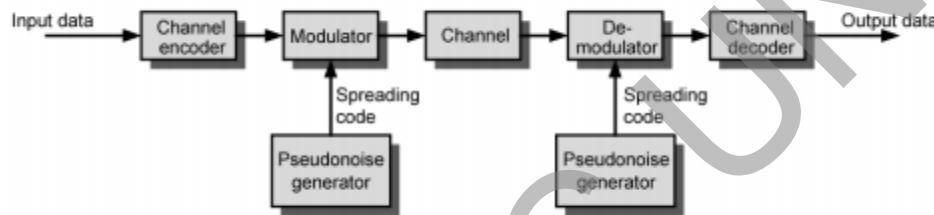
- Increases bandwidth significantly

— Spreads spectrum

- Receiver uses same sequence to demodulate signal

- Demodulated signal fed into channel decoder

General model of spread spectrum

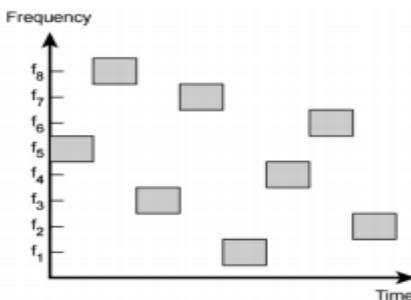
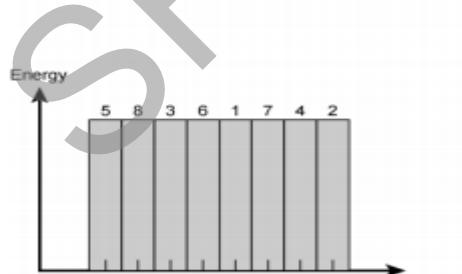


Frequency Hopping Spread Spectrum

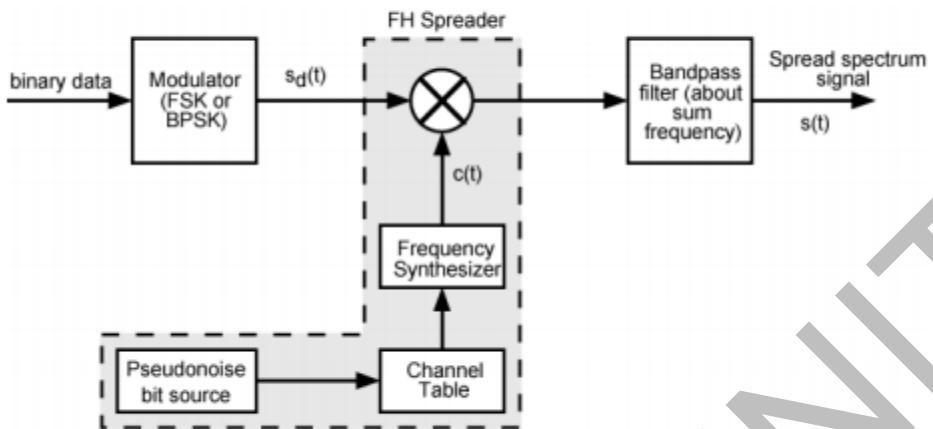
Signal broadcast over seemingly random series of frequencies

- Receiver hops between frequencies in sync with transmitter
- Eavesdroppers hear unintelligible blips
- Jamming on one frequency affects only a few bits

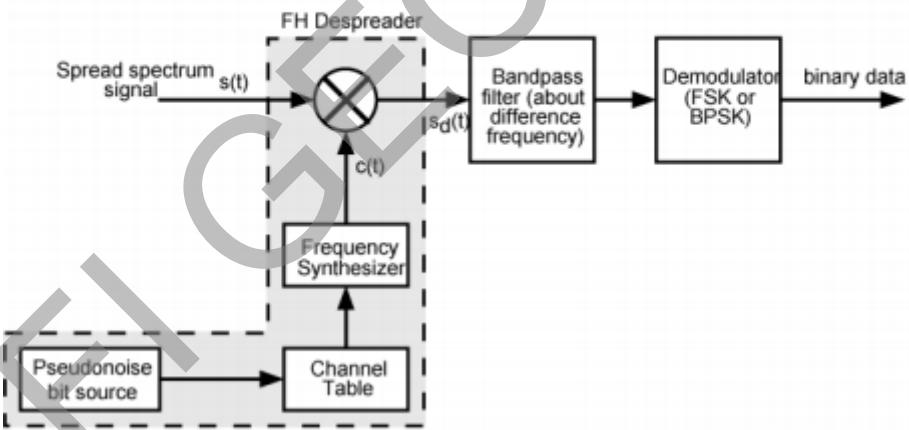
Frequency Hopping Example



Frequency Hopping Spread Spectrum System (Transmitter)



Frequency Hopping Spread Spectrum System (Receiver)



- Frequency shifted every T_c seconds
- Duration of signal element is T_s seconds
- Slow FHSS has $T_c \geq T_s$
- Fast FHSS has $T_c < T_s$
- Generally fast FHSS gives improved performance in noise (or jamming)

- Frequency shifted every T_c seconds
- Duration of signal element is T_s seconds
- Slow FHSS has $T_c \geq T_s$
- Fast FHSS has $T_c < T_s$
- Generally fast FHSS gives improved performance in noise (or jamming)

PART B(Answer any two questions)

(2x9=18)

1. What are the different tiers in mobile computing architecture

?- describe the functions of each tiers.

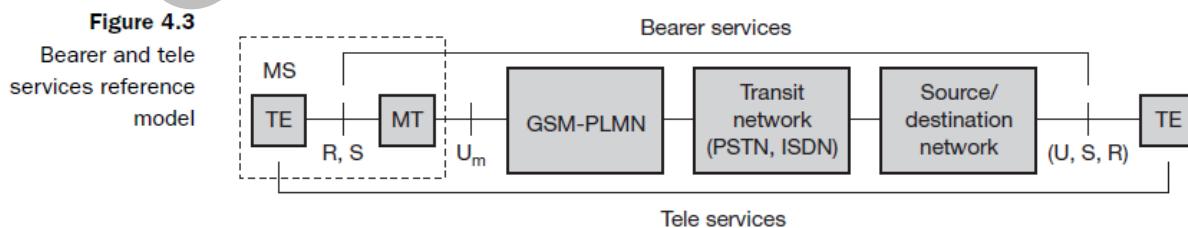
- Presentation-tier [3 marks for each functions]
- Application-tier
 - Fuctions of application tier and Middlewares
- Data-tier
 - syncML

2. Describe GSM architecture with neat diagram

[Architecture diagram 2 mark]

[GSM Services with Explanation 8 marks]

- GSM permits the integration of several types of connections,voice connections, data connections, short message service,multi-service options (combination of basic services)
- GSM had defined three different category of services
 - Bearer Services
 - Telematic Services
 - Supplementary Services



MS (Mobile Station) → TE - Terminal Equipment

→ MT – Mobile Termination

GSM-PLMN	→ GSM-Public Land Mobile Network
(R,S), Um, (U,S,R)	→ Interface
ISDN	→ Integrated Service Digital Network
PSTN	→ Public Switched Telephone Network

Bearer services

Services between first and last interface. It comprises all services that enable transparent transmission of data between interfaces of the network.

In a classical GSM model Bearer service is always connection oriented and Circuit or Packet switched.

Bearer Services uses only the 3 lower layers. Ie.,

- 1) Physical layer
- 2) Data Link layer
- 3) Network layer

It permits mainly 4 types of data transmission

- 1) Transparent data transmission
- 2) Non Transparent data transmission
- 3) Synchronous data transmission
- 4) Asynchronous data transmission

1) Transparent data transmission

- Transparent bearer services use only the physical layer. Here the transmission has a constant *delay* and *throughput*, if no error occurs.
- To increase the transmission quality, forward error correction method is introduced.
- It does not try to recover any lost data.

Example: Shadowing or interruptions due to handover

2) NonTransparent data transmission

- It uses protocols of data link and network layers for error corrections and flow of control

3) Synchronous data transmission

- Proper time slots basis data transmission.

4) Asynchronous data transmission No time slots.[Tele services](#)

Voice oriented tele services comprises encrypted voice transmission, message services and basic data communication between terminals.

Main characteristics of tele services

- 1) Telephony
- 2) Emergency number services
- 3) Short message services → which offers transmission of 160 character per message
- 4) Fax services

Supplementary services

Services are used to enhance it's standard telephony services.

- 1) Call forwarding
- 2) Call redirecting
- 3) User identification
- 4) Call waiting
- 5) Multi party communication
- 6) Conference call

System Architecture

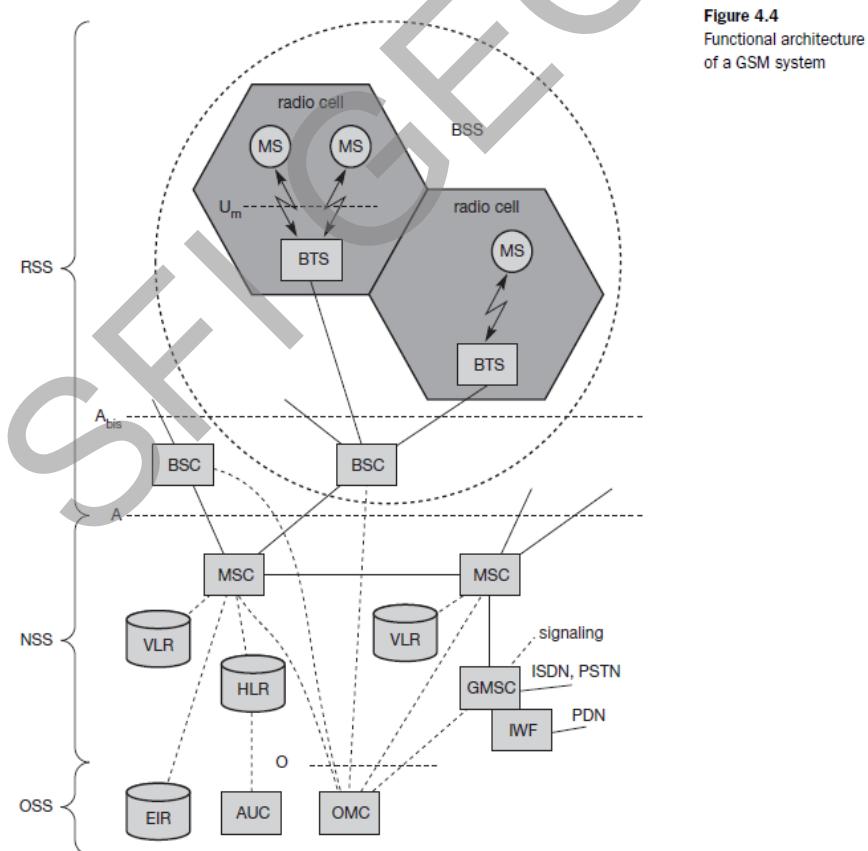


Figure 4.4
Functional architecture
of a GSM system

GSM has a hierarchical, complex structure comprising different entities.

3 Sub systems are:-

- 1) RSS (Radio sub system)
- 2) NSS (Network and switching sub system)
- 3) OSS (Operation sub system)

1) **RSS**

Higher layer in the architecture.

Entities involved are:-

MS (Mobile Station)
BSS (Base Station Sub System)
BTS (Base Transceiver Station)
BSC (Base Station Controller)

As the name implies RSS comprises all radio specific entities ie., MS and BSS

i) **BSS**

- A GSM network comprises many BSSs, each controlled by a base station controller (BSC).
- The BSS performs all functions necessary to maintain radio connections to an MS, coding/decoding of voice, and rate adaptation to/from the wireless network part.
- BSC, the BSS contains several BTSs.

ii) **BTSs.**

- BTS comprises all radio equipment, i.e., antennas, signal processing, amplifiers necessary for radio transmission.
- BTS connected to MS via the Um interface. And BTS connected to the BSC via the **Abis interface**.
- The Um interface contains all the mechanisms necessary for wireless transmission.
- The Abis interface consists of 16 or 64 kbit/s connections.

iii) **BSC**

- The BSC basically manages the BTSs.
- It reserves radio frequencies, handles the handover from one BTS to another within the BSS, and performs paging of the MS.

iv) **MS**

- The MS comprises all user equipment and software needed for communication with a GSM network.
- An MS consists of a **subscriber identity module (SIM)**, which stores all user-specific data that is relevant to GSM.3

- While an MS can be identified via the **international mobile equipment identity (IMEI)**

2) NSS

- The “heart” of the GSM system is formed by the **network and switching subsystem(NSS)**.
- The NSS connects the wireless network with standard public Networks.
- **NSS consist of the following switches and databases:-**

- 1) Mobile services switching center (MSC):
- 2) Home Location Register (HLR)
- 3) Visitor location register (VLR)

i) MSC

- MSCs are high-performance digital ISDN switches.
- They set up connections to other MSCs and to the BSCs via the A interface.
- **Gateway MSC (GMSC)** has additional connections to other fixed networks, such as **PSTN** and **ISDN**. Using additional **interworking functions (IWF)**, an MSC can also connect to **public data networks (PDN)** such as X.25.

ii) HLR

- i. HLR is the most important database in a GSM system as it stores all user-relevant information.

iii) VLR

- ii. VLR associated to each MSC is a dynamic database which stores all important information needed for the MS users currently in the LA that is associated to the MSC. (It stores all the information where the mobile visited.)

4) OSS

The third part of a GSM system, the **OSS**, contains the necessary functions for network operation and maintenance. The following entities have been defined:

- i. **Operation and maintenance center (OMC)**
- ii. **Authentication centre (AuC)**
- iii. **Equipment identity register (EIR)**

1.) Operation and maintenance center (OMC)

- The OMC monitors and controls all other network entities via the O interface

- OMCs use the concept of **telecommunication management network (TMN)**

2.) Authentication centre (AuC):

- AuC used to protect user identity and data transmission.
- The AuC contains the algorithms for authentication as well as the keys for encryption

3.) Equipment identity register (EIR):

- The EIR is a database for all IMEIs, i.e., it stores all device identifications registered for this network.
- The EIR also contains a list of valid IMEIs, and a list of malfunctioning devices.

3. Explain Medium access control in detail.

1. SDMA

- **Space Division Multiple Access (SDMA)** is used for allocating a separated space to users in wireless networks.
- A typical application involves assigning an optimal base station to a mobile phone user.
- The mobile phone may receive several base stations with different quality.
- A **MAC algorithm** could now decide which base station is best.
- SDMA is **never used in isolation** but always in combination with one or more other schemes.
- The basis for the SDMA algorithm is formed by cells and sectorized antennas which constitute the infrastructure implementing **space division multiplexing (SDM)**.
- Single users are separated in space by individual beams. This can improve the overall capacity of a cell tremendously.

2. FDMA

- **Frequency division multiple access (FDMA)** comprises all algorithms allocating frequencies to transmission channels according to the **frequency division multiplexing (FDM)**.
- Allocation can either be **fixed or dynamic**.
- Channels can be assigned to the same frequency at all times, i.e., **pure FDMA**, or change frequencies according to a certain pattern, i.e., **FDMA combined with TDMA**.

- Narrowband interference at certain frequencies, known as frequency hopping.
- Sender and receiver have to agree on a hopping pattern, otherwise the receiver could not tune to the right frequency.
- Hopping patterns are typically fixed, at least for a longer period.
- FDM is often used for simultaneous access to the medium by base station and mobile station in cellular networks. Here the two partners typically establish a **duplex channel**, i.e., a channel that allows for simultaneous transmission in both directions.
- The two directions, mobile station to base station and vice versa are now separated using different frequencies. This scheme is then called **frequency division duplex (FDD)**.
- Again, both partners have to know the frequencies in advance; The two frequencies are also known as **uplink**, i.e., from mobile station to base station or from ground control to satellite.
- **Downlink**, i.e., from base station to mobile station or from satellite to ground control.

Example:

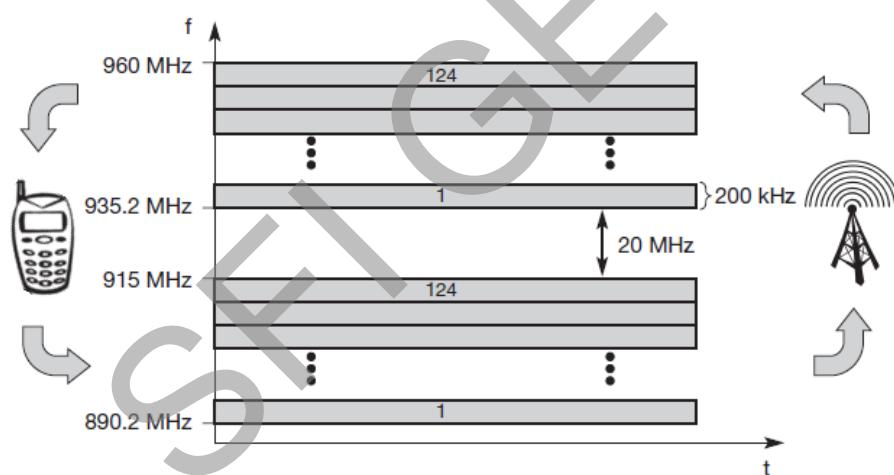


Figure 3.3
Frequency division multiplexing for multiple access and duplex

- As for example FDM and FDD, Figure 3.3 shows the situation in a mobile phone network based on the GSM standard for 900 MHz. The basic frequency allocation scheme for GSM is fixed and regulated by national authorities.
- All uplinks use the band between 890.2 and 915 MHz, all downlinks use 935.2 to 960 MHz.

- According to FDMA, the base station, shown on the right side, allocates a certain frequency for up- and downlink to establish a duplex channel with a mobile phone.
- Up- and downlink have a fixed relation.
- If the uplink frequency is $f_u = 890 \text{ MHz} + n \cdot 0.2 \text{ MHz}$, the downlink frequency is $f_d = f_u + 45 \text{ MHz}$, i.e., $f_d = 935 \text{ MHz} + n \cdot 0.2 \text{ MHz}$ for a certain channel n .
- The base station selects the channel. Each channel (uplink and downlink) has a bandwidth of 200 kHz. This illustrates the use of FDM for multiple access and duplex according to a predetermined scheme.

3. TDMA

Time division multiple access (TDMA) offers a much more flexible schemes than FDMA, which comprises all technologies that allocate certain time slots for communication.

The receiver can always stays at the same frequency. Using only one frequency, So the process is very simple. Because of that many different algorithms available to control medium access.

Listening to many channels separated in time at the same frequency is simple. Almost all MAC schemes for wired networks work according to this principle, e.g., Ethernet, Token Ring, ATM etc.

Now synchronization between sender and receiver has to be achieved in the time domain.

