MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2013 Certified)

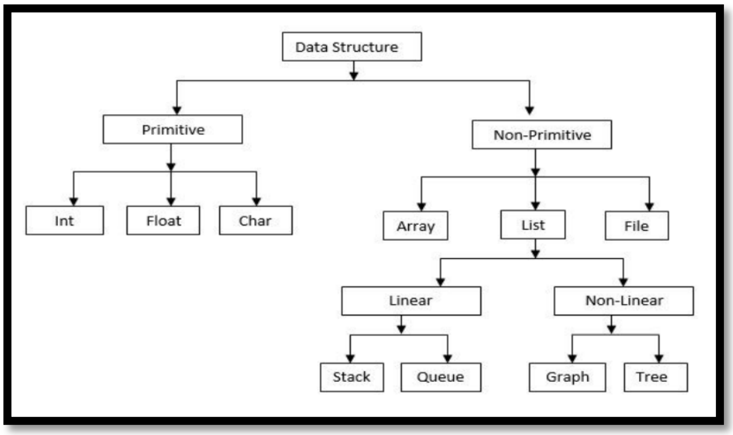
SUMMER – 19 EXAMINATION

Subject Name: Data Structure Model Answer Subject Code: 17330

Important Instructions to examiners:

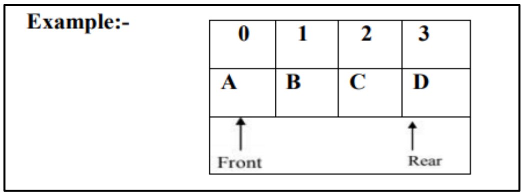
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgments on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

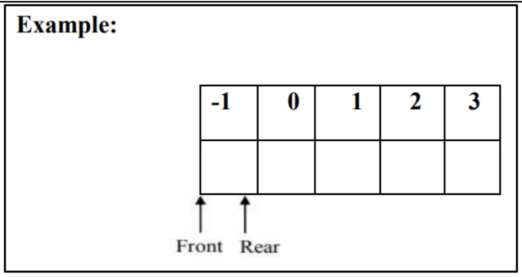
Q. No.	Sub Q. N.	Answer	Marking Scheme
1	A	**Attempt any SIX of the following :**	**12 M**
	a	**Define searching. Give its type.**	**2 M**
	Ans	**Searching** is an operation or a technique that helps finds the place of a given element or value in the list. **Types:** • Linear Search or Sequential Search • Binary Search	Definition : 1M, Types : 1M
	b	**Define a complete binary tree.**	**2 M**
	Ans	A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.	Relevant definition : 2M
	c	**Define following w.r.t. tree:** **i. tree** **ii. leaf node**	**2 M**
	Ans	**1. Tree**: A tree data structure can be defined recursively as a collection of nodes (starting at a root node), where each node is a data structure consisting of a value, together with a list of references to nodes (the "children"), with the constraints that no reference is duplicated, and none points to the root.	Each Definition : 1M

		2. Leaf node: A node having no child or of degree zero is called a terminal node or leaf node.	
	d	**Give classification of data structure.**	**2 M**
	Ans	 Data Structure Primitive Non-Primitive Int Float Char Array List File Linear Non-Linear Stack Queue Graph Tree	Relevant answer : 2M
	e	**Define following w.r.t. tree:** **i. node** **ii. arcs**	**2 M**
	Ans	**Node:** A tree is a collection of entities called nodes. Nodes are connected by edges. Each node contains a value or data, and it may or may not have a child node. **Arcs:** A (rooted) tree consists of a set of nodes (or vertices) and a set of arcs (or edges). Each arc links a parent node to one of the parent's children.	Each definition : 1M
	f	**Write polish notations**.	**2 M**
	Ans	1. Infix Notation 2. Prefix (Polish) Notation 3. Postfix (Reverse-Polish) Notation	Each : 1M
	g	**Explain space complexity with example.**	**2 M**