Allocating a certain time slot for a channel, or by using a dynamic allocation scheme.

Dynamic allocation schemes require an identification for each transmission as this is the case for typical wired MAC schemes or the transmission has to be announced beforehand. MAC addresses are quite often used as identification.

Fixed schemes do not need an identification, but are not as flexible considering varying bandwidth requirements.

However, both of the schemes can be combined with FDMA to achieve even greater flexibility and transmission capacity.

3.1 Fixed TDM

The simplest algorithm for using TDM is allocating time slots for channels in a fixed pattern. This results in a *fixed bandwidth* and is the typical solution for wireless phone systems.

MAC gave the reserved time slot at the right moment. If this synchronization is assured, each mobile station knows its turn and no interference will happen. The fixed pattern can be assigned by the base station.

Fixed access patterns fit perfectly well for connections with a fixed bandwidth. Furthermore, these patterns guarantee a fixed delay – one can transmit.

TDMA schemes with fixed access patterns are used for many digital mobile phone systems like IS-54, IS-136, GSM, DECT, PHS, and PACS.

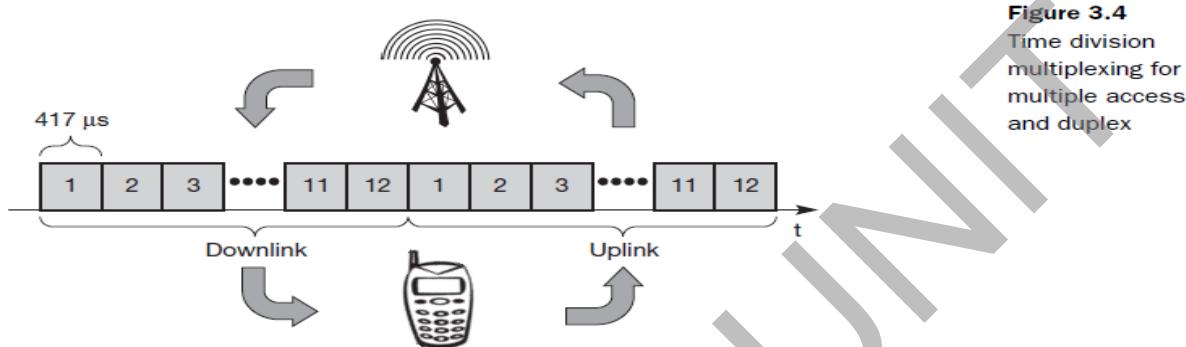
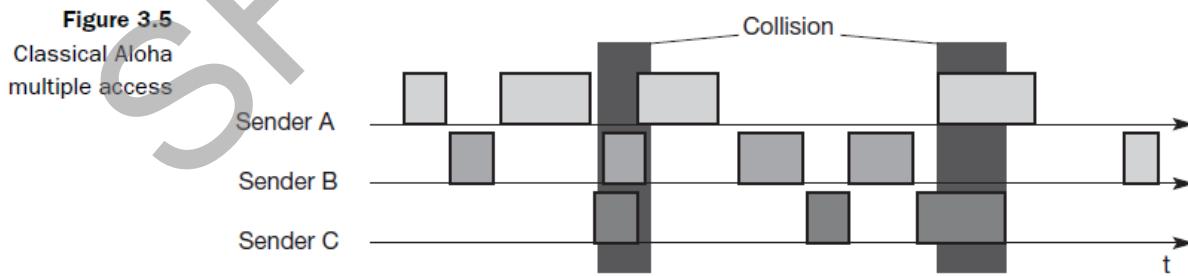


Figure 3 shows how these fixed TDM patterns are used to implement multiple access and a duplex channel between a base station and mobile station.

Assigning different slots for uplink and downlink using the same frequency is called **time division duplex (TDD)**.

In figure, the base station uses one out of 12 slots for the downlink, whereas the mobile station uses one out of 12 different slots for the uplink. Uplink and downlink are separated in time. Up to 12 different mobile stations can use the same frequency without interference using this scheme. Each connection is allotted its own up- and downlink pair.

3.2 Classical Aloha



TDM without controlling access is exactly what the classical **Aloha** scheme does.

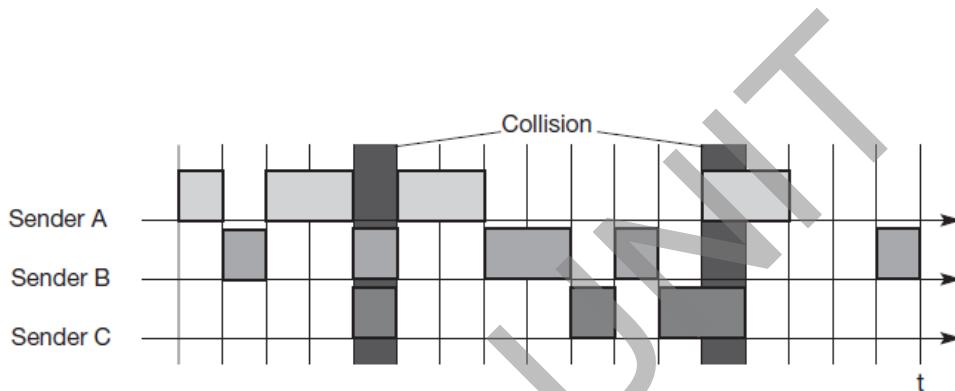
Each station can access the medium at any time as shown in Figure 3.5. This is a random access scheme, without a central arbiter controlling access and without coordination among the stations.

If two or more stations access the medium at the same time, a **collision** occurs and the transmitted data is destroyed. Resolving this problem is left to higher layers.

The simple Aloha works fine for a light load and does not require any complicated access mechanisms. On the classical assumption1 that data packet arrival follows a Poisson distribution, maximum **throughput** is achieved for an **18** per cent load.

3.3 Slotted Aloha

Figure 3.6
Slotted Aloha
multiple access



In this case, all senders have to be **synchronized**, transmission can only start at the beginning of a **time slot** as shown in Figure 3.6. Still, access is not coordinated.

Assumption stated above is, the introduction of slots raises the throughput from 18 per cent to **36 per cent**, i.e., slotting doubles the throughput.

Aloha systems work perfectly well under a light load, but they cannot give any hard transmission guarantees, such as maximum delay before accessing the medium, or minimum throughput.

Here one needs additional mechanisms, e.g., combining fixed schemes and Aloha schemes.

However, even new mobile communication systems like UMTS have to rely on slotted Aloha for medium access in certain situations (random access for initial connection set-up)

PART C (Answer all questions)

(4x3=12)

1. Describe the advantages and disadvantages of WLAN

Wireless LAN

The global goal of WLANs is to replace office cabling and additionally to introduce a higher flexibility for ad-hoc communication in, e.g., group meetings.

Advantages of Wireless LAN

Flexibility: Within radio coverage, nodes can communicate without further restriction.

Planning: Only wireless ad-hoc networks allow for communication without previous planning.

Design: Wireless networks allow for the design of small, independent devices which can be put into a pocket.

Disadvantages of wireless LAN

Quality of service: WLANs typically offer lower quality than their wired counterparts. The main reasons for this are the lower bandwidth due to limitations in radio transmission, higher error rates due to interference, and higher delay/delay variation due to extensive error correction and detection mechanisms.

Proprietary solutions: Due to slow standardization procedures, many companies have come up with proprietary solutions offering standardized functionality plus many enhanced features.

Safety and security: Using radio waves for data transmission might interfere with other high-tech equipment in, e.g., hospitals.

2. Give short notes on classification of routing algorithms and list out its types.

[1 marks for each algorithm]

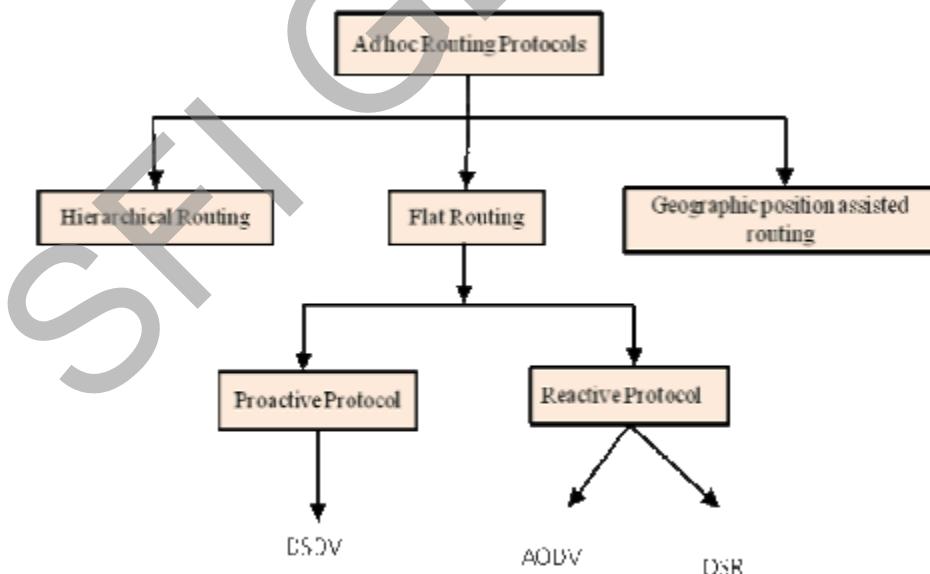


Figure 1. Classification of Routing Protocols In Mobile Ad-hoc Networks

Distance Vector Protocol

Distant vector protocol is also known as Distributed Bellman-Ford or RIP (Routing Information Protocol). Every node maintains a routing table and it contains all available destinations details, the next node to reach to destination, the number of hops to reach the destination. To maintain topology in a network nodes periodically send table to all neighbors. Following figure depicts the tables maintained by three nodes (A,B & C).

Destination Sequence Distance Vector (DSDV)

DSDV protocol is a proactive routing protocol which is a modification of conventional Bellman-Ford routing algorithm. In this protocol each nodes maintains routing table. This routing information must be periodically updated. With the help of routing information nodes can transmit data to other node in a network. The fields of routing table are as following: destination, next, metric, sequence number, installs time, stable data etc.. Sequence numbers are basically originated from destination itself which ensures loop freeness. Install time are used to delete fake entries from table. Stable data is basically a pointer to a table holding information on how stable a route is and also used to damp fluctuations in network.

Ad-hoc On-Demand Distance Vector Algorithm

AODV protocol of MANET doesn't have a fixed topology in a network. This is basically needed for wireless communication for the nodes and links are created as and when required. The sequence number of routing table is used to determine whether the routing information is up-to-date or not and also it is useful to prevent routing loop problem. To routes are created on demand, source node broadcast (RREQ) request packet to their neighbours and neighbours relay the same until it reached to its destination. Then destination node sends reply packet to source node (RREP) using the same path from which request packet come.

3. Discuss the design goals of Mobile IP.

- Mobile IP is an internet protocol designed to support host mobility.
- Mobile IP, is designed to enable mobile node (host or router) may change its location without changing their IP address.
- It solves the problem by allowing each mobile node to have two IP addresses and by transparently maintaining the binding between the two addresses.
- One of the IP addresses is the permanent home address that is assigned at the home network and is used to identify communication endpoints.
- The other is a temporary care-of address that represents the current location of the mobile node.
- Two IP addresses for mobile node

- Home address: static
- Care-of address: topologically significant address

Main Goals of Mobile IP

- To make mobility transparent to the higher level protocols
- To make minimum changes to the existing Internet infrastructure
- To provide the host stay connected to the internet regardless of their location.

Features of Mobile IP

- No geographical limitations
- No physical connection required
- Modifications to other routers and hosts is not required
- No modifications to the current IP address and IP address format Supports security

Requirements to Mobile IP

- Transparency
 - mobile end-systems keep their IP address
 - continuation of communication after interruption of link possible
 - point of connection to the fixed network can be changed
- Compatibility
 - support of the same layer 2 protocols as IP
 - no changes to current end-systems and routers required
 - mobile end-systems can communicate with fixed systems
- Security
 - authentication of all registration messages
- Efficiency and scalability
 - only little additional messages to the mobile system required
(connection typically via a low bandwidth radio link)
 - world-wide support of a large number of mobile systems in the whole Internet

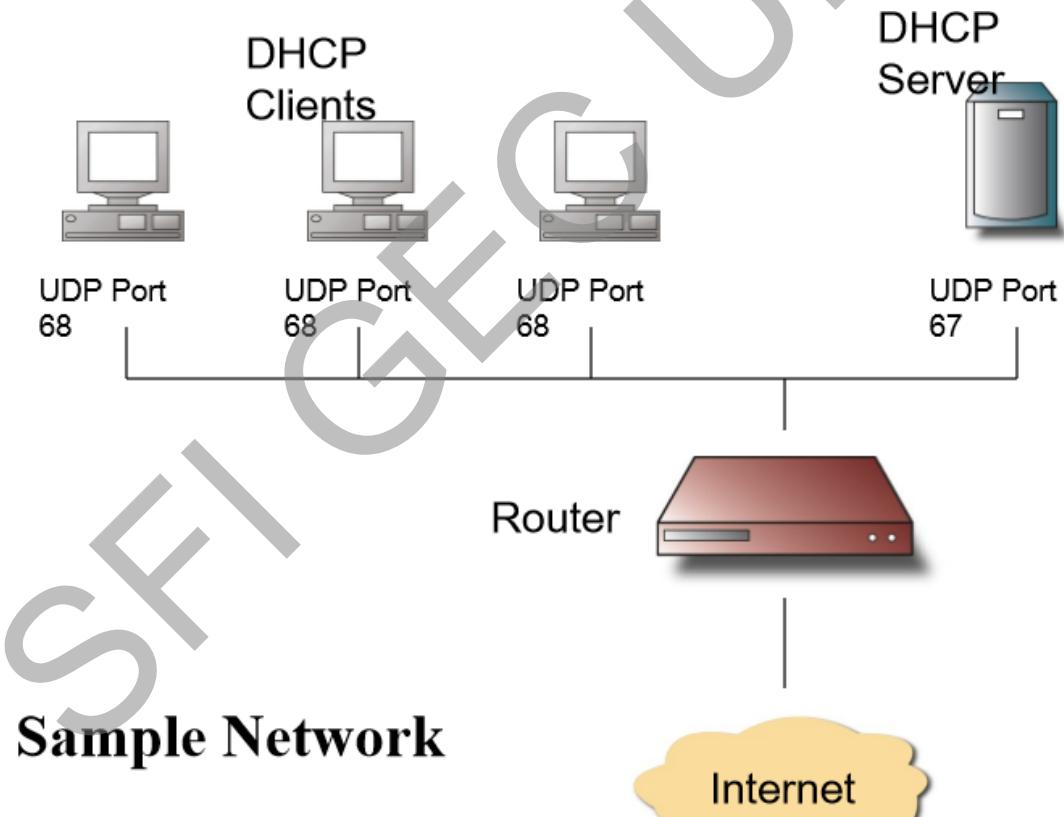
4. What is basic purpose of DHCP ? Name the entities of DHCP.

- ➔ Protocol for providing configuration parameters to host over network
- ➔ Dynamic allocation of IP address
- ➔ Minimal human Intervention

- ➔ Ip address

- ➔ Router address

- ➔ Subnet mask



PART D (Answer any two questions)

(2x9=18)

1. Explain Routing algorithms in detail

Distance Vector Routing

Distant vector protocol is also known as Distributed Bellman-Ford or RIP (Routing Information Protocol). Every node maintains a routing table and it contains all available destinations details, the next node to reach to destination, the number of hops to reach the destination. To maintain topology in a network nodes periodically send table to all neighbors. Following figure depicts the tables maintained by three nodes (A,B & C).

By using the distance vector protocols, each router over the internetwork send the neighbouring routers, the information about destination that it knows how to reach. Moreover to say the routers sends two pieces of information first, the router tells, how far it thinks the destination is and secondly, it tells in what direction (vector) to use to get to the destination. When the router receives the information from the others, it could then develop a table of destination addresses, distances and associated neighbouring routers, and from this table then select the shortest route to the destination. Using a distance vector protocol, the router simply forwards the packet to the neighbouring host (or destination) with the available shortest path in the routing table and assumes that the receiving router will know how to forward the packet beyond that point. The best example for this is the routing information protocol (RIP).[3]

Dest.	Next	Metric	...
A	A	0	
B	B	1	
C	B	2	

Dest.	Next	Metric	...
A	A	1	
B	B	0	
C	C	2	

Dest.	Next	Metric	...
A	B	3	
B	B	2	
C	C	0	

Figure 2. Distant Vector Table

The below figure represents that, how node updates their routing information by maintaining a routing table.

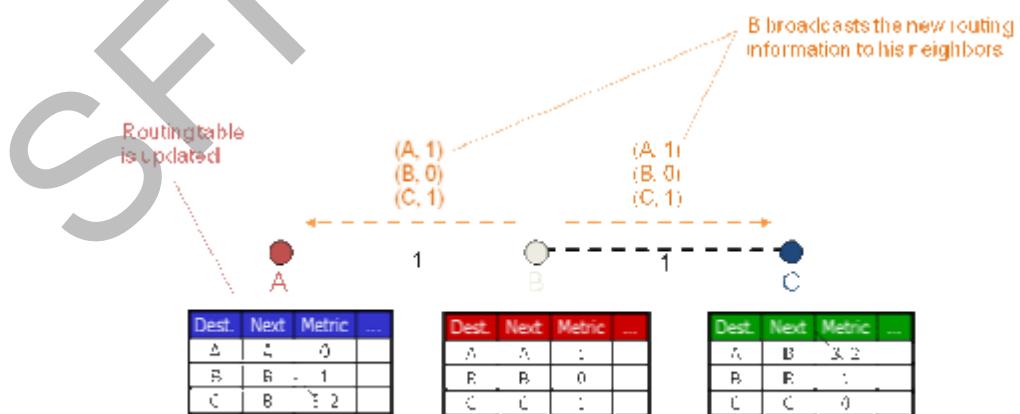


Figure 3. Updation in routing table

Destination Sequence Distance Vector algorithm

DSDV protocol is a proactive routing protocol which is a modification of conventional Bellman-Ford routing algorithm. In this protocol each node maintains routing table. This routing information must be periodically updated. With the help of routing information nodes can transmit data to other node in a network. The fields of routing table are as following: destination, next, metric, sequence number, installs time, stable data etc.. Sequence numbers are basically originated from destination itself which ensures loop freeness. Install time are used to delete fake entries from table. Stable data is basically a pointer to a table holding information on how stable a route is and also used to damp fluctuations in network.

4.1. Protocol Activities

In MANET, each node maintains and update the routing table, which are needed for the transmission of data. The table contains all the available destination and the number of hops to reach it. To maintain consistency, each node in a network transmits packet and update it periodically. The packets are being broadcasted to find out which stations are in vicinity and how many number of hops required to reach the destination. Packets may be transmitted containing the layer 2 or layer 3 address.

This protocol ensures that each node in a network continuously advertises to each of their neighbours, so that the updation of routing table has been known to all nodes and they might be in a position to find out shortest path to reach destination. Therefore, even if there is no direct link between the nodes, can exchange data. The data broadcast by each node contain new sequence number and the following information for each new route:

- The destination address
- The number of hops required to reach the destination
- The new sequence number, originally stamped by the destination

Advantages of DSDV

1. DSDV protocol guarantees loop free paths.
2. In DSDV count to infinity problem is reduced which was a major problem in Distance vector protocol.
3. Extra traffic can be avoided with incremental updates.
4. In routing table, DSDV does not maintain multiple paths to destination. A good practice in DSDV is to maintain best paths to a destination only. Because of this space consumed by routing table is reduced.

Disadvantages of DSDV

1. Because of unnecessary advertisement of routing information bandwidth is wasted.
2. DSDV doesn't support Multi path Routing.
3. It is difficult to determine a time delay for the advertisement of routes .
4. For larger network it is difficult to maintain routing table..As all nodes in a network maintains table, it leads to overhead. which consumes more bandwidth.
5. Problem of fluctuation and damping fluctuation.

Ad-hoc On-Demand Distance Vector Algorithm

AODV protocol of MANET doesn't have a fixed topology in a network. This is basically needed for wireless communication for the nodes and links are created as and when required. The sequence number of routing table is used to determine whether the routing information is up-to-date or not and also it is useful to prevent routing loop problem. To routes are created on demand, source node broadcast (RREQ) request packet to their neighbours and neighbours relay the same until it reached to its destination. Then destination node sends reply packet to source node (RREP) using the same path from which request packet come.

5.1. Working of AODV

As previously stated, paths are formed as and when required in the network therefore, each node acts as a specialised router. The routing information also maintains two separate counters: a node sequence number and a broadcast-id. When a source node wants to communicate with destination, it increments its broadcast-id and initiates path discovery by broadcasting a route request packet RREQ to its neighbours. The RREQ contains the following fields:

1. source-addr
2. source-sequence
3. dest-addr
4. dest-sequence
5. hop-cnt

The RREQ packet is identify uniquely with the help of source-addr and broadcast id pair only. Later the dynamic route entries from source to destination for all nodes are maintained. The intermediate node updates routing information and propagates new RREP only,

- If the Destination sequence number is greater, or
- If the new sequence number is same and hop count is small, or Otherwise, it just skips the new RREP. This ensures that algorithm is loop-free and only the most effective route is used .[2]

Limitations of AODV

1. Requirement on broadcast medium: The algorithm expects/requires that the nodes in the broadcast medium can detect each others' broadcasts.
2. As the route is not initially known, so the request packet travels from node to other nodes in a network to find out route information on demand. It also maintains the address of all nodes through which it passing, so the reverse path formed. Therefore the overhead on bandwidth is occurred.
3. It lacks an efficient route maintenance technique; because the routing information is always obtained on demand which also contain common traffic information, which will may not be reusable.
4. AODV lacks support for high throughput routing metrics: AODV is designed to support the shortest hop count metric. This metric favours long, low bandwidth links over short, high-bandwidth links.
5. As this protocol is reactive in nature, so until the flow is initiated, it will not discover a route. Therefore route discovery latency is high for large networks like mesh.

2. Explain snooping TCP. What are its advantages and disadvantages

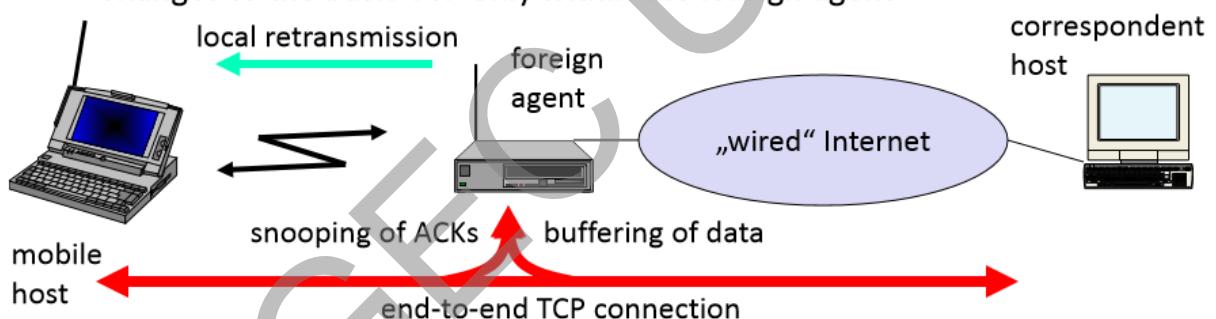
Definition -3 mark

Advantages – 3 mark

Disadvantages -3 marks

„Transparent“ extension of TCP within the foreign agent

- buffering of packets sent to the mobile host
- lost packets on the wireless link (both directions!) will be retransmitted immediately by the mobile host or foreign agent, respectively (so called “local” retransmission)
- the foreign agent therefore “snoops” the packet flow and recognizes acknowledgements in both directions, it also filters ACKs
- changes to the basic TCP only within the foreign agent



- Data transfer to the mobile host
 - FA buffers data until it receives ACK of the MH, FA detects packet loss via duplicated ACKs or time-out
 - fast retransmission possible, transparent for the fixed network
- Data transfer from the mobile host
 - FA detects packet loss on the wireless link via sequence numbers, FA answers directly with a NACK to the MH
 - MH can now retransmit data with only a very short delay
- Problems
 - snooping TCP does not isolate the wireless link as good as I-TCP
 - snooping might be useless depending on encryption schemes

Advantages of snooping TCP

- Local recovery from wireless losses
- High throughput can be achieved
- End-to-end semantics retained
- Soft state at base station
 - loss of the soft state affects performance, but not correctness

Disadvantages of snooping TCP

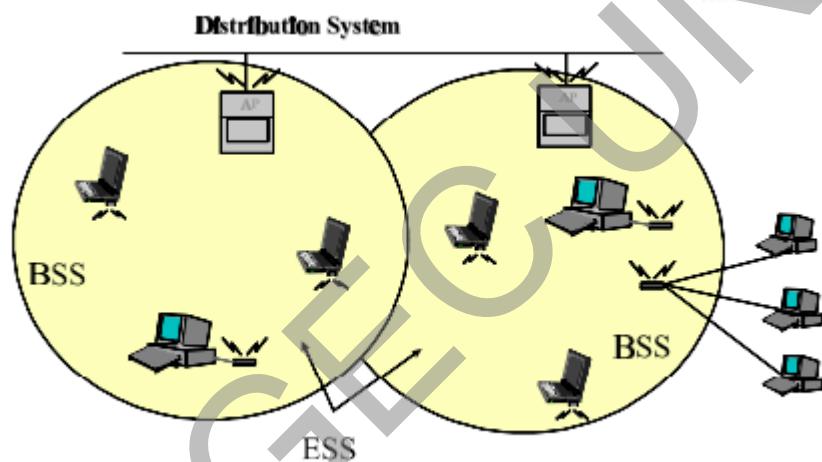
- Snooping TCP does not isolate the behavior of the wireless link as well as I-TCP.
- The quality of the isolation, which snooping TCP offers, strongly depends on the quality of the wireless link, time-out values, and further traffic characteristics.
- Not useful if TCP headers are encrypted (IPsec).

3. Draw and discuss the protocol architecture of IEEE 802.11.

Architecture – 3 marks

Explanation- 6 marks

Architecture Components An 802.11 LAN is based on a cellular architecture where the system is subdivided into cells. Each cell (called Basic Service Set, or BSS, in the 802.11 nomenclature) is controlled by a Base Station (called Access Point or, in short, AP). Although a wireless LAN may be formed by a single cell, with a single Access Point, (and as will be described later, it can also work without an Access Point), most installations will be formed by several cells, where the Access Points are connected through some kind of backbone (called Distribution System or DS). This backbone is typically Ethernet and, in some cases, is wireless itself. The whole interconnected Wireless LAN, including the different cells, their respective Access Points and the Distribution System, is seen as a single 802 network to the upper layers of the OSI model and is known in the Standard as Extended Service Set (ESS)



The standard also defines the concept of a Portal. A portal is a device that interconnects between an 802.11 and another 802 LAN. This concept is an abstract description of part of the functionality of a “translation bridge”. Even though the standard does not necessarily request it, typical installations will have the AP and the Portal on a single physical entity. This is also the case with BreezeCOM’s AP which provides both functions.

IEEE 802.11 Layers Description

As any 802.x protocol, the 802.11 protocol covers the MAC and Physical Layer. The Standard currently defines a single MAC which interacts with three PHYs (all of them running at 1 and 2 Mbit/s) as follows:

- Frequency Hopping Spread Spectrum in the 2.4 GHz Band
- Direct Sequence Spread Spectrum in the 2.4 GHz Band, and
- InfraRed

802.2			Data Link Layer
802.11 MAC			
FH	DS	IR	PHY Layer

Beyond the standard functionality usually performed by MAC Layers, the 802.11 MAC performs other functions that are typically related to upper layer protocols, such as Fragmentation, Packet Retransmissions, and Acknowledges.

PART E (Answer any four questions)

(4x10=40)

1. Explain in detail, Security issues in mobile computing with examples.

Explanation of security issues -7 marks

Examples for security issues-3 marks

The security services of ad hoc networks are not altogether different than those of other network communication paradigms. The goal is to protect the information and the resources from attacks and misbehavior. In dealing with network security, we shall explain the following requirements that an effective security paradigm must ensure: Availability: ensures that the desired network services are available whenever they are expected, in spite of attacks. Systems that ensure availability seek to combat denial of service and energy starvation attacks that we will present later. Authenticity: ensures communication from one node to another is genuine. It ensures that a malicious node cannot masquerade as a trusted network node. Data confidentiality: is a core security primitive for ad hoc networks, It ensures that a given message cannot be understood by anyone else than its (their) desired recipient(s). Data confidentiality is typically enabled by applying cryptography . Integrity: denotes the authenticity of data sent from one node to another. That is, it ensures that a message sent from node A to node B was

not modified by a malicious node, C, during transmission. If a robust confidentiality mechanism is employed, ensuring data integrity may be as simple as adding oneway hashes to encrypted messages. Non-repudiation ensures that the origin of the message is legitimate. i.e when one node receives a false message from another, nonrepudiation allows the former to accuse the later of sending the false message and enables all other nodes to know about it. Digital signature may be used to ensure nonrepudiation

. Attacks It includes any action that intentionally aims to cause any damage to the network, it can be divided according to their origins or their nature. Origin based classification splits attacks up into two categories; external and internal, whereas, nature based classification splits them up into passive attacks and active attacks External attacks: This category Includes attacks launched by a node that do not belong to the logical network, or is not allowed to access to it. Such a node penetrates the network area to launch its attack . Internal attacks: This category includes attacks launched by an internal compromised node, It is a more several kind of threat to the network since the proposed defence toward external attacks is ineffective against compromised and internal malicious nodes. Passive attacks: A passive attack is a continuous collection of information, these information would be used later when launching an active attack. That means the attacker eavesdrops packets and analyzes them to pick up required information. The security attribute that must be provided here is information confidentiality. Active attacks: Include almost all the other attacks launched by actively interacting with victims, like sleep deprivation torture that aims the batteries charges, hijacking, in which the attacker takes control of a communication between two entities and masquerades as one of them, jamming, that causes channel unavailability, attacks against routing protocols, etc... most of these attacks result in a denial of service (DoS), that is a degradation or a complete halt in communication between nodes.

Confidentiality, preventing unauthorized users from gaining access to critical information of any particular user.

- Integrity, ensures unauthorized modification, destruction or creation of information cannot take place.
- Availability, ensuring authorized users getting the access they require.
- Legitimate, ensuring that only authorized users have access to services.
- Accountability, ensuring that the users are held responsible for there securityrelated activities by arranging the user and his/her activities are linked if and when necessary.

The way these goals are achieved depends on the security policy adopted by the service providers.

2. Describe mobile transport layer in detail

Traditional TCP

- Mobility support on only lower layer is not enough to provide mobility support for applications
 - As application is directly communicates with transport Layer only.
- Congestion Control
 - TCP designed for fixed n/w with fixed end-systems.
 - Congestion may appear from time to time even in carefully designed networks.
 - Sender notices the missing ACK for the lost packet and assumes a packet loss due to congestion.
 - Retransmitting the missing packets , might only increase the congestion.
 - Solution – TCP slow down the transmission rate dramatically.

Slow start

- Sender always calculates a congestion window for a receiver.
- The start size of the congestion window is one segment.
- Double the window size after receiving ACK.
- Maintain the congestion threshold

Fast retransmit/fast recovery

- Two things lead to a reduction of the congestion threshold
 - Fast retransmit
 - continuous receiving of ACK for the same packet.
 - Fast recovery
- receipt of ACK shows that there is no congestion to justify slow start. The sender performs fast recovery from the packet loss.

Implications On Mobility

- The reason for this is of using wrong assumptions.
- Error rate on wireless links are higher as compared to wired links.
- Retransmission may increase duplicates at layer 2 and more connections are end-to-end encryption.
- Mobility itself causes packet loss.
 - TCP detects missing ACK via time-outs and concludes packet loss due to congestion control only.

Snooping TCP

- Objective – is to buffer data close to the mobile host to perform fast local retransmission in case of packet loss.
- FA/AP buffers all packets with destination mobile host and additionally 'snoop' the packet flow in both directions.
- FA not ACK data to the correspondent host.
 - FA/AP will retransmits the packet to mobile host directly from the buffer.
- Data transfer from the mobile host with destination correspondent host
 - FA snoops into packet stream to detect gaps in the seq. no. of TCP.
 - If FA detect missing packet, it return a NACK to the mobile host.
 - Now mobile host retransmit the missing packet immediately.

Mobile TCP

Dropping of packets due to a handover or higher bit error rate is not the only problem occurs.

- The occurrence of lengthy and/or frequent disconnections is another problem.
- I-TCP when mobile disconnected:
 - Has to buffer more and more data – need more buffer.
- Snooping TCP when mobile disconnected:
 - Mobile will not able to send ACK.

3. Explain how XML can be used for mobile computing.

Explanation with examples – 10 marks

- An encapsulated text— processed, displayed, or printed as per the tag
- A browser— for the presentation

Not only encapsulates the data and metadata but can represent a behaviour or set of actions

- XML document— a text with the tags
- The XML document has an extension .xml
- A tag in the document specifies the meaning of the text encapsulated within the start and corresponding end tag

A derivative of SGML (standard generalized markup language)

- Extensible— special instances of the tagbased languages can be defined such that each instance observes the fundamental rules of representing and structuring the XML document

XML Document Representation of a database

Using tags and a pair of start and end tag in the document specifies the start and end of a record in the database

- A database— used to retrieve the specific record or set of records by querying the database or by business logic transaction

Heirarchical structure in mobile component

The elements name_record, address, and telnumber

- name_record has a textual content (Raj Kamal) and two elements address (ABC Street,) and telnumber (9876543210) within it
- A document formed by XML document creating a database named Contacts

4. Describe Networks LTE Architecture & Interface in detail.

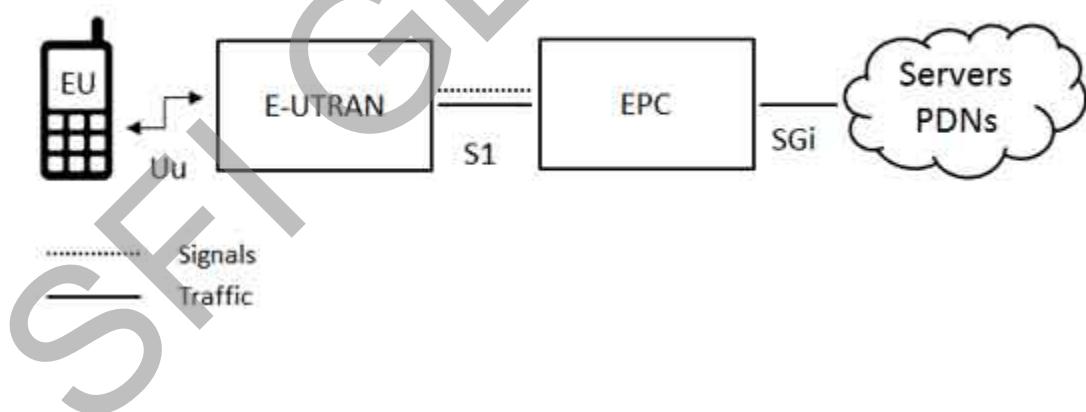
Architecture-3 marks

Explanation- 6 marks

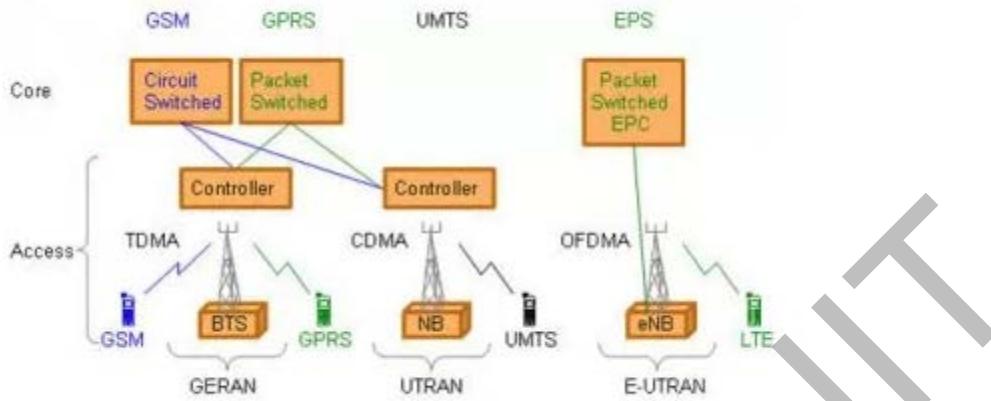
The high-level network architecture of LTE is comprised of following three main components:

- The User Equipment (UE).
- The Evolved UMTS Terrestrial Radio Access Network (E-UTRAN).
- The Evolved Packet Core (EPC).

The evolved packet core communicates with packet data networks in the outside world such as the internet, private corporate networks or the IP multimedia subsystem. The interfaces between the different parts of the system are denoted Uu, S1 and SGi as shown below:



Architecture Diagram for LTE



This diagram from standards body 3GPP shows network evolution from GSM to LTE. The core technologies have moved from (moving left to right). Meanwhile, access has evolved from TDMA (Time Division Multiple Access) to OFDMA (Orthogonal Frequency Division Multiple Access) as the need for higher data speeds and volumes as increased.