	Ans	**Space complexity:** Space complexity of a program/algorithm is the amount of memory that it needs to run to completion. The space needed by the program is the sum of the following components:-
		Example: - additional space required when function uses recursion.
		void main()
		{
		int i,,n,sum ,x;
		sum=0;
		printf("\n enter no of data to be added");
		scanf("%d",&n);
		for(i=1;i<=n;i++)
		{
			Definition : 1M,
			Example : 1M

		scanf(“%d”,&x);
		sum=sum+x;
		}
		printf(“\n sum =%d”, sum);
		}
		Space required to store the variables i,,n,sum and x=2+2+2+2=8 (int requires 2 bytes of memory space)	

	h	**Write Linear search algorithm.**	**2 M**
	Ans	**Algorithm:** Linear Search (Array A, Value x) Step 1: Set i to 1 Step 2: if i > n then go to step 7 Step 3: if A[i] = x then go to step 6 Step 4: Set i to i + 1 Step 5: Go to Step 2 Step 6: Print Element x Found at index i and go to step 8 Step 7: Print element not found Step 8: Exit	Relevant algorithm : 2M
	B	**Attempt any TWO :**	**8 M**
	a	**Explain ‘Queue Full’ and ‘Queue Empty’ condition with suitable example.**	**4 M**
	Ans	**Queue full:** A queue is full when its rear pointer points to max -1 position. Max is maximum number of elements in a queue. If rear pointer is not equal to max-1 then a new element can be added to a queue  Example:- 0 1 2 3 A B C D **Queue empty:** A queue is empty when its front pointer points to -1 position. When front pointer is -1 then one cannot delete any data from a queue.	Explanation : 2M, Example : 2M

		 Example:	
	b	**State the need of data structure. Write the operations performed using data structure.**	**4 M**
	Ans	**Need of data structure:** • Data structures are an important way of organizing information or data in a computer. • It has a different ways of storing & organizing data in a computer. • It helps to store data in logical manner. • It allows collection of data to grow & shrink dynamically over time & to organize the information so that one can access it using efficient algorithms. • Specific data structures are essential ingredients of many efficient algorithms, & they make possible management of huge amount of data, such as large collections of databases. **Operations perform on data structure:** • Insertion • Deletion • Searching • Sorting • Traversing • Merging	Need : 2M, Operations : 2M
	c	**Describe binary search algorithm. Give example to search an element using binary search.**	**4 M**
	Ans	1. Binary search algorithm locates the position of an element in a sorted array. 2. Binary search works by comparing an input value to the middle element of the array. 3. The comparison determines whether the element equals the input, less than the input or greater. 4. When the middle element being compared to equals the input the search stops and typically returns the position of the element displaying search is	Description : 2M, Example : 2M

successful.

5. If the middle element is not equal to the input then a comparison is made to determine whether the input is less than or greater than the middle element.

6. Accordingly given input array is divided into two sub arrays, lower subarray containing elements less than the middle element and upper subarray containing elements greater than the middle element.

7. This process continues from step 2-6 till either the input value is found or the search is unsuccessful.

Searching Element in Given List:

List: 23, 12, 5, 29, 10, 65, 55, 70

Pre-condition for Binary search is array elements must be in ascending order. The given list is not sorted.

Sorted List A= {5, 10, 12, 23, 29, 55, 65, 70}

Search element (k) = 65

i. Low=0, high=7

Mid= (0+7)/2 = 3

A[mid] =a [3] =23

65>23

k>a [mid]