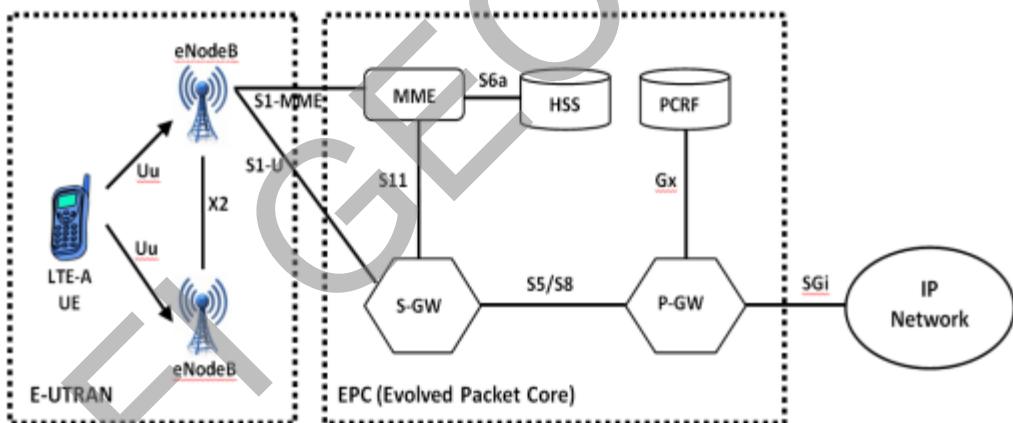


Fig.2. Diagram of LTE Network Architecture.

Radio Planning and Tools

ASSET is TEOCO's market leading radio planning tool, providing RF coverage, capacity, cell parameter and neighbor planning for wireless and mobile cellular networks. For 2016, Analysys Mason again ranked TEOCO as the largest vendor of network planning and optimization software.

A multi-technology software product for the planning of wireless networks including GSM, UMTS, LTE, Wi-Fi, small cells and other technologies in a single project. ASSET can be customized through a range of productivity packs, adding additional functionality.

Along with the existing 2G, 3G and 4G support, ASSET is now also well on its way to being a fully capable 5G radio planning tool, already supporting key initial 5G modeling capabilities such as Massive MIMO, NB-IoT and 200MHz+ carrier bandwidths at up to 60GHz frequencies. As a founding member of the 5G Innovation Centre, TEOCO has provided a number of tools to the project. ASSET is one of those tools and it is being used on a daily basis by researchers in the 5G test bed, providing us with an excellent opportunity to understand the engineering requirements of 5G network planning and expand the tools 5G capabilities.

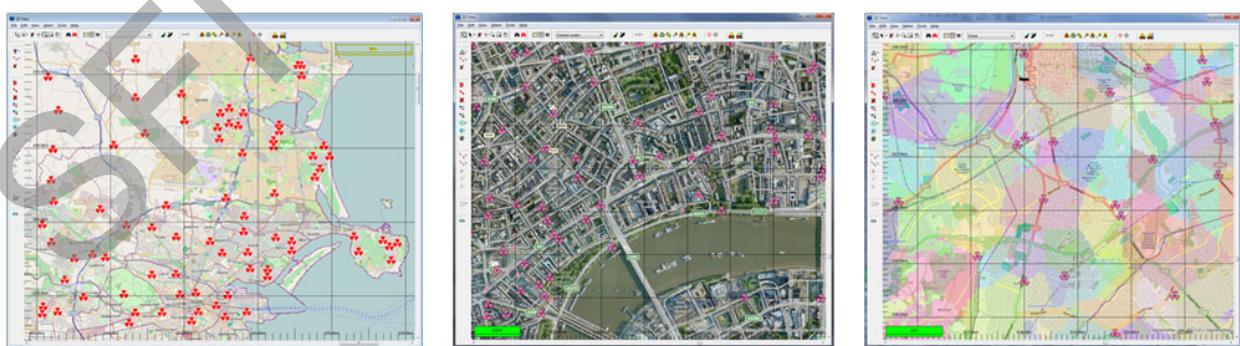
RADIO NETWORK PLANNING

ASSET is part of the TEOCO planning portfolio, a set of tightly integrated products to make the entire planning process as seamless as possible. Other products in the portfolio include ASSET Design (automated network design), ASSET Geo (geo-located traffic maps), ASSET Backhaul (backhaul planning) and ASSET Capacity (capacity planning).

Radio network planning is no longer a standalone business activity; a planning tool needs to connect to the wider OSS/BSS eco-system for maximum impact. ASSET seamlessly and reliably integrates with other systems, including:

- Inventory
- Site acquisition
- Site rollout
- Configuration management
- Performance management

This data sharing significantly improves radio network planning accuracy and drives greater business process efficiency.



Product highlights include:

MULTI-TECHNOLOGY SUPPORT

ASSET allows multiple technologies to be simultaneously modeled in a single project.

It supports GSM, UMTS and LTE (both FDD and TDD) as well as GPRS/EDGE, AMPS, TDMA, TACS, PCS, PMR/TETRA/iDEN, HSDPA, HSUPA, HSPA+, CDMA2000, EV-DO, DVB-H, Fixed WiMAX and Mobile WiMAX.

ASSET also supports emerging IoT standards including NB-IoT, LTE-M and unlicensed LPWA technologies such as LoRa and Sigfox.

5G support is well under way with capabilities such as Massive MIMO, NB-IoT, and wide bandwidth, high frequency carriers already supported and new functionality being constantly added through our collaboration with the Surrey University 5G Innovation Centre.

The ASSET database can be shared across multiple users and sites, enabling a constant view of the current network, without requiring manual synchronization.

Excellent security provides a true multi-user environment for large-scale corporate deployments and network sharing projects. Privileges may be specified to manage individual and group access at multiple levels within the tool.

GIS – MAP VIEW

At the heart of ASSET is a GIS designed specifically with the radio network planner in mind. It features a comprehensive set of display and layering functionality, including the ability to display Web Maps and to edit cell parameters directly from the GIS.

PROPAGATION MODELING

ASSET includes a number of empirical and deterministic propagation models, as well as the option to add 3rd party models from providers such as Siradel, Wavecall and Altair(previously AWE Communications). Models can be manually or automatically tuned, and their accuracy can be improved by using measurement data.

COVERAGE MAP GENERATION AND PRINTING

By combining ASSET and ARRAYWIZARD, you can schedule and produce nationwide coverage plots and statistical reports. These can then be published over the Web or printed.

TRAFFIC MODELING

Traffic modeling with ASSET allows you to forecast traffic and analyze capacity, and to model new services prior to implementation. Running ‘What-if?’ scenarios ensures the best design is deployed.

NEIGHBOR PLANNING

ASSET features the most powerful neighbor planner on the market. Neighbor relations between cells, including IRAT handovers, can be planned and analyzed. Neighbor relation parameter planning is also included.

MEASUREMENT TOOLBOX

Network measurements from multiple sources, such as drive tests, carrier wave measurements and mobile measurement reports, can be imported into ASSET to tune propagation models and enhance neighbor planning.

INTEGRATION

A modern approach to integration, based on SOA-compliant web services, provides ASSET with the ability to easily integrate into large-scale, multi-application environments.

5. Explain in detail about protocol architecture of WAP.

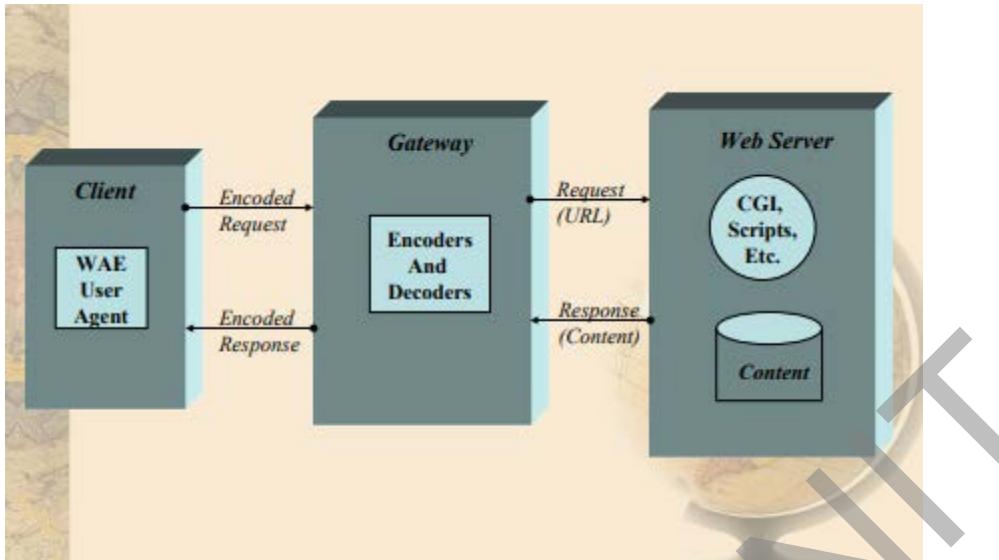
Explanation -7 marks

Architecture- 3 marks

Wireless Application Protocol commonly known as WAP is used to enable the access of internet in the mobile phones or PDAs. An open, global specification that empowers mobile users with wireless devices to easily access and interact with internet information and services instantly.

- WAP stands for Wireless Application Protocol
- WAP is an application communication protocol
- WAP is used to access services and information WAP is for handheld devices such as mobile phones
- WAP enables the creating of web— applications for mobile devices.
- WAP uses the mark-up language WML (not— HTML) WML is defined as an XML 1.0 application

WAP Architecture



- Leverage existing standards whenever possible Define a layered and extensible architecture
- Support as many wireless networks as possible
- Provide support for secure applications and communication
- Optimize for efficient use of device resources
- WAP DEVICE - Is used to access WAP applications and content. It might be a PDA, handheld computer.
- WAP CLIENT - Entity that receives content from Internet via a WAP Gateway. This is usually the WAP Browser.
- WAP CONTENT/ORIGIN/APPLICATION SERVER - Element in the network where the information or web/WAP applications resides.
-

6. Give a short note about [2 marks for each]

- **Bluetooth**
a standard for the short-range wireless interconnection of mobile phones, computers, and other electronic devices.
Radio receivers and television sets pick up programs beamed in radio waves hundreds (possibly even thousands) of km/miles through the air. Cordless telephones use similar technologies to carry calls from a handset to a base station somewhere in your home. If you use Wi-Fi (wireless Internet), your computer sends and receives a steady stream of Internet data to and from a router that's probably wired directly to the Net. All these technologies

involve sending information back and forth not along copper cables but in radio waves buzzing invisibly through the air.

➤ **Palm OS**

Also known as Garnet OS is a discontinued mobile operating system initially developed by Palm, Inc., for personal digital assistants (PDAs) in 1996. Palm OS was designed for ease of use with a touchscreen-based graphical user interface. ... Several other licensees have manufactured devices powered by Palm OS.

➤ **J2ME**

J2ME (Java 2 Platform, Micro Edition) is a technology that allows programmers to use the Java programming language and related tools to develop programs for mobile wireless information devices such as cellular phones and personal digital assistants (PDAs).

J2ME (Java 2 Platform, Micro Edition) is a technology that allows programmers to use the Java programming language and related tools to develop programs for mobile wireless information devices such as cellular phones and personal digital assistants (PDAs). J2ME consists of programming specifications and a special virtual machine, the K Virtual Machine, that allows a J2ME-encoded program to run in the mobile device. There are two programming specifications: Connected, Limited Device Configuration (CLDC) and the Mobile Information Device Profile (MIDP). CLDC lays out the application program interface (API) and virtual machine features needed to support mobile devices. MIDP adds to the CLDC the user interface, networking, and messaging details needed to interface with mobile devices. MIDP includes the idea of a midlet, a small Java application similar to an applet but one that conforms with CLDC and MIDP and is intended for mobile devices.

Devices with systems that exploit J2ME are already available and are expected to become even more available in the next few years.

➤ **JavaCard**

Java Card refers to a software technology that allows Java-based applications (applets) to be run securely on smart cards and similar small memory footprint devices. Java Card is the tiniest of Java platforms targeted for embedded devices. It is widely used in SIM cards (used in GSM mobile phones) and ATM cards.

Java Card technology was originally developed for the purpose of securing sensitive information stored on smart cards. Security is determined by various aspects of this technology:

Data encapsulation

Data is stored within the application, and Java Card applications are executed in an isolated environment (the Java Card VM), separate from the underlying operating system and hardware.

Applet Firewall

Unlike other Java VMs, a Java Card VM usually manages several applications, each one controlling sensitive data. Different applications are therefore separated from each other by an applet firewall which restricts and checks access of data elements of one applet to another.

Cryptography

Commonly used symmetric key algorithms like DES, Triple DES, AES, and asymmetric key algorithms such as RSA, elliptic curve cryptography are supported as well as other cryptographic services like signing, key generation and key exchange.

Applet

The applet is a state machine which processes only incoming command requests and responds by sending data or response status words back to the interface device.

➤ **XML**

- An encapsulated text— processed, displayed, or printed as per the tag
- A browser for the presentation
- Not only encapsulates the data and metadata but can represent a behaviour or set of actions
- XML document a text with the tags
- The XML document has an extension .xml
- A tag in the document specifies the meaning of the text encapsulated within the start and corresponding end tag
- A derivative of SGML (standard generalized markup language)
- Extensible
- special instances of the tagbased languages can be defined such that each instance observes the fundamental rules of representing and structuring the XML document

Reg No.....

Name.....

**SIXTH SEMESTER B.TECH DEGREE MODEL EXAMINATION, MARCH
2018**

**COURSE CODE: CS368
COURSE NAME: Web Technologies**

Max Marks:100

Duration: 3 Hours

Part A (Answer all questions. 3 Marks each)

1. Explain img tag, what are the two required attributes img elements?
2. How are scroll bars specified for <textarea> controls?
3. What is the form of an IP address?
4. What is a plug-in?

Part B (Answer any 2 questions. 9 Marks each)

5. Write short notes on
 - a. Web server
 - b. Web browser
 - c. CGI
 - d. Servlet
 - e. Applet
 - f. XML
6. Create an HTML document containing an ordered list of three items—ice cream, softserve and frozen yogurt. Each ordered list should contain a nested, unordered list of your favoriteflavors. Provide three flavors in each unordered list.
7. Create a website registration form to obtain a user's first name, last name and e-mail address. In addition, include an optional survey question that asks the user's year in college (First, second third and final year,). Add a number element that the user can enter gpa.

Part C (Answer all questions. 3 Marks each)

8. List the advantages of embedded style sheets over inline method.
9. What are the possible values of the list-style-type property when it is used with ordered lists?
10. What are the five primitive data types in JavaScript?
11. Describe the semantics of the join method of Array.

Part D (Answer any 2 questions. 9 Marks each)

12. Describe the parameters and actions of the setTimeout function with suitable example.
13. Create an HTML document using JavaScript, to display an image and one button. The button should be labeled simply “next...” When pressed, the content of the image should change to a different image.

14. Create an HTML document that includes at least two images and enough text to precede the images, flow around them (one on the left and one on the right), and continue after the last image.

Part E (Answer any 4 questions. 10 Marks each)

15. Explain XML document structure.
16. Create an XML document for a catalog of cars, where each car has the child elements make, model, year, color, engine, number_of_doors, transmission_type, and accessories. The engine element has the child elements number_of_cylinders and fuel_system (carbureted or fuel injected). The accessories element has the attributes radio, air_conditioning, power_windows, power_steering, and power_brakes, each of which is required and has the possible values yes and no.
17. Write an HTML document that includes an anchor tag that calls a PHP document. Also, write the called PHP script document, which returns a randomly chosen greeting from a list of five different greetings. The greetings must be stored as constant strings in the script.
18. How can a script determine whether a particular cookie exists?
19. What is the difference between the sort and asort functions?
20. Create an HTML form to pass form data to PHP script and print the data through the script.

**SIXTH SEMESTER B.TECH DEGREE MODEL EXAMINATION, MARCH
2018 ANSWER KEY**

**COURSE CODE: CS368
COURSE NAME: Web Technologies**

Max Marks:100

Duration: 3 Hours

Part A (Answer all questions. 3 Marks each)

1. The `` tag defines an image in an HTML page. The `` tag has two required attributes: `src` and `alt`. Images are not technically inserted into an HTML page, images are linked to HTML pages. The `` tag creates a holding space for the referenced image. (With example)
2. The `<textarea>` tag defines a multi-line text input control. A text area can hold an unlimited number of characters, and the text renders in a fixed-width font. The size of a text area can be specified by the `cols` and `rows` attributes. Use style `overflow-y: scroll;`
3. IP address is short for Internet Protocol (IP) address. An IP address is an identifier for a computer or device on a TCP/IP network. The format of an IP address is a 32-bit numeric address written as four numbers separated by periods. Known as dotted decimal notation. Each number can be zero to 255. For example, 192.168.10.240 could be an IP address.
4. Plug-in (or plugin, add-in, addin, add-on, addon, or extension) is a software component that adds a specific feature to an existing computer program. When a program supports plug-ins, it enables customization. The common examples are the plug-ins used in web browsers to add new features such as search-engines, virus scanners, or the ability to use a new file type such as a new video format. Well-known browser plug-ins includes the Adobe Flash Player, the QuickTime Player, and the Java plug-in, which can launch a user-activated Java applet on a web page to its execution on a local Java virtual machine.

Part B (Answer any 2 questions. 9 Marks each)

5. Write short notes on
 - a. Web servers are programs that provide documents to requesting browsers. Servers are slave programs: They act only when requests are made to them by browsers running on other computers on the Internet. The most commonly used Web servers are Apache, which has been implemented for a variety of computer platforms, and Microsoft's Internet Information Server (IIS), which runs under Windows operating systems
 - b. When two computers communicate over some network, in many cases one acts as a client and the other as a server. The client initiates the communication, which is often a request for information stored on the server, which then sends that information back to the client. The Web, as well as many other systems, operates

in this client-server configuration. Documents provided by servers on the Web are requested by browsers, which are programs running on client machines. They are called browsers because they allow the user to browse the resources available on servers. Example: Mozilla Firefox, Internet Explorer, Google Chrome

- c. The common gateway interface (CGI) is a standard way for a Web server to pass a Web user's request to an application program and to receive data back to forward to the user. When the user requests a Web page (for example, by clicking on a highlighted word or entering a Web site address), the server sends back the requested page. However, when a user fills out a form on a Web page and sends it in, it usually needs to be processed by an application program. The Web server typically passes the form information to a small application program that processes the data and may send back a confirmation message. This method or convention for passing data back and forth between the server and the application is called the common gateway interface (CGI).
 - d. A servlet is a compiled Java class, an object of which is executed on the server system when requested by the HTML document being displayed by the browser. A servlet produces an HTML document as responses, some parts of which is static and are generated by simple output statements, while other parts are created dynamically when the servlet is called. When an HTTP request is received by a Web server, it examines the request. If a servlet must be called, the Web server passes the request to the servlet processor, called a servlet container. The servlet container determines which servlet must be executed, makes sure that it is loaded, and calls it.
 - e. A Java applet is a small application that is written in the Java programming language, or another programming language that compiles to Java bytecode, and delivered to users in the form of Java bytecode. The user launches the Java applet from a web page, and the applet is then executed within a Java virtual machine
 - f. eXtensible Markup Language (XML) is designed to describe data and it has strict syntax rules. The XML specification requires that XML processors not accept XML documents with any errors.
6. <html><body>
<h1>My Favorite Flavors</h1>

Ice Cream

 Chocolate
 Moose Tracks
 Vanilla

Soft Serve

 Twist
 Chocolate
 Vanilla


```

<li>Frozen Yogurt</li>
    <ul>
        <li>Cherry Chocolate</li>
        <li>Vanilla</li>
        <li>Blackberry</li>
    </ul>
</ol>
</body>
</html>
7. <form action="action_page.php" method="get" autocomplete="on">
<p>
<label> First Name: <input type="text" > </label>
</p>
<p> <label> Last Name: <input type="text" > </label> <br> </p>
<p> <label> Email: <input type="email" > </label> <br> </p>
<p> <label> Year: <input type="text" > </p>
<select name="year">
    <option value="First Year">
    <option value="Second Year">
    <option value="Third Year">
    <option value="Final Year">
</select> </label>
<label>Optional Question: </label> <label> What is your GPA?
<input type="number" step = 0.10 min = 0.0 max =10> </label> </details> </form>.

```

Part C (Answer all questions. 3 Marks each)

8. Inline style sheets apply to the content of a single HTML element, document-level style sheets apply to the whole body of a document, and external style sheets can apply to the bodies of any number of documents. Inline style sheets have precedence over document style sheets, which have precedence over external style sheets. Document-level style specifications appear in the document head section and apply to the entire body of the document. This is obviously an effective way to impose a uniform style on the presentation of all the content of a single document.
9. The list-style-type property for ordered list can be used to specify the types of sequence values.

Property Value	Sequence Type
decimal	Arabic numerals starting with 1
decimal-leading-zero	Arabic numerals starting with 0
lower-alpha	Lowercase letters
upper-alpha	Uppercase letters
lower-roman	Lowercase Roman numerals
upper-roman	Uppercase Roman numerals
lower-greek	Lowercase Greek letters
lower-latin	Same as lower-alpha

upper-latin	Same as upper-alpha
armenian	Traditional Armenian numbering
georgian	Traditional Georgian numbering
Example: <style type = "text/css">	
ol {list-style-type: upper-roman;} </style>	

10. Number, String, Boolean, Undefined, and Null. The Number type values are represented internally in double-precision floating-point form. A string literal is a sequence of zero or more characters delimited by either single quotes ('') or double quotes (""). Null is the reserved word null, which indicates no value. The only value of type Undefined is undefined. Unlike null, there is no reserved word undefined. If a variable has been explicitly declared, but not assigned a value, it has the value undefined. The only values of type Boolean are true and false. These values are usually computed as the result of evaluating a relational or Boolean expression
11. The join method converts all the elements of an array to strings and concatenates them into a single string. If no parameter is provided to join, the values in the new string are separated by commas. If a string parameter is provided, it is used as the element separator.

Consider the following example:

```
var names = new Array["Mary", "Murray", "Murphy", "Max"];
```

```
var name_string = names.join(" : ");
```

The value of name_string is now "Mary : Murray : Murphy : Max".

Part D (Answer any 2 questions. 9 Marks each)

12. The setTimeout method takes two parameters: a string of JavaScript code to be executed and a number of milliseconds of delay before executing the given code.
Example, the call setTimeout("mover()", 20); causes a 20-millisecond delay, after which the function mover is called.

*Program required

13. <!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script>
var dd = 5;
var cnt = 0;
function change() {

var rotator = document.getElementById('rr'); // change to match image ID
var imageDir = 'Images/'; // change to match images folder

var images = ['2.jpg', '3.jpg', '4.jpg', '5.jpg'];

for (var i = 0; i < images.length; i++) {
if (dd == 5) {
rotator.src = imageDir + images[i];
dd = 0;
}
dd++;
}
}

</script>
</head>
<body>

<button onclick="change();">Change Image</button>
</body>
</html>

```
cnt++;

var len = images.length;
if (cnt < dd) {
    rr.src = imageDir + images[cnt];
}
else if (cnt == len) {
    rr.src = imageDir + images[0];
    cnt = 0;
}
}

</script>

</head>
<body>


    <input type="button" value=" next . ." onclick="change()" />
</body>
</html>
```

14.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>Baseball Icons</title>
<style type="text/css">

body {
    background: #EEF;
    padding: 20px 0;
}
#wrapper {
    margin: 0 auto;
    width: 600px;
    background: #EEF;
    overflow: hidden; /* float containment for modern browsers */
}
.left {float: left; margin: 5px 10px 5px 0;}
.right {float: right; margin: 5px 0 5px 10px;}

img /*for testing without images (remove this entire syntax on final page)*/
display: block;
width: 80px;
```

```
height:100px;
background:red;
}
</style>

</head>
<body>

<div id="wrapper">
    <h2>Baseball Icons</h2>
    <p>
        Hank Aaron is one of baseball's legendary icons. He was born on February 5, 1934 in Mobile, Alabama. He is a retired baseball player.
         His career consisted of the years of 1954 through 1976. Hank was considered by many people as one of the greatest baseball players of all time. Hank played with the Indianapolis Clowns of the Negro American League.
    </p>
    <p>
        Aaron started his major league baseball career in 1954. He played 21 seasons with the Milwaukee and Atlanta Braves. His greatest achievement was setting the MLB record for most career home runs. He hit 24 or more home runs every year from 1955 through 1973. Hank Aaron made the All-Star team every year from 1955 until 1976. He won 3 Rawlings Gold Glove Awards. In 1957, he won the National League MVP.
        The Braves won the World Series that same year. That would be his one and only world series won.
    </p>
    
    <p>
        Lou was an baseball player in the 1920s and 1930s, most remembered for his prowess as a hitter, his consecutive games-played record. At the age of 36, when he was stricken with a fatal neurological disease.
    </p>
    <p>
        Gehrig set many major league records. He holds the record for most career grand slams with 23. Gehrig was elected to the Baseball Hall of Fame in 1939. In 1969 he was voted the greatest first baseman of all time by the Baseball Writers' Association. Gehrig set many major league records. He holds the record for most career grand slams with 23.
        Gehrig was elected to the Baseball Hall of Fame in 1939. In 1969 he was voted the greatest first baseman of all time by the Baseball Writers' Association.
    </p>
</div>
</body>
</html>
```

Part E (Answer any 4 questions. 10 Marks each)

15. The syntax of XML can be thought of at two distinct levels.

1. General low-level syntax of XML, which imposes its rules on all XML documents.
2. The other syntactic level is specified by XML schemas.

XML schemas specify the set of elements and attributes that can appear in a particular document or collection of documents and also the orders and arrangements in which they can appear. An XML schema can be used to define an XML tag set. The most common are the data elements of the document. XML documents may also include markup declarations, which are instructions to the XML parser, and processing instructions, which are instructions for an application program that will process the data described in the document.

All XML documents begin with an XML declaration, which looks like a processing instruction but technically is not one. The XML declaration identifies the document as XML and provides the version number of the XML standard used. It may also specify an encoding standard. Comments in XML are the same as in HTML. They cannot contain two adjacent dashes, for obvious reasons. XML names are used to name elements and attributes.

An XML name must begin with a letter or an underscore and can include digits, hyphens, and periods. XML names are case sensitive, so Body, body, and BODY are all distinct names. There is no length limitation for XML names. A small set of syntax rules applies to all XML documents. XHTML uses the same rules, and the HTML markup in this book complies with them. Every XML document defines a single root element, whose opening tag must appear on the first line of XML code. All other elements of an XML document must be nested inside the root element. The root element of every HTML document is html, but in XML it has whatever name the author chooses. XML tags, like those of HTML, are surrounded by angle brackets.

Every XML element that can have content must have a closing tag. Elements that do not include content must use a tag with the following form: <element_name />

As is the case with HTML, XML tags can have attributes, which are specified with name-value assignments. All attribute values must be enclosed by either single or double quotation marks. An XML document that strictly adheres to these syntax rules is considered *well formed*.

The following is a simple, but complete, example:

```
<?xml version = "1.0" encoding = "utf-8"?>
<ad>
  <year> 1960 </year>
  <make> Cessna </make>
  <model> Centurian </model>
  <color> Yellow with white trim </color>
  <location>
    <city> Gulfport </city>
    <state> Mississippi </state>
  </location>
</ad>
```

```
16.<?xml version = "1.0" encoding = "utf-8"?>
<!-- cars.xml - A solution -->
<!DOCTYPE car_catalog SYSTEM "cars.dtd">
<?xml-stylesheet type = "text/css" href = "cars.css"?>
<car_catalog>
  <car>
    <year> 1997 </year>
    <make> &c; </make>
    <model> Impala </model>
    <color> Light blue </color>
    <engine>
      <number_of_cylinders>8cylinder</number_of_cylinders>
      <fuel_system> multi-port fuel injected </fuel_system>
    </engine>
    <number_of_doors> 4 door </number_of_doors>
    <transmission_type> 4 speed automatic </transmission_type>
    <accessories radio = "yes" air_conditioning = "yes" power_windows =
    "yes" power_steering = "yes" power_brakes = "yes" />
  </car>
```

17. HTML Document

```
<html>
<head>
  <title>
    Random Greetings
  </title>
</head>
<body>
  <h1>
    Random Greetings
  </h1>
  <a href=" http://localhost/prog5.php "> Click Here</a>
</body>
</html>
```

PHP Script

```
<?php
$stor1="Good Morning";
$stor2="Good Afternoon";
$stor3="Good Evening";
$stor4="Good Night";
$stor5="Good Bye";
$number=mt_rand(0,4); //mt_rand() generates random numbers
if($number==0)
  print $stor1;
else if($number==1)
  print $stor2;
else if($number==2)
  print $stor3;
```

```

else if($number==3)
    print $stor4;
else if($number==4)
    print $stor5;
?>
```

18. A **cookie** is a small object of information that includes a name and a textual value. A cookie is created by some software system on the server. Every Hypertext Transfer Protocol (HTTP) communication between a browser and a server includes a header, which stores information about the message. The header part of an HTTP communication can include cookies. So every request sent from a browser to a server, and every response from a server to a browser, can include one or more cookies. At the time it is created, a cookie is assigned a lifetime. When the time a cookie has existed reaches its associated lifetime, the cookie is deleted from the browser's host machine. Every browser request includes all the cookies its host machine has stored that are associated with the Web server to which the request is directed. Only the server that created a cookie can ever receive the cookie from the browser, so a particular cookie is information that is exchanged exclusively between one specific browser and one specific server.

A cookie is set in PHP with the `setcookie` function. This function takes one or more parameters. The first parameter, which is mandatory, is the cookie's name given as a string. The second, if present, is the new value for the `cookie`, also a string. If the value is absent, `setcookie` undefines the cookie. The third parameter, when present, is the expiration time in seconds for the cookie, given as an integer.

```
setcookie("voted", "true", time() + 86400);
```

This call creates a cookie named "voted" whose value is "true" and whose lifetime is one day (86,400 is the number of seconds in a day).

The other cookie operation is getting the cookies and their values from subsequent browser requests. In PHP, cookie values are treated much, as are form values. All cookies that arrive with a request are placed in the implicit `$_COOKIE` array, which has the cookie names as keys and the cookie values as values. A PHP script can test whether a cookie came with a request by using the `IsSet` predicate function on the associated variable.

Once the cookies have been set, they can be accessed on the next page load with the `$_COOKIE` or `$HTTP_COOKIE_VARS` arrays. This is because cookies are sent in response headers to the browser and the browser must then send them back with the next request. This is why they are only available on the second page load.

```

<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
```

```
echo "Cookie named " . $cookie_name . "" is not set!";
} else {
    echo "Cookie " . $cookie_name . "" is set!<br>;
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

19. PHP have sort and asort built in array functions. From the name itself, both will sort an array elements.

sort() function will sort an array by values and array keys will be automatically reset.
asort() function will sort an array by values and array keys will be the same as per original array.

```
<?php

$fruits = array("lemon", "orange", "banana", "apple");
sort($fruits);
foreach ($fruits as $key => $val) {
    echo "fruits[" . $key . "] = " . $val . "\n";
}

?>
```

Output

```
fruits[0] = apple
fruits[1] = banana
fruits[2] = lemon
fruits[3] = orange
```

```
<?php
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");
asort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val\n";
}
?>
output:
c = apple
b = banana
d = lemon
a = orange
```

```
20. <html>
<head>
<title>Process the HTML form data with the POST method</title>
</head>
<body>
<form name="myform" action="process.php" method="POST">
<input type="hidden" name="check_submit" value="1" />
Name: <input type="text" name="Name" /><br />
Password: <input type="password" name="Password" maxlength="10" /><br />
Select something from the list: <select name="Seasons">
<option value="Spring" selected="selected">Spring</option>
<option value="Summer">Summer</option>
<option value="Autumn">Autumn</option>
<option value="Winter">Winter</option>
</select><br /><br />
Choose one:
<input type="radio" name="Country" value="USA" /> USA
<input type="radio" name="Country" value="Canada" /> Canada
<input type="radio" name="Country" value="Other" /> Other
<br />
Choose the colors:
<input type="checkbox" name="Colors" value="green" checked="checked" /> Green
<input type="checkbox" name="Colors" value="yellow" /> Yellow
<input type="checkbox" name="Colors" value="red" /> Red
<input type="checkbox" name="Colors" value="gray" /> Gray
<br /><br />
Comments:<br />
<textarea name="Comments" rows="10" cols="60">Enter your comments here</textarea><br />
<input type="submit" />
</form>
</body>
</head>
</html>
```

```
process.php
<?php
//Check whether the form has been submitted
if (array_key_exists('check_submit', $_POST)) {
    //Converts the new line characters (\n) in the text area into HTML line breaks (the <br />
    tag)
    $_POST['Comments'] = nl2br($_POST['Comments']);
    //Check whether a $_GET['Languages'] is set
    if ( isset($_POST['Colors']) ) {
        $_POST['Colors'] = implode(', ', $_POST['Colors']); //Converts an array into a single
        string
    }

    //Let's now print out the received values in the browser
    echo "Your name: {$_POST['Name']}<br />";
    echo "Your password: {$_POST['Password']}<br />";
    echo "Your favourite season: {$_POST['Seasons']}<br /><br />";
    echo "Your comments:<br />{$_POST['Comments']}<br /><br />";
    echo "You are from: {$_POST['Country']}<br />";
    echo "Colors you chose: {$_POST['Colors']}<br />";
} else {
    echo "You can't see this page without submitting the form.";
}
?>
```

Reg No _____

Name _____

**FIFTH SEMESTER B.TECH DEGREE EXAMINATION MODEL TEST
QUESTION PAPER, March 2018**

CS 368: WEB TECHNOLOGY

Max. Marks: 100

Duration: 3 Hours

PART A

Answer all questions. Each carries 3 marks

1. What are the stages and strategies required to develop a web project ?
2. Explain the concept of url and ip address? How an ip address is assigned to host
3. List the difference between html and xhtml?
4. List the rule used to write xml

PART B

Answer any Two questions. Each carries 9 marks

5.
 - a. When you download a file using HTTP and FTP, which one is normally used why?
 - b. Define the terms Web site, Web page, Web server, LIRL and home page.
- 6 (a) Create a web page using HTML having frames as follows :
 - a. Your Name
 - b. Punch line etc.
 - c. Objective
 - d. Personal Information
 - e. Family Information
 - f. Educational Information
 - g. Display information here of selected link
 - h. Display relevant
 - i. im6ges here
 - j. Experience
 - k. Achievements
 - l. The frame which includes Objective, Personal Information, etc., are the hyperlinks. Display the relevant information in the next fraine as selecting the link. The colour scheme of hyperlinks should be as follows :default - green; active- red; visited-blue
7. What is CGI ? How does CGI works ? Give an example program using CGI to retrieve the user information (name, age, mobile.no) from the HTML

PART C

Answer all questions. Each carries 3 marks

8. What do you mean by CSS write down the different levels of CSS used in websites

9 How do we handle in JAVAscript ? What is DHTML ?

10 What is the difference between Java and Jave Script

11. Write short notes on : (i) VB Script (ii) AJAX

PART D

Answer any Two questions. Each carries 9 marks

12 .What do you mean by JSP? Explain the Architecture of JSP How JSP provides better performance.

13 What is the difference between java and java script? How is java strongly associated with internet? Draw a flow chart to show how various java tools are used in application development

14 Explain JavaScript HTML DOM in detail

PART E

Answer any four questions. Each carries 10 marks

15What is XML. Write down the syntax of XML

16Discuss XML Documents with CSS, XSLT Style Sheet.

17Explain JSON .Compare with XML.

18Explain PHP in detail

19Discuss Cookies and Session tracking in detail.

20How to handle forms in PHP

Reg No _____

Name _____

**FIFTH SEMESTER B.TECH DEGREE EXAMINATION MODEL TEST
QUESTION PAPER, March 2018**

CS 368: WEB TECHNOLOGY

Max. Marks: 100

Duration: 3 Hours

PART A

1.project summary: Outlines the general overview of the project, organizational background, the environment the organization exists in, the people the organization serves and the unique value it provides to its audience.

Goals: What are two or three specific measurable goals that the site should achieve? Clear goals allow the Web team the ability to focus on what will provide the most impact and move the organization forward.

Target audiences: Who will help the organization achieve its stated goals? Most organizations speak to multiple organizations (such as customers, stakeholders, internal audience, suppliers, partners, shareholders and/or government institutions). Audience profiles include demographics, psychographics, brand perceptions, audience needs, online goals and tasks routinely performed.

Messages: What are the key messages that attract and motivate key audiences to engage with the organization? What are the key brand messages that help differentiate the organization from its peers?

Competition: Who are rival organizations that provide similar offerings to your audience? Include an overview of competitive organizations' Web sites, considering visual branding, messaging, navigation, calls to action and key differentiators.