ii. Low=mid+1, High=7

Mid= (4+7)/2=5

A[mid] =a [5] =29

65>29

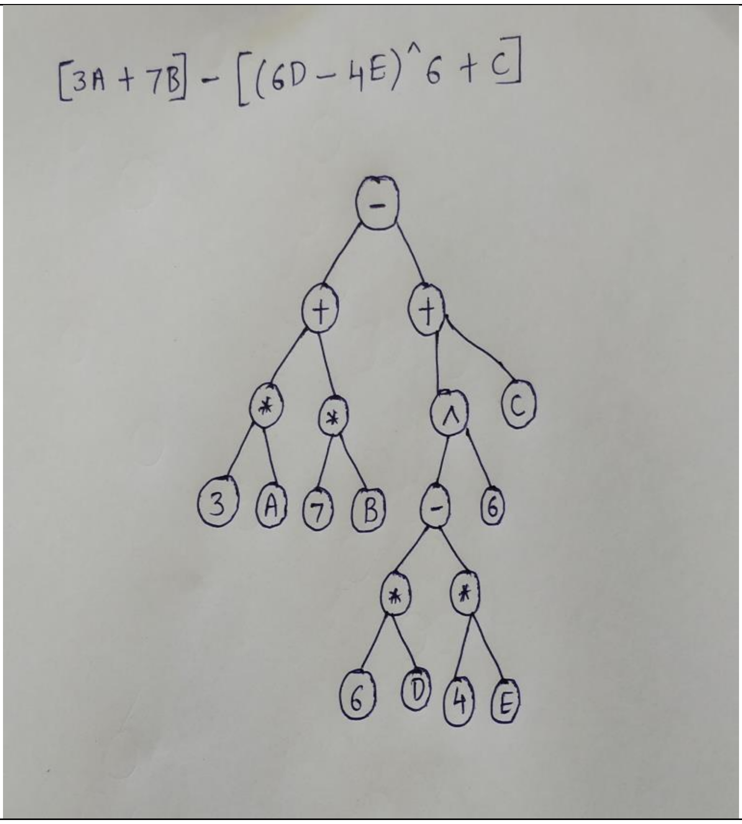
iii. Low=6, high=7

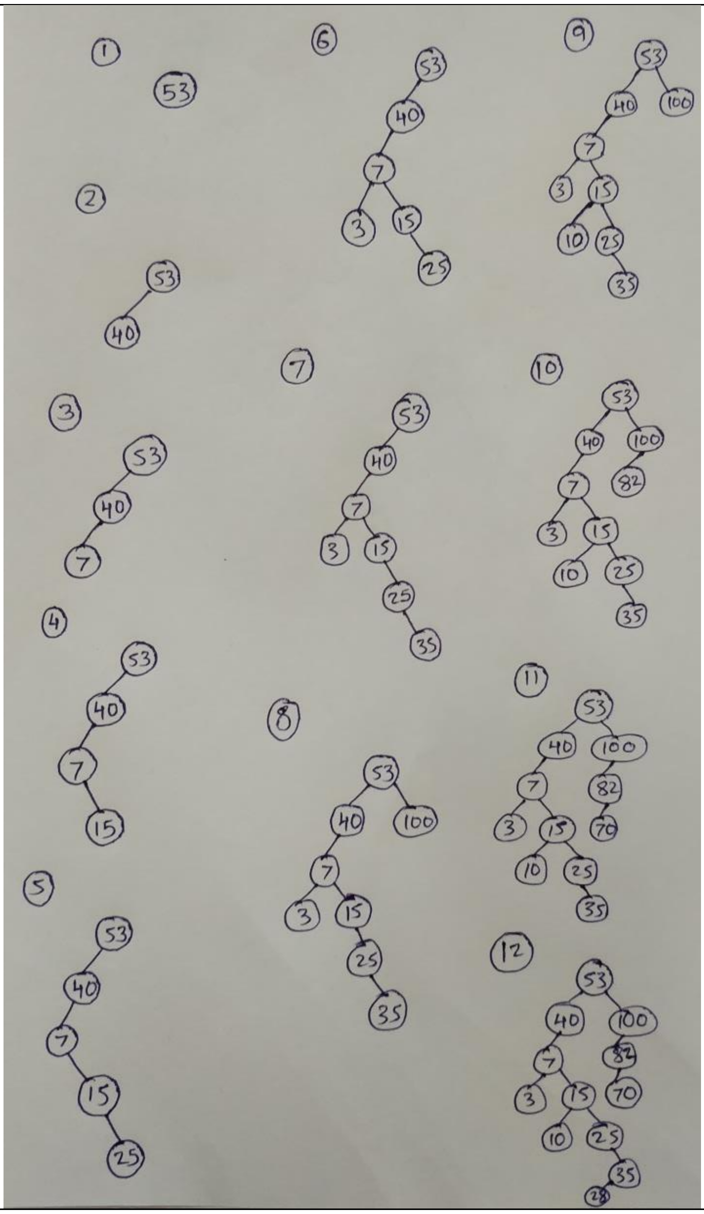
Mid= (6+7)/2 = 6

A[mid] =a [6] =65

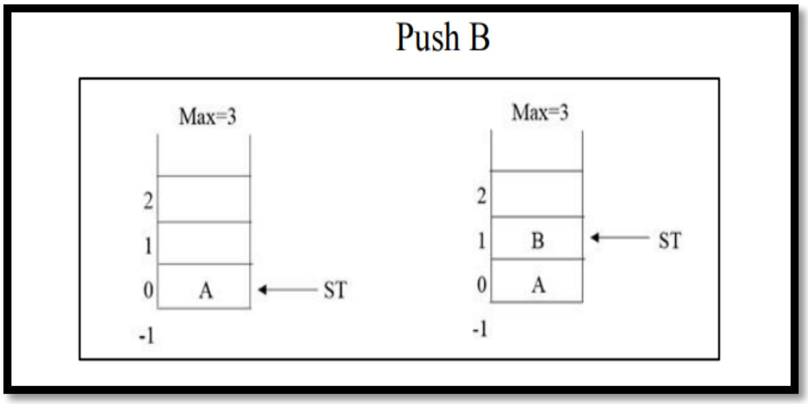
A[mid] =k

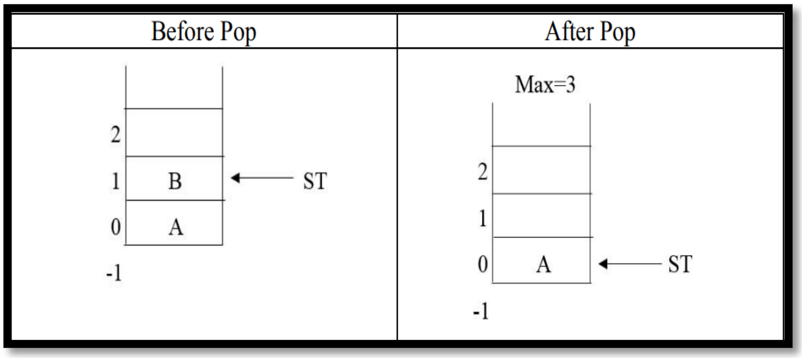
Therefore key element is found at 6th position, no. of comparison required = 3.

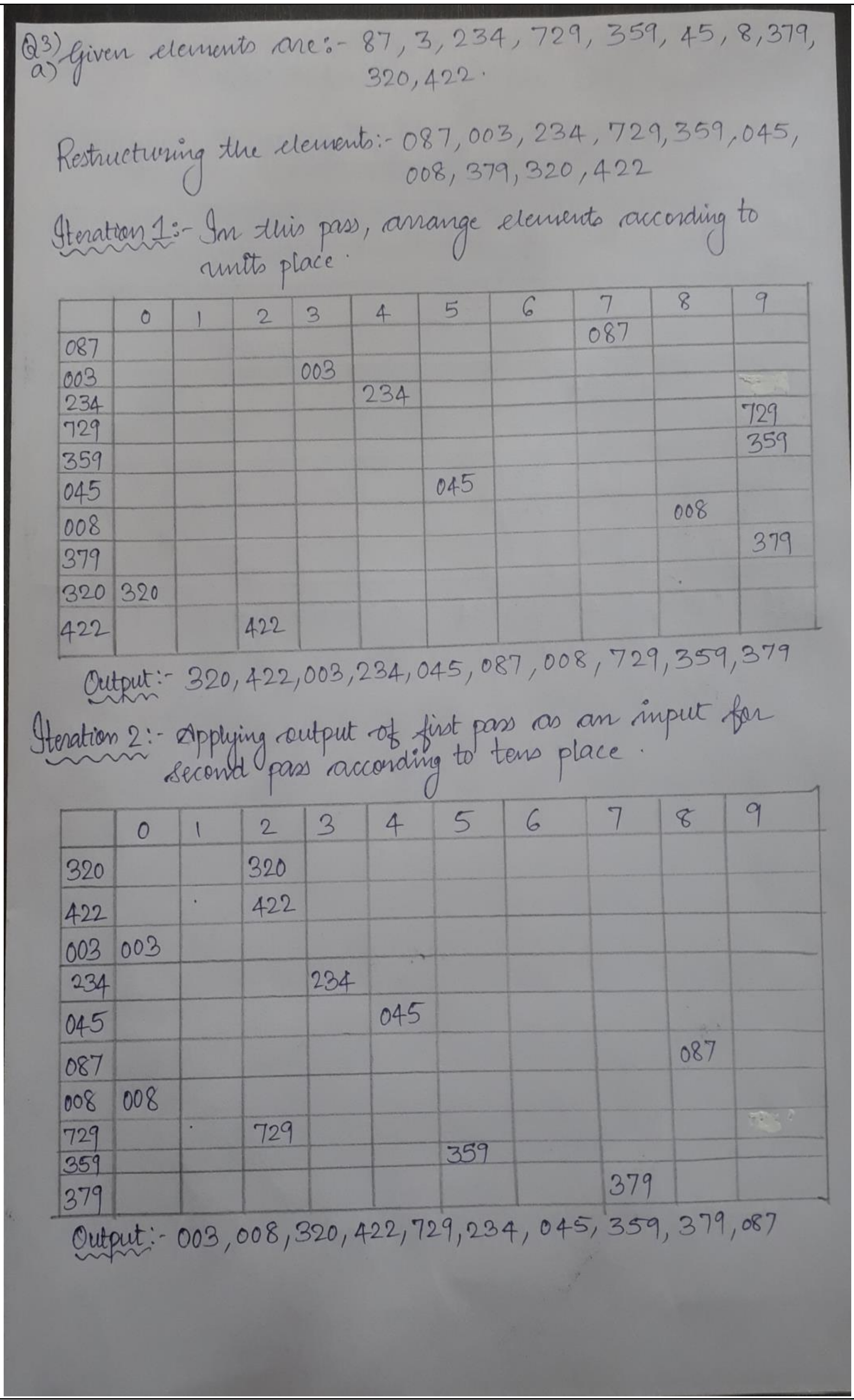
2		**Attempt any FOUR :**	**16 M**
	a	**What is collision resolution techniques? State its types.**	**4 M**
	Ans	Collision Resolution Techniques are the techniques used for resolving or handling the collision. **Types:** 1. Separate Chaining 2. Open Addressing i. Linear probing ii. Quadratic probing iii. Double hashing	Definition : 2M, Types : 2M
	b	**Draw tree structure for following expression:** **[3A + 7B] – [(6D – 4E) ^ 6 + C]**	**4 M**
	Ans	 [3A + 7B] – [(6D – 4E)^6 + C] – + + * * ^ C 3 A 7 B – 6 * * 6 D 4 E	Correct diagram : 4M
	c	**Draw binary search tree using following elements:** **53,40,7,15,25,3,35,100,10,82,70,28**	**4 M**