Project Scope

Defining the scope of the project is a critical step. One of the most common frustrations with Web projects is scope creep. By creating a well-defined project scope plan that outlines specific activities and deliverables, along with specific timelines, you will be able to clearly set expectations for your clients. One of the most common ways of tracking Web projects is through the use of a Gantt chart. A Gantt chart not only outlines major activities but also the tasks associated with each activity and start and end dates. The Gantt chart provides a visual reference for the team, showing the timeframe of each step and the dependencies between steps. The Gantt chart also creates accountability between the Web team and the client (which could be an outside client or simply your boss), letting the client and the team know

that the delivery schedule is dependent on everyone hitting their marks; if someone misses a date by a day, the schedule shifts by a day.

Wireframes and Site Architecture

Site architecture includes the sitemap and wireframes of pages. Creating the sitemap ensures that you've considered all the key pages in the site, showing their relationship to each other and defining how the site's overall navigation should be structured. Wireframes provide a detailed view of the content that will appear on each page. Although they do not show any actual design elements, the wireframes provide a guide for defining content hierarchy on the page.

Visual Design

Once the blueprint for the site has been defined through the creation of the sitemap and wireframes, the next step is to create a visual style. The overall visual style will most likely be determined by the visual brand of the organization; the goal being to connect the Web with all other forms of the organization's communications. The organization's brand plays an important role in this part of the process, as designers will want to visually convey key brand perceptual ideas within the design.

Site Development

With designs approved, it's time to flesh out the design of the pages, develop new content and refine old content, create videos, slideshows, podcasts and other media that will appear on the site as well as start to build out the HTML and CSS of the site.

Site Testing

Before the site is launched, it will be placed on a production server where only internal audiences and anyone who you share the link with can view it. Testing of the site is critical as there will inevitably be issues that need to be addressed before the site goes live. There is nothing that erodes a brand more than a site that doesn't function properly or that has misspellings or broken design elements. At this stage the site will need to be reviewed on multiple browsers (Firefox, Safari, Internet Explorer) and multiple devices (laptops, tablets, and mobile) to see if and where breaks occur.

Launch

The big day. You've tested the site, had it reviewed and approved by the project stakeholders, and you're ready to launch. But once the site is launched, the project isn't over — you should be prepared to address feedback from users adapting to the new site. Expect to make some immediate changes to the site, such as fixing broken links, editing copy and making adjustments. The Web is a fluid medium that changes on a daily, if not hourly basis — change is inevitable.

Site Maintenance

Websites are living, breathing entities and need constant care and maintenance. Updating content, making changes to the backend and fixing broken links are all in a day's work. All of these phases are critical to the Web design process. But the thread that runs through the process is strategy: the desire to achieve a goal, to move the organization forward, to prosper in a competitive environment. Let's take a look at what strategy is, how it is formulated and how it translates to the Web.

2. IP Address (Internet Protocol Address)

- An identifier for a computer or device on a TCP/IP network.
- Used to distinguish one computer from other computers connected to a TCP/IP network such as Internet or Intranet
- Assigned by ICANN (Internet Corporation for Assigned Names and Numbers) to every computer connected to an IP network and strictly controlled to avoid duplicates
- Represented as a group of four numbers from 0 to 255 separated by periods (e.g. “123. 45. 225. 1.”) The format is based on IPv4 (Internet Protocol version 4)
- Organized into hierarchical classes as follows:
 - Class A – all are huge organizations and require the highest possible categorization (i.e. General Electric Company, IBM Corporation, AT&T, Hewlett Packard Company, Ford Motor Company, and the Defense Information Systems Agency). Supports 16 million hosts on each of 128 networks
 - Class B – supports 65,000 hosts on each of 16,000 networks
 - Class C – supports 254 hosts on each of 2 million networks

- Class D – intended for multicast purposes
- Class E – reserved for future use
- Because IP addresses are represented by numbers and they are too complicated to remember for humans, the idea of naming sites resulted in Web URLs and e-mail addresses.

URL (Uniform Resource Locator)

- Represents an address of a certain file on the TCP/IP network and leads a user to a file on any computer connected to the Internet anywhere in the world.
- The format is standardized as <the method of protocol to be used://the domain name/the directory name/the file name>
- An example:
`<http://www.gmu.edu/departments/telecomm/special.html>`
- http: — protocol to be used to search for the other protocols such as ftp, telnet, gopher
- www.gmu.edu — the domain or host name, the name of the network or the computer you are trying to contact, should be unique in the world
- departments/telecomm/ — indicates there is a folder or a directory named department and subdirectory named telecomm on the web site's computer disk, should be connected with a slash (/)
- special.html — the file name in the directory, could end with such as html, htm, cgi, etc.

3 HTML documents are composed of elements that have three components- a pair of element tags – start tag, end tag; element attributes given within tags and actual, textual and graphic content. HTML element is everything that lies between and including tags. (Tag is a keyword which is enclosed within angle brackets).

XHTML documents has only one root element. All elements including variables must be in lower case, and values assigned must be surrounded by quotation marks, closed and nested

for being recognized. This is a mandatory requirement in XHTML unlike HTML where it is optional. The declaration of DOCTYPE would determine rules for documents to follow.

Aside from the different opening declarations for a document, the differences between an HTML 4.01 and XHTML 1.0 document—in each of the corresponding DTDs—are largely syntactic. The underlying syntax of HTML allows many shortcuts that XHTML does not, such as elements with optional opening or closing tags, and even EMPTY elements which must not have an end tag. By contrast, XHTML requires all elements to have an opening tag or a closing tag. XHTML, however, also introduces a new shortcut: an XHTML tag may be opened and closed within the same tag, by including a slash before the end of the tag like this:
. The introduction of this shorthand, which is not used in the SGML declaration for HTML 4.01, may confuse earlier software unfamiliar with this new convention. A fix for this is to include a space before closing the tag, as such:
.

XHTML vs HTML Specification

HTML and XHTML are closely related and therefore can be documented together. Both HTML 4.01 and XHTML 1.0 have three sub specifications – strict, loose and frameset. The difference opening declarations for a document distinguishes HTML and XHTML. Other differences are syntactic. HTML allows shortcuts like elements with optional tags, empty elements without end tags. XHTML is very strict about opening and closing tags. XHTML uses built in language defining functionality attribute. All syntax requirements of XML are included in a well formed XHTML document.

Note, though, that these differences apply only when an XHTML document is served as an application of XML; that is, with a MIME type of application/xhtml+xml, application/xml, or text/xml. An XHTML document served with a MIME type of text/html must be parsed and interpreted as HTML, so the HTML rules apply in this case. A style sheet written for an XHTML document being served with a MIME type of text/html may not work as intended if the document is then served with a MIME type of application/xhtml+xml. For more information about MIME types, make sure to read [MIME Types](#).

4. All XML must have a root element.

All tags must be closed.

All tags must be properly nested.

Tag names have strict limits.

Tag names are case sensitive.

Tag names cannot contain spaces.

Attribute values must appear within quotes ("").

PART B

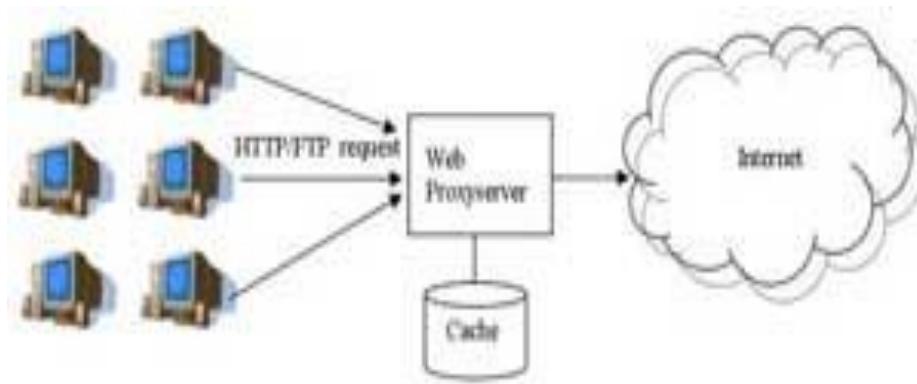
Answer any Two questions. Each carries 9 marks

5. FTP vs HTTP

HTTP (Hypertext Transfer Protocol) and FTP (File Transfer Protocol) are only two of the multitude of protocols that are being used in the internet, each with its own function. The purpose of HTTP is to serve as a means of accessing the world wide web. Websites are accessed using http with the help of browsers. FTP, as the name implies, is used in transferring files from one computer to another. It is a less popular protocol due to small number of people who actually use FTP, and even fewer people who know that they are using it.

Most people don't actually know it, but the HTTP protocol is in use every time we open a site, check our email, or update our blogs. This can be easily checked by looking at the address bar and looking at the very first few letters; chances are its HTTP. The great majority of people who use the FTP protocol are the people who maintain and routinely upload files to websites. FTP provides an easy and hassle free method of site maintenance.

FTP is also a good option for people who want to download files. There are FTP servers who host files and allow people to anonymously login to their site and download huge files. But downloading isn't a capability that is purely held by FTP, it can also be done with HTTP. The rise in popularity of downloads in HTTP is due largely to its linkage with the world wide web. Most sites that offer content for downloads have their files hosted in an HTTP server so that visitors can easily browse and select their files.



Foe example:

```

1. 6.<frameset rows="60,* ,50" frameborder="0">
2.   <frame src="top-frame.html" name="topFrame" scrolling="no">
3.   <frameset cols="200,*" frameborder="0">
4.     <frame src="left-frame.html" name="leftFrame"
      scrolling="no">
5.     <frame src="content-frame.html" name="contentFrame">
6.   </frameset>
7.   <frame src="bottom_frame3.html" name="linksFrame"
      scrolling="no">
8. </frameset>
9. <frameset rows="15%,85%">
10.    <frame src="top.html" noresize scrolling="no"
      frameborder="0">
11.    <frame sec="bottom.html">
12.  </frameset>
```

7. The common gateway interface (CGI) is a standard way for a Web server to pass a Web user's request to an application program and to receive data back to forward to the user. When the user requests a Web page (for example, by clicking on a highlighted word or entering a Web site address), the server sends back the requested page. However, when a user fills out a form on a Web page and sends it in, it usually needs to be processed by an application program. The Web server typically passes the form information to a small application program that processes the data and may send back a confirmation message. This method or convention for passing data back and forth between the server and the application is called

the common gateway interface (CGI). It is part of the Web's Hypertext Transfer Protocol (HTTP).

If you are creating a Web site and want a CGI application to get control, you specify the name of the application in the uniform resource locator (URL) that you code in an HTML file. This URL can be specified as part of the FORMS tags if you are creating a form. For example, you might code:

```
<FORM METHOD=POST ACTION=http://www.mybiz.com/cgi-bin/formprog.pl>
```

and the server at "mybiz.com" would pass control to the CGI application called "formprog.pl" to record the entered data and return a confirmation message. (The ".pl" indicates a program written in PERL but other languages could have been used.)

The common gateway interface provides a consistent way for data to be passed from the user's request to the application program and back to the user. This means that the person who writes the application program can make sure it gets used no matter which operating system the server uses (PC, Macintosh, UNIX, OS/390, or others). It's simply a basic way for information to be passed from the Web server about your request to the application program.

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language.[1] Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.[2]

CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts.[3] This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. It can also display the web page differently depending on the screen size or viewing device. Readers can

also specify a different style sheet, such as a CSS file stored on their own computer, to override the one the author specified.

Changes to the graphic design of a document (or hundreds of documents) can be applied quickly and easily, by editing a few lines in the CSS file they use, rather than by changing markup in the documents.

The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities (or weights) are calculated and assigned to rules, so that the results are predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents. d back again.

PART C

Answer all questions. Each carries 3 marks

8. JavaScript (/dʒɑ:vəskrɪpt/[6]), often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content engineering. It is used to make webpages interactive and provide online programs, including video games. The majority of websites employ it, and all modern web browsers support it without the need for plug-ins by means of a built-in JavaScript engine. Each of the many JavaScript engines represent a different implementation of JavaScript, all based on the ECMAScript specification, with some engines not supporting the spec fully, and with many engines supporting additional features beyond ECMA.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

Although there are strong outward similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design; JavaScript was influenced by programming languages such as Self and Scheme

Dynamic HTML, or DHTML, is an umbrella term for a collection of technologies used together to create interactive and animated websites[1] by using a combination of a static markup language (such as HTML), a client-side scripting language (such as JavaScript), a presentation definition language (such as CSS), and the Document Object Model (DOM).[2] The application of DHTML was introduced by Microsoft with the release of Internet Explorer 4 in 1997.

DHTML allows scripting languages to change variables in a web page's definition language, which in turn affects the look and function of otherwise "static" HTML page content, after the page has been fully loaded and during the viewing process. Thus the dynamic characteristic of DHTML is the way it functions while a page is viewed, not in its ability to generate a unique page with each page load.

By contrast, a dynamic web page is a broader concept, covering any web page generated differently for each user, load occurrence, or specific variable values. This includes pages created by client-side scripting, and ones created by server-side scripting (such as PHP, Perl, JSP or ASP.NET) where the web server generates content before sending it to the client.

DHTML is differentiated from Ajax by the fact that a DHTML page is still request/reload-based. With DHTML, there may not be any interaction between the client and server after the page is loaded; all processing happens in JavaScript on the client side. By contrast, an Ajax page uses features of DHTML to initiate a request (or 'subrequest') to the server to perform additional actions. For example, if there are multiple tabs on a page, pure DHTML approach would load the contents of all tabs and then dynamically display only the one that is active, while AJAX could load each tab only when it is really needed.

10 he JavaScript programming language, developed by Netscape, Inc., is not part of the Java platform.

JavaScript does not create applets or stand-alone applications. In its most common form, JavaScript resides inside HTML documents, and can provide levels of interactivity to web pages that are not achievable with simple HTML.

Key differences between Java and JavaScript:

Java is an OOP programming language while Java Script is an OOP scripting language.

Java creates applications that run in a virtual machine or browser while JavaScript code is run on a browser only.

Java code needs to be compiled while JavaScript code are all in text.

They require different plug-ins..

11. VBScript ("Microsoft Visual Basic Scripting Edition") is an Active Scripting language developed by Microsoft that is modeled on Visual Basic. It allows Microsoft Windows system administrators to generate powerful tools for managing computers with error handling, subroutines, and other advanced programming constructs. It can give the user complete control over many aspects of their computing environment.

VBScript uses the Component Object Model to access elements of the environment within which it is running; for example, the FileSystemObject (FSO) is used to create, read, update and delete files. VBScript has been installed by default in every desktop release of Microsoft Windows since Windows 98;^[1] in Windows Server since Windows NT 4.0 Option Pack;^[2] and optionally with Windows CE (depending on the device it is installed on).

A VBScript script must be executed within a host environment, of which there are several provided with Microsoft Windows, including: Windows Script Host (WSH), Internet Explorer (IE), and Internet Information Services (IIS).^[3] Additionally, the VBScript hosting environment is embeddable in other programs, through technologies such as the Microsoft Script Control (msscript.ocx).

<% Option Explicit

%><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

```

<title>VBScript Example</title>

</head>

<body>

<div><%
    ' Grab current time from Now() function.

    ' An '=' sign occurring after a context switch (<%>) is shorthand
    ' for a call to the Write() method of the Response object.

    Dim timeValue = Now %>

    The time, in 24-hour format, is

    <%=Hour(timeValue)%>:<%=Minute(timeValue)%>:<%=Second(timeValue)%>.

</div>

</body>

</html>

```

AJAX Ajax (also AJAX; /'eɪdʒæks/; short for "Asynchronous JavaScript And XMLHttpRequest") is a set of Web development techniques using many Web technologies on the client side to create asynchronous Web applications. With Ajax, Web applications can send and retrieve data from a server asynchronously (in the background) without interfering with the display and behavior of the existing page. By decoupling the data interchange layer from the presentation layer, Ajax allows for Web pages, and by extension Web applications, to change content dynamically without the need to reload the entire page. In practice, modern implementations commonly utilize JSON instead of XML due to the advantages of JSON being native to JavaScript.

Ajax is not a single technology, but rather a group of technologies. HTML and CSS can be used in combination to mark up and style information. The webpage can then be modified by JavaScript to dynamically display – and allow the user to interact with — the new information. The built-in XMLHttpRequest object within JavaScript is commonly used to execute Ajax on webpages allowing for websites to load content onto the screen without refreshing the page. Ajax is also not a new technology, or another different language, just existing technologies used in new ways.

PART D
Answer any Two questions. Each carries 9 marks

12. The web server needs a JSP engine, i.e, a container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages. This tutorial makes use of Apache which has built-in JSP container to support JSP pages development.

A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs.

Following diagram shows the position of JSP container and JSP files in a Web application.

JSP Architecture

JSP Processing

The following steps explain how the web server creates the Webpage using JSP –

As with a normal page, your browser sends an HTTP request to the web server.

The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with .jsp instead of .html.

The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to `println()` statements and all JSP elements are converted to Java code. This code implements the corresponding dynamic behavior of the page.

The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.

A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format. The output is further passed on to the web server by the servlet engine inside an HTTP response.

The web server forwards the HTTP response to your browser in terms of static HTML content.

Finally, the web browser handles the dynamically-generated HTML page inside the HTTP response exactly as if it were a static page.

All the above mentioned steps can be seen in the following diagram –

JSP Processing

Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet. If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents. This makes the process more efficient than with the other scripting languages (such as PHP) and therefore faster.

So in a way, a JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet.

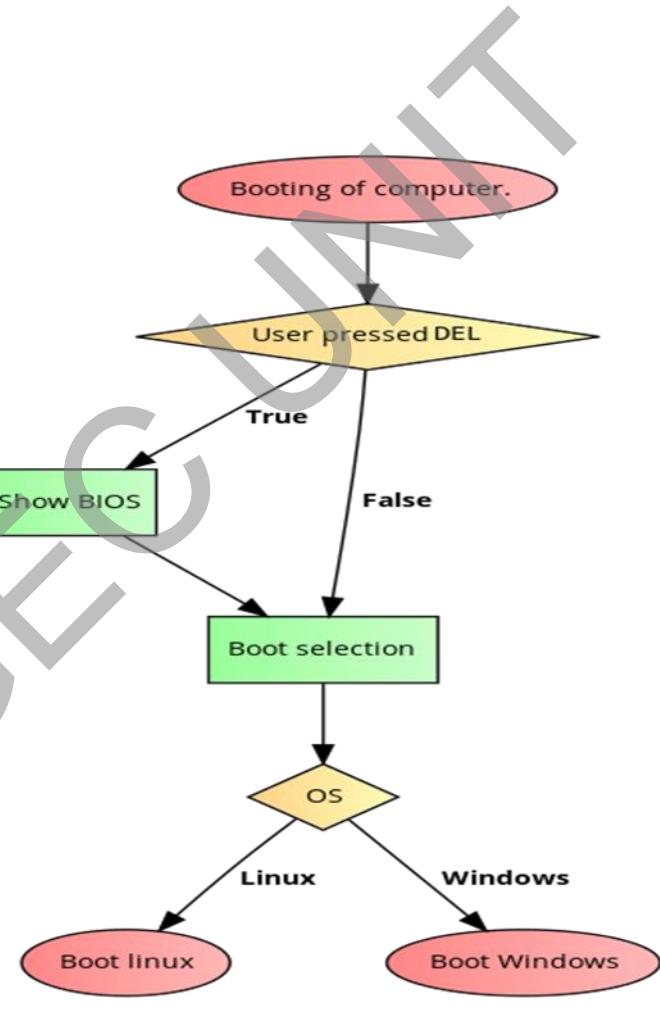
13. JavaScript (/dʒɑːvə,skrɪpt/[6]), often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content engineering. It is used to make webpages interactive and provide online programs, including video games. The majority of websites employ it, and all modern web browsers support it without the need for plug-ins by means of a built-in JavaScript engine. Each of the many JavaScript engines represent a different implementation of JavaScript, all based on the ECMAScript specification, with some engines not supporting the spec fully, and with many engines supporting additional features beyond ECMA.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

Although there are strong outward similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design; JavaScript was influenced by programming languages such as Self and Scheme

```
Booting of computer.;
if(User pressed `DEL` ) {
    Show BIOS;
}
Boot selection;
switch(OS) {
    case Linux:
        Boot linux;
        break;
    case Windows:
        Boot Windows;
        break;
}
```



14.The HTML DOM Tree of Objects

DOM HTML tree

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

JavaScript can change all the HTML elements in the page

JavaScript can change all the HTML attributes in the page

JavaScript can change all the CSS styles in the page

JavaScript can remove existing HTML elements and attributes

JavaScript can add new HTML elements and attributes

JavaScript can react to all existing HTML events in the page

JavaScript can create new HTML events in the page

What You Will Learn

In the next chapters of this tutorial you will learn:

How to change the content of HTML elements

How to change the style (CSS) of HTML elements

How to react to HTML DOM events

How to add and delete HTML elements

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

Core DOM - standard model for all document types

XML DOM - standard model for XML documents

HTML DOM - standard model for HTML documents

What is the HTML DOM?

The HTML DOM is a standard object model and programming interface for HTML. It defines:

The HTML elements as objects

The properties of all HTML elements

The methods to access all HTML elements

The events for all HTML elements

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

PART E

Answer any four questions. Each carries 10 marks

15 XML stands for eXtensible Markup Language.

XML was designed to store and transport data.

XML was designed to be both human- and machine-readable.

XML Example 1

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<note>
```

```
 <to>Tove</to>
```

```
 <from>Jani</from>
```

```
<heading>Reminder</heading>  
<body>Don't forget me this weekend!</body>  
</note>
```

Display the XML File » Display the XML File as a Note »

XML Example 2

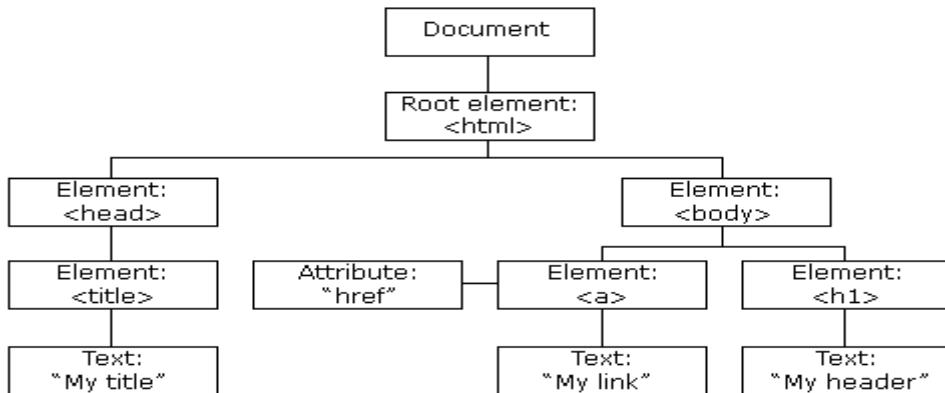
```
<?xml version="1.0" encoding="UTF-8"?>  
<breakfast_menu>  
<food>  
  <name>Belgian Waffles</name>  
  <price>$5.95</price>  
  <description>  
    Two of our famous Belgian Waffles with plenty of real maple syrup  
  </description>  
  <calories>650</calories>  
</food>  
<food>  
  <name>Strawberry Belgian Waffles</name>  
  <price>$7.95</price>  
  <description>  
    Light Belgian waffles covered with strawberries and whipped cream  
  </description>  
  <calories>900</calories>  
</food>  
<food>  
  <name>Berry-Berry Belgian Waffles</name>  
  <price>$8.95</price>
```

<description>
Belgian waffles covered with assorted fresh berries and whipped cream
</description>
<calories>900</calories>
</food>

<food>
<name>French Toast</name>
<price>\$4.50</price>
<description>
Thick slices made from our homemade sourdough bread
</description>
<calories>600</calories>
</food>

<food>
<name>Homestyle Breakfast</name>
<price>\$6.95</price>
<description>
Two eggs, bacon or sausage, toast, and our ever-popular hash browns
</description>
<calories>950</calories>
</food>

</breakfast_menu>



15. XML stands for eXtensible Markup Language.

XML was designed to store and transport data.

XML was designed to be both human- and machine-readable.

XML Example 1

```
<?xml version="1.0" encoding="UTF-8"?>  
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>
```

Display the XML File » Display the XML File as a Note »

XML Example 2

```
<?xml version="1.0" encoding="UTF-8"?>  
<breakfast_menu>  
  <food>
```

<name>Belgian Waffles</name>
<price>\$5.95</price>
<description>
Two of our famous Belgian Waffles with plenty of real maple syrup
</description>
<calories>650</calories>
</food>
<food>
<name>Strawberry Belgian Waffles</name>
<price>\$7.95</price>
<description>
Light Belgian waffles covered with strawberries and whipped cream
</description>
<calories>900</calories>
</food>
<food>
<name>Berry-Berry Belgian Waffles</name>
<price>\$8.95</price>
<description>
Belgian waffles covered with assorted fresh berries and whipped cream
</description>
<calories>900</calories>
</food>
<food>
<name>French Toast</name>
<price>\$4.50</price>
<description>

Thick slices made from our homemade sourdough bread

</description>

<calories>600</calories>

</food>

<food>

<name>Homestyle Breakfast</name>

<price>\$6.95</price>

<description>

Two eggs, bacon or sausage, toast, and our ever-popular hash browns

</description>

<calories>950</calories>

</food>

</breakfast_menu>

15. An XSLT style sheet can emit HTML <STYLE> elements, including CSS specifications, directly into the HTML that results from the XSLT transformation. This option works best when the number of CSS rules is small and easily managed.

To place a <STYLE> element and its CSS rules into the result tree of a transformation, you can include them in the XSLT template that instantiates the HTML <HEAD> element. The book.xsl file below illustrates this.

In book.xsl, the first template rule matches the root element of the XML document, <book>. When it finds a match in the source tree, it instantiates <HTML> and <HEAD> elements in the result tree.

The <HEAD> element includes both a <TITLE> element and a <STYLE> element. The <STYLE> element includes a single CSS rule, which says to display all <H1> elements in the result tree in the indicated font.

Finally, this template rule says to instantiate a <BODY> element, and then to process any children of the matching <book> element. This is the purpose of the <xsl:apply-templates/> element. During this process, the template rule also checks for other matching template rules, and transforms them as necessary.

The second template rule is invoked by the <xsl:apply-templates/> element of the first template rule. This second template rule instantiates an <H1> element. The content of the <H1> element is that of a <book_title> element in the source tree.

17. JSON stands for JavaScript Object Notation

JSON is a lightweight data-interchange format

JSON is "self-describing" and easy to understand

JSON is language independent *

```
var myObj = { "name": "John", "age": 31, "city": "New York" };
```

```
var myJSON = JSON.stringify(myObj);
```

```
window.location = "demo_json.php?x=" + myJSON;
```

//Storing data:

```
myObj = { "name": "John", "age": 31, "city": "New York" };
```

```
myJSON = JSON.stringify(myObj);
```

```
localStorage.setItem("testJSON", myJSON);
```

//Retrieving data:

```
text = localStorage.getItem("testJSON");
```

```
obj = JSON.parse(text);
```

```
document.getElementById("demo").innerHTML = obj.name
```

18. <!DOCTYPE html>

```
<html>
```

```
<body>
```

```
<?php
```

```
echo "My first PHP script!";
```

```
?>
```

```
</body>
```

```
</html>
```

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

19. A cookie is a small piece of text stored on a user's computer by their browser. Common uses for cookies are authentication, storing of site preferences, shopping cart items, and server session identification.

Each time the users' web browser interacts with a web server it will pass the cookie information to the web server. Only the cookies stored by the browser that relate to the domain in the requested URL will be sent to the server. This means that cookies that relate to www.example.com will not be sent to www.exampledomain.com.

In essence, a cookie is a great way of linking one page to the next for a user's interaction with a web site or web application.

What is a "Session"?

A session can be defined as a server-side storage of information that is desired to persist throughout the user's interaction with the web site or web application.

Instead of storing large and constantly changing information via cookies in the user's browser, only a unique identifier is stored on the client side (called a "session id"). This session id is passed to the web server every time the browser makes an HTTP request (ie a page link or AJAX request). The web application pairs this session id with its internal database and retrieves the stored variables for use by the requested page.

Setting and reading cookies

Using the [cookie_set] method we can set cookies to store information for use in later pages. The following code shows how easy it is to store a user's details such as their name and email address which they may have entered on a "Contact Us" form. This would then allow later pages to pre-populate forms with this information.

Cookie_Set(

```
'UserDetails='John Doe|john.doe@example.com',
-Domain='example.com',
-Expires='1440',
-Path='/'
)
```

In this example the cookie named "UserDetails" contains the user name and email address delimited by a "pipe" character. This can be read and interpreted, then output in the following code.

```
local( userDetails = decode_url(cookie('UserDetails'))->split('|'))
```

```
if(#userDetails->size)=>{^
    'User Name =' +#userDetails->get(1)
    '<br />'
    'Email Address =' +#userDetails->get(2)
^}
```

Using Sessions

To store information that is not appropriate to store client-side, we use sessions. Lasso has built in session handling, and deals with the setting and retrieval of the cookie itself. It will automatically set and retrieve the session id, which is the only thing stored client-side.

To set up a new session, we first start the session, then add to it the variables we would like to store in it. Those variables are stored within Lasso's session database.

```
// Start the session.  
  
session_start(  
  
    'mySessionName',  
  
    -expires = 1440,  
  
    -usecookie=true  
)  
  
  
// Add variables to the session  
  
if(session_result('mySessionName') != 'load') => {  
  
    session_addVar('mySessionName', 'sv_userId')  
  
    session_addVar('mySessionName', 'sv_userName')  
  
    session_addVar('mySessionName', 'sv_userEmail')  
  
    session_addVar('mySessionName', 'sv_favouriteColour')  
  
}  
  
  
!var_defined('sv_userId') ? var('sv_userId' = integer)  
!var_defined('sv_userName') ? var('sv_userName' = string)  
!var_defined('sv_userEmail') ? var('sv_userEmail' = string)  
!var_defined('sv_favouriteColour') ? var('sv_favouriteColour' = 'red')  
  
20..<!DOCTYPE html>  
  
<html>  
  
<body>  
  
<?php  
  
echo "My first PHP script!";  
  
?>
```

</body>

</html>

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

SFI GEC UNIT

COLLEGE OF ENGINEERING THALASSERY
S6 BTECH DEGREE MODEL EXAMINATION – 2018
CS368: WEB TECHNOLOGY

Max marks: 100

Duration: 3 hrs

Part A

Answer all questions. Each question carries 3 marks

1. Explain URI

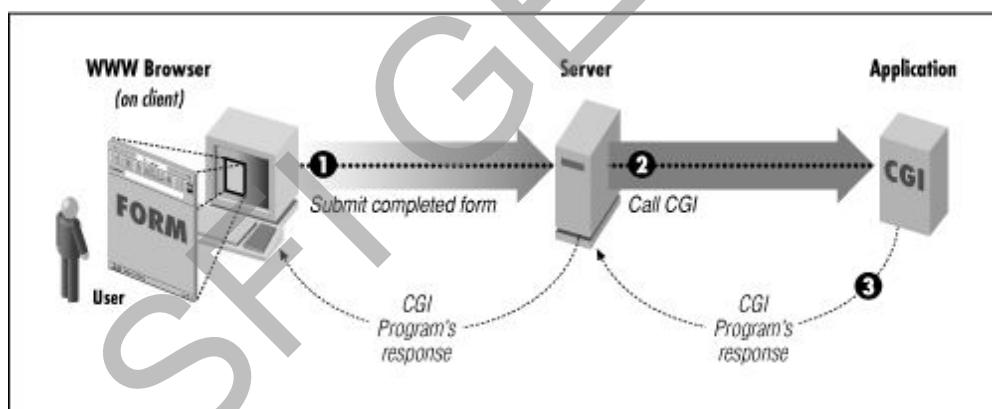
URI (Uniform Resource Identifier) is a sequence of characters that identifies a logical or physical resource

A generic URI is of the form:

scheme://[user[:password]@]host[:port]][/path][?query][#fragment]

2. What is CGI

Common Gateway Interface (CGI) offers a standard protocol for web servers to execute programs running on a server that generates web pages dynamically



3. Differentiate between HTML and XHTML

HTML

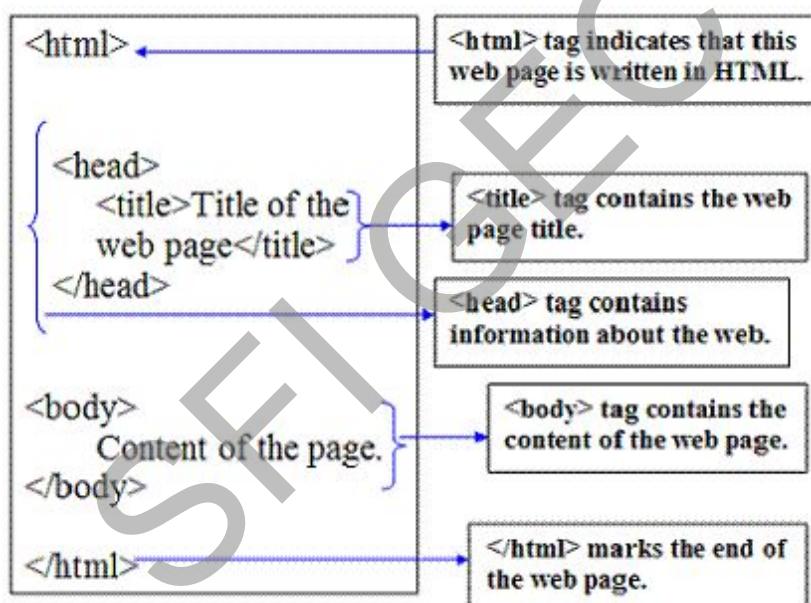
- Start tags are not required for every element.
- End tags are not required for every element.
- Only **void elements** such as br, img, and link may be “self-closed” with />.
- Tags and attributes are case-insensitive.
- Attributes do not need to be quoted.

- Some attributes may be empty (such as checked and disabled).
- Special characters, or entities, do not have to be escaped.
- The document must include an HTML5 DOCTYPE.

XHTML

- All elements must have a start tag.
- Non-void elements with a start tag must have an end tag (p and li, for example).
- Any element may be “self-closed” using />.
- Tags and attributes are case sensitive, typically lowercase.
- Attribute values must be enclosed in quotes.
- Empty attributes are forbidden (checked must instead be checked="checked" or checked="true").
- Special characters must be escaped using character entities.
- XHTML documents must be well-formed

4. Describe the standard HTML document structure



Part B

Answer any 2 questions. Each question carries 9 marks

5.

1. What is the form of HTML comment (2)
 <!-- Write your comments here -->

Example:

<!-- This is a comment -->

<p>This is a paragraph.</p>

<!-- Remember to add more information here -->

2. Create an unordered list in an HTML page (4)

Tags:ul and li

```
<ul>
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ul>
```

3. Explain table tag with attributes (3)

- a. Use the HTML **<table>** element to define a table
- b. Use the HTML **<tr>** element to define a table row
- c. Use the HTML **<td>** element to define a table data
- d. Use the HTML **<th>** element to define a table heading
- e. Use the HTML **<caption>** element to define a table caption
- f. Use the CSS **border** property to define a border
- g. Use the CSS **border-collapse** property to collapse cell borders
- h. Use the CSS **padding** property to add padding to cells
- i. Use the CSS **text-align** property to align cell text
- j. Use the CSS **border-spacing** property to set the spacing between cells
- k. Use the **colspan** attribute to make a cell span many columns
- l. Use the **rowspan** attribute to make a cell span many rows
- m. Use the **id** attribute to uniquely define one table

Example:

```
<table style="width:100%">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
</tr>
<tr>
<td>Jill</td>
<td>Smith</td>
<td>50</td>
</tr>
<tr>
```

```
<td>Eve</td>
<td>Jackson</td>
<td>94</td>
</tr>
</table>
```

6.

1. Create an HTML page and add an image and link to it (4.5)

Tags:

Img and a

```
<!DOCTYPE html>
<html>
  <head>
    <title>Using Image in Webpage</title>
  </head>

  <body>
    <p>Simple Image Insert</p>
    <img src = "/html/images/test.png" alt = "Test Image" />
    <a href = "https://www.tutorialspoint.com" target = "_self">Tutorials Point</a>
  </body>
</html>
```

2. Create an HTML form for user registration with necessary elements (4.5)

Main tags: form, input

Form:

The HTML `<form>` tag is used for creating a form for user input. A form can contain textfields, checkboxes, radio-buttons and more. Forms are used to pass user-data to a specified URL.

Input:

The `<input>` tag specifies an input field where the user can enter data.

`<input>` elements are used within a `<form>` element to declare input controls that allow users to input data.

An input field can vary in many ways, depending on the type attribute.

Example:

```
<!DOCTYPE html>
<html>
```

```
<head>
  <title>HTML form Tag</title>
</head>

<body>
  <form action = "/cgi-bin/hello_get.cgi" method = "get">
    First name:
    <input type = "text" name = "first_name" value = "" maxlength = "100" />
    <br />

    Last name:
    <input type = "text" name = "last_name" value = "" maxlength = "100" />
    <input type = "submit" value = "Submit" />
  </form>
</body>