	Ans	 1 53 2 3 4 5 6 7 8 9 10 11 12 40 7 15 3 25 35 100 10 82 70 28	Correct diagram : 4M
	d	**Describe the following operations on single linked list:** **i. Inserting a new node in a linked list** **ii. Deleting a new node from a linked list**	**4 M**
	Ans	**Inserting node at the beginning:** 1. Create a New Node with two fields as “Data” and “Next”. 2. Store information (data) into “Data Field“. 3. Store address from start node (first node address) in its “Next field”.	For any one type from insertion and deletion : 2M Any other

		4. Make the new node as first node by storing its address start pointer. **Inserting node at the end:** 1. Create a New Node with two fields as “Data” and “Next”. 2. Store information (data) into “Data Field“. 3. Store NULL value in “Next field”. 4. Traverse the list up to the last node (temp). 5. Store address of New Node inside “Next” field of temp node to make the New Node last node in the list. **Inserting node after a position:** 1. Traverse the Linked list up to position-1 nodes. 2. Once all the position-1 nodes are traversed, allocate memory and the given data to the new node. 3. Point the next pointer of the new node to the next of current node. 4. Point the next pointer of current node to the new node. **Delete a node from the beginning:** 1. Create temporary node ‘temp’. 2. Assign address of first node to ‘temp’ pointer. 3. Store address of second node (temp->next) in header pointer ‘start’. 4. Free temp. **Delete a node from in between position:** 1. Create temporary node ‘temp’, ’q’. 2. Assign address of first node to ‘temp’ pointer. 3. Traverse list up to previous node of node to be deleted. 4. Mark the node to be deleted ‘q’. 5. Store address from node ‘q’ in address field of ‘temp’ node (Temp->next=q->next). 6. Free q	relevant description can be considered .

		Delete a node from the end: 1. Create temporary node 'temp','q'. 2. Assign address of first node to 'temp' pointer. 3. Traverse list up to second last node. 4. Mark last node's address in node 'q'. 5. Store NULL value in address field of second last node (temp->next). 6. Free q	
	e	**Write any four applications of Queue.**	**4 M**
	Ans	1. Serving requests on a single shared resource, like a printer, CPU task scheduling etc. 2. In real life scenario, Call Center phone systems uses Queues to hold people calling them in an order, until a service representative is free. 3. Handling of interrupts in real-time systems. 4. Queue of packets in data communication.	Each : 1M
	f	**Describe PUSH and POP operation on stack.**	**4 M**
	Ans	**push()** : This operation is used to insert an element in a stack. Inserting an element in stack requires increment of stack top by one position and then store data element in it.  Push B Max=3 2 1 0 A ST -1 Max=3 2 1 B ST 0 A -1 **pop():** This operation is used to remove an element from stack. Removing an element from stack requires decrement of stack top by one position.	Each : 2M