</html>
```

7.

1. Explain interaction between web browser and web client (4.5)

Step 1

The user specifies what domain and port to connect to. Say that the user want to visit www.ilopia.com. He/she then types the URL www.ilopia.com in the browser (Internet Explorer). Internet Explorer will in this case default to the HTTP protocol, and default to port 80. If the user wanted to connect to another port, he or she had to write the port number.

Step 2

The browser must now know what IP to connect to. This is not special for the communication between the webclient and webserver. The DNS name is just a name, which is translated into an IP. This IP is then used to connect to the server.

The browser will ask the local system for this IP. The system will then find the IP in one or another way. It will first see if it is cached in the local machine (for Windows it could also use the lmhosts file). If it is not, it will ask the DNS server specified in the network settings (probably your ISP's DNS server). If this DNS server does not have this cached, it will ask the top level DNS server.

Step 3

The **client connects to the server**. This is done using the IP and port only. The DNS name (www.ilopia.com) is not in any way used to make this connection. So far, there is only a connection, the server still does not know what to do.

Step 4

The client now **sends a request message** using the HTTP protocol. The simplest request would look like:

GET /index.html HTTP/1.1

This requests ask for the file index.html. A request like this does not specify any host header. So if the webserver is configured so that a host header is required, the webserver does not know what to do with this request. The server should in this case (says the RFC) reply with a "400 Bad Request". A valid request looks like:

GET /index.html HTTP/1.1

Host: www.ilopia.com

Step 5

The **server examines the request message**, and takes action. In the second example above, the webserver will use the host header sent to see if there is something matching that host header. If there is, it will serve index.html from the home folder of that website. If www.ilopia.com is not found as a host header in the webserver, it will use the default (if any) website (this is IIS default settings, it might not be true for other webservers).

Step 6

The **server responds to the request** by sending some header information, and the content of the webpage (see demonstration below).



2. What is MIME

(4.5)

MIME (Multi-Purpose Internet Mail Extensions) is an extension of the original Internet e-mail protocol that lets people use the protocol to exchange different kinds of data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text.

MIME (Multipurpose Internet Mail Extension) media types were originally devised so that e-mails could include information other than plain text. MIME media types indicate the following things –

- How different parts of a message, such as text and attachments, are combined into the message.
- The way in which each part of the message is specified.
- The way different items are encoded for transmission so that even software that was designed to work only with ASCII text can process the message.

MIME content types consist of two parts –

- A main type
- A sub-type

Part C

Answer all questions. Each question carries 3 marks

8. Explain the features of CSS

CSS stands for Cascading Style Sheets

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

CSS saves a lot of work. It can control the layout of multiple web pages all at once

9. Write short note on span and div tags

The Element

The element is often used as a container for some text.

The element has no required attributes, but both **style** and **class** are common.

When used together with CSS, the element can be used to style parts of the text:

Eg:

```
<h1>My <span style="color:red">Important</span> Heading</h1>
```

The <div> Element

The <div> element is often used as a container for other HTML elements.

The <div> element has no required attributes, but both **style** and **class** are common.

When used together with CSS, the <div> element can be used to style blocks of content:

Eg:

```
<div style="background-color:black;color:white;padding:20px;">
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
</div>
```

10. Explain screen output and keyboard input in javascript

The JavaScript model for the HTML document is the Document object

The model for the browser display window is the Window object

The Window object has two properties, document and window, which refer to the document and window objects, respectively

The Document object has a method, write, which dynamically creates content

The parameter is a string, often catenated from parts, some of which are variables
e.g., `document.write("Answer: " + result + "
");`

The Window object has three methods for creating dialog boxes, alert, confirm, and prompt

```
alert("Hej! \n");
```

Parameter is plain text, not HTML

Opens a dialog box which displays the parameter string and an OK button
`confirm("Do you want to continue?");`

Opens a dialog box and displays the parameter and two buttons, OK and Cancel

```
prompt("What is your name?", "");
```

Opens a dialog box and displays its string parameter, along with a text box and two buttons, OK and Cancel

The second parameter is for a default response if the user presses OK without typing a response in the text box (waits for OK)

Example for finding Roots of a quadratic equation

```
!DOCTYPE html>
<!-- roots.html
A document for roots.js
-->
<html lang = "en">
<head>
<title> roots.html </title>
<meta charset = "utf-8" />
</head>
<body>
<script type = "text/javascript" src = "roots.js" >
</script>
</body>
</html>
```

```
// roots.js
// Compute the real roots of a given quadratic
// equation. If the roots are imaginary, this script
// displays NaN, because that is what results from
// taking the square root of a negative number

// Get the coefficients of the equation from the user

var a = prompt("Find the solution for ax^2 + bx +c \n What is the value of 'a'?
\n", "");
var b = prompt("What is the value of 'b'? \n", "");
var c = prompt("What is the value of 'c'? \n", "");

// Compute the square root and denominator of the result

var root_part = Math.sqrt(b * b - 4.0 * a * c);
var denom = 2.0 * a;

// Compute and display the two roots

var root1 = (-b + root_part) / denom;
var root2 = (-b - root_part) / denom;
document.write("The first root is: ", root1, "<br />");
```

```
document.write("The second root is: ", root2, "<br />");
```

11. What parameter passing does javascript use

Parameter Rules

JavaScript function definitions do not specify data types for parameters.

JavaScript functions do not perform type checking on the passed arguments.

JavaScript functions do not check the number of arguments received.

Parameter Defaults

If a function is called with missing arguments (less than declared), the missing values are set to: undefined

Sometimes this is acceptable, but sometimes it is better to assign a default value to the parameter:

Example

```
function myFunction(x, y) {  
    if (y === undefined) {  
        y = 0;  
    }  
}
```

The Arguments Object

JavaScript functions have a built-in object called the arguments object.

The argument object contains an array of the arguments used when the function was called (invoked).

This way you can simply use a function to find (for instance) the highest value in a list of numbers:

Example

```
x = findMax(1, 123, 500, 115, 44, 88);
```

```
function findMax() {  
    var i;  
    var max = -Infinity;  
    for (i = 0; i < arguments.length; i++) {  
        if (arguments[i] > max) {  
            max = arguments[i];  
        }  
    }  
}
```

```
return max;  
}
```

Arguments are Passed by Value

The parameters, in a function call, are the function's arguments.

JavaScript arguments are passed by value: The function only gets to know the values, not the argument's locations.

If a function changes an argument's value, it does not change the parameter's original value.
Changes to arguments are not visible (reflected) outside the function.

Objects are Passed by Reference

In JavaScript, object references are values.

Because of this, objects will behave like they are passed by reference:

If a function changes an object property, it changes the original value.

Changes to object properties are visible (reflected) outside the function.

Part D

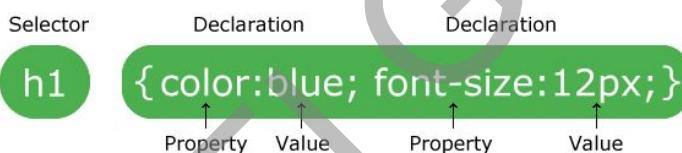
Answer any 2 questions. Each question carries 9 marks

12.

1. Give the syntax of CSS (4.5)

CSS Syntax

A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

In the following example all <p> elements will be center-aligned, with a red text color:

Example

```
p {  
    color: red;  
    text-align: center;  
}
```

2. List any 5 CSS text properties (4.5)

Property	Description
<u>color</u>	Sets the color of text
<u>direction</u>	Specifies the text direction/writing direction
<u>letter-spacing</u>	Increases or decreases the space between characters in a text
<u>line-height</u>	Sets the line height
<u>text-align</u>	Specifies the horizontal alignment of text
<u>text-decoration</u>	Specifies the decoration added to text
<u>text-indent</u>	Specifies the indentation of the first line in a text-block
<u>text-shadow</u>	Specifies the shadow effect added to text
<u>text-transform</u>	Controls the capitalization of text
<u>text-overflow</u>	Specifies how overflowed content that is not displayed should be signaled to the user
<u>unicode-bidi</u>	Used together with the <u>direction</u> property to set or return whether the text should be overridden to support multiple languages in the same document
<u>vertical-align</u>	Sets the vertical alignment of an element
<u>white-space</u>	Specifies how white-space inside an element is handled
<u>word-spacing</u>	Increases or decreases the space between words in a text

13.

1. Explain how functions can be written in javascript with example (4.5)

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

Example

```
function myFunction(p1, p2) {  
    return p1 * p2;          // The function returns the product of p1 and p2  
}
```

JavaScript Function Syntax

A JavaScript function is defined with the **function** keyword, followed by a **name**, followed by parentheses **()**.

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:

(parameter1, parameter2, ...)

The code to be executed, by the function, is placed inside curly brackets: **{ }{ }**

```
function name(parameter1, parameter2, parameter3) {
```

code to be executed

```
}
```

Function Invocation

The code inside the function will execute when "something" **invokes** (calls) the function:

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

Function Return

When JavaScript reaches a **return statement**, the function will stop executing.

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Functions often compute a **return value**. The return value is "returned" back to the "caller":

Example

Calculate the product of two numbers, and return the result:

```
var x = myFunction(4, 3); // Function is called, return value will end up in x
```

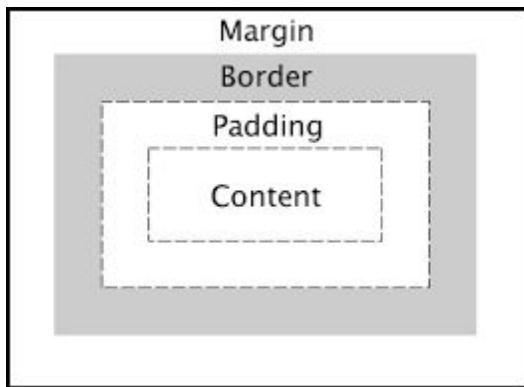
```
function myFunction(a, b) {  
    return a * b;          // Function returns the product of a and b  
}
```

2. Explain CSS box model in detail

(4.5)

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

14.

1. Explain how javascript handles arrays with example (4.5)

JavaScript arrays are used to store multiple values in a single variable.

Example

```
var cars = ["Saab", "Volvo", "BMW"];
```

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";
```

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

```
var array_name = [item1, item2, ...];
```

Using the JavaScript Keyword new

The following example also creates an Array, and assigns values to it:

Example

```
var cars = new Array("Saab", "Volvo", "BMW");
```

Access the Elements of an Array

You refer to an array element by referring to the index number.

This statement accesses the value of the first element in cars:

```
var name = cars[0];
```

This statement modifies the first element in cars:

```
cars[0] = "Opel";
```

Example

```
var cars = ["Saab", "Volvo", "BMW"];  
document.getElementById("demo").innerHTML = cars[0];
```

2. Write short note on object creation and modification in javascript (4.5)

Objects are composed of attributes. If an attribute contains a function, it is considered to be a method of the object, otherwise the attribute is considered a property.

Object Properties

Object properties can be any of the three primitive data types, or any of the abstract data types, such as another object. Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables that are used throughout the page.

The syntax for adding a property to an object is –

```
objectName.objectProperty = propertyName;
```

Accessing Object methods:

```
objectName.methodName()
```

Part E

Answer any 4 questions. Each question carries 10 marks

15.

1. What is XML namespace (2)

Name conflicts in XML can easily be avoided using a name prefix.

This XML carries information about an HTML table, and a piece of furniture:

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

```
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

In the example above, there will be no conflict because the two `<table>` elements have different names.

2. What is XSLT (2)

XSL (eXtensible Stylesheet Language) is a styling language for XML.

XSLT stands for XSL Transformations.

XSLT is a language for transforming XML documents.

3. Explain XML schema with Example (6)

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:
the elements and attributes that can appear in a document

the number of (and order of) child elements

data types for elements and attributes

default and fixed values for elements and attributes

One of the greatest strength of XML Schemas is the support for data types.

It is easier to describe allowable document content

It is easier to validate the correctness of data

It is easier to define data facets (restrictions on data)

It is easier to define data patterns (data formats)

It is easier to convert data between different data types

Eg:

```
<xs:element name="note">

<xs:complexType>
<xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
</xs:sequence>
</xs:complexType>

</xs:element>
```

The Schema above is interpreted like this:

- `<xs:element name="note">` defines the element called "note"
- `<xs:complexType>` the "note" element is a complex type
- `<xs:sequence>` the complex type is a sequence of elements
- `<xs:element name="to" type="xs:string">` the element "to" is of type string (text)
- `<xs:element name="from" type="xs:string">` the element "from" is of type string
- `<xs:element name="heading" type="xs:string">` the element "heading" is of type string
- `<xs:element name="body" type="xs:string">` the element "body" is of type string

16.

1. How is XML document displayed using CSS (5)

Eg:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
    <CD>
        <TITLE>Empire Burlesque</TITLE>
        <ARTIST>Bob Dylan</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Columbia</COMPANY>
        <PRICE>10.90</PRICE>
        <YEAR>1985</YEAR>
    </CD>
    <CD>
        <TITLE>Hide your heart</TITLE>
        <ARTIST>Bonnie Tyler</ARTIST>
```

```
<COUNTRY>UK</COUNTRY>
<COMPANY>CBS Records</COMPANY>
<PRICE>9.90</PRICE>
<YEAR>1988</YEAR>
</CD>

.
.

</CATALOG>
```

2. How is raw XML document displayed

(5)

Eg:

```
<?xml version="1.0" encoding="UTF-8"?>
- <note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

17. What is JSON

(10)

JSON: JavaScript Object Notation.

JSON is a syntax for storing and exchanging data.

JSON is text, written with JavaScript object notation.

Uses of JSON

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.
- It can be used with modern programming languages.

Characteristics of JSON

- JSON is easy to read and write.
- It is a lightweight text-based interchange format.
- JSON is language independent.

Simple Example in JSON

The following example shows how to use JSON to store information related to books based on their topic and edition.

```
{  
  "book": [  
    {  
      "topic": "Technology",  
      "edition": "1st",  
      "title": "Java Programming",  
      "author": "John Doe",  
      "price": 120  
    },  
    {  
      "topic": "Science",  
      "edition": "2nd",  
      "title": "Physics",  
      "author": "Jane Smith",  
      "price": 150  
    }  
  ]  
}
```

```
{  
    "id": "01",  
    "language": "Java",  
    "edition": "third",  
    "author": "Herbert Schildt"  
},  
  
{  
    "id": "07",  
    "language": "C++",  
    "edition": "second",  
    "author": "E.Balagurusamy"  
}  
]  
}
```

18. Explain form handling in PHP with example

(10)

Example1;

```
<html>  
<body>  
  
<form action="welcome.php" method="post">  
Name: <input type="text" name="name"><br>  
E-mail: <input type="text" name="email"><br>  
<input type="submit">  
</form>  
  
</body>  
</html>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables.

The "welcome.php"

```
<html>  
<body>
```

Welcome <?php echo \$_POST["name"]; ?>

Your email address is: <?php echo \$_POST["email"]; ?>

```
</body>
```

```
</html>
```

output

Welcome John

19. Explain PHP arrays with example

(10)

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";  
$cars2 = "BMW";  
$cars3 = "Toyota";
```

In PHP, the array() function is used to create an array:

```
array();
```

In PHP, there are three types of arrays:

- Indexed arrays - Arrays with a numeric index
- Associative arrays - Arrays with named keys
- Multidimensional arrays - Arrays containing one or more arrays

Eg1:Indexing

```
<?php  
$cars = array("Volvo", "BMW", "Toyota");  
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".  
?>
```

Eg2:length of arrays:

```
<?php  
$cars = array("Volvo", "BMW", "Toyota");  
echo count($cars);  
?>
```

Eg3:Looping through arrays

```
<?php  
$cars = array("Volvo", "BMW", "Toyota");  
$arrlength = count($cars);  
  
for($x = 0; $x < $arrlength; $x++) {  
    echo $cars[$x];  
    echo "<br>";  
}  
?>
```

20.

1. Describe Syntactic characteristics of PHP (5)

PHP scripts:

Embedded in HTML documents

The <script> tag can be used but cumbersome

The embedding tag <?php and ?>

The only way that it will work with XML and XHTML documents

In files that are referenced by HTML documents

Using the include command:

include("table2.inc");

Content of file table2.inc are copied into the document where the call appears

PHP preprocessor changes from interpreter to copy mode when an include is encountered

All variable in PHP begin with the dollar sign (\$)

Followed by a letter or underscore, followed by any number of letters, digits, or underscores

Variable names are case sensitive

Neither reserved names nor functions are case sensitive

PHP Comments:

Single-line comments: Specified with \$ as in Perl

Single-line comments: Specified with // as in JavaScript

Multiple-line comments: Specified with /* and */ as in JavaScript

PHP statements terminated with a ;)

Braces used to form compound statements for control structures

Braces cannot define blocks with logically scoped variables, unless it is the body of a function

2. Write note on PHP control statements (5)

if and else

elseif

while

do...while

for

foreach

break and continue

Switch

KAITHANG

(Write syntax and/or example)

SFI GEC UNIT

SFI GEC UNIT

SFI GEC UNIT

**COLLEGE OF ENGINEERING THALASSERY
S6 BTECH DEGREE MODEL EXAMINATION – 2018
CS368: WEB TECHNOLOGY**

Max marks: 100

hrs

Duration: 3

Part A

Answer all questions. Each question carries 3 marks

1. Explain URI
2. What is CGI
3. Differentiate between HTML and XHTML
4. Describe the standard HTML document structure

Part B

Answer any 2 questions. Each question carries 9 marks

5.
 1. What is the form of HTML comment (2)
 2. Create an unordered list in an HTML page (4)
 3. Explain table tag with attributes (3)
6.
 1. Create an HTML page and add an image and link to it (4.5)
 2. Create an HTML form for user registration with necessary elements (4.5)
7.
 1. Explain interaction between web browser and web client (4.5)
 2. What is MIME (4.5)

Part C

Answer all questions. Each question carries 3 marks

8. Explain the features of CSS
9. Write short note on span and div tags
10. Explain screen output and keyboard input in javascript
11. What parameter passing does javascript use

Part D

Answer any 2 questions. Each question carries 9 marks

12.

1. Give the syntax of CSS (4.5)
2. List any 5 CSS text properties (4.5)

13.

1. Explain how functions can be written in javascript with example (4.5)
2. Explain CSS box model in detail (4.5)

14.

1. Explain how javascript handles arrays with example (4.5)
2. Write short note on object creation and modification in javascript (4.5)

Part E

Answer any 4 questions. Each question carries 10 marks

15.

1. What is XML namespace (2)
2. What is XSLT (2)
3. Explain XML schema with Example (6)

16.

1. How is XML document displayed using CSS (5)
2. How is raw XML document displayed (5)

17. What is JSON (10)

18. Explain form handling in PHP with example (10)

19. Explain PHP arrays with example (10)

20.

1. Describe Syntactic characteristics of PHP (5)
2. Write note on PHP control statements (5)