		 Before Pop After Pop 2 1 B ST 0 A -1 Max=3 2 1 0 A ST -1	
3		**Attempt any FOUR :**	**16 M**
	a	**Sort following elements by Radix sort algorithm:** **87, 3, 234, 729, 359, 45, 8, 379, 320, 422.**	**4 M**

	Ans				Correct answer : 4M

Q3) a) Given elements are:- 87, 3, 234, 729, 359, 45, 8, 379, 320, 422.

Restructuring the elements:- 087, 003, 234, 729, 359, 045, 008, 379, 320, 422

Iteration 1:- In this pass, arrange elements according to units place.

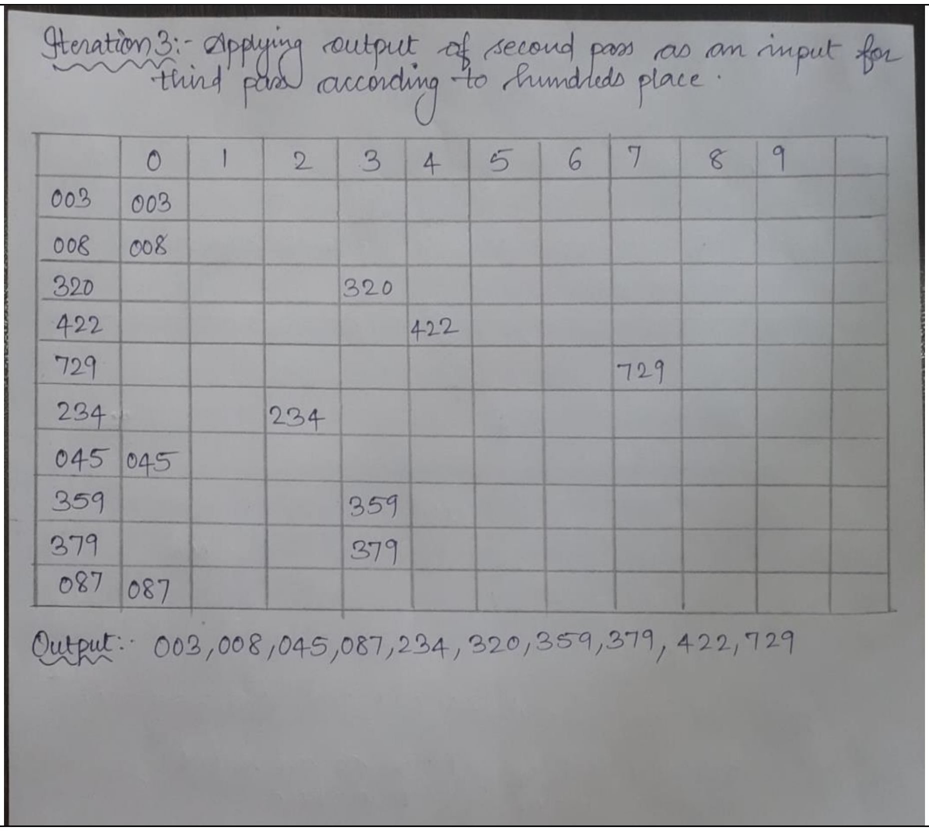
	0	1	2	3	4	5	6	7	8	9
087								087		
003				003						
234					234					
729										729
359										359
045						045				
008									008	
379										379
320	320									
422			422							

Output:- 320, 422, 003, 234, 045, 087, 008, 729, 359, 379

Iteration 2:- Applying output of first pass as an input for second pass according to tens place.

	0	1	2	3	4	5	6	7	8	9
320			320							
422			422							
003	003									
234				234						
045					045					
087									087	
008	008									
729			729							
359						359				
379								379		

Output:- 003, 008, 320, 422, 729, 234, 045, 359, 379, 087



Iteration 3:- Applying output of second pass as an input for third pass according to hundreds place.

	0	1	2	3	4	5	6	7	8	9	
003	003										
008	008										
320				320							
422					422						
729								729			
234			234								
045	045										
359				359							
379				379							
087	087										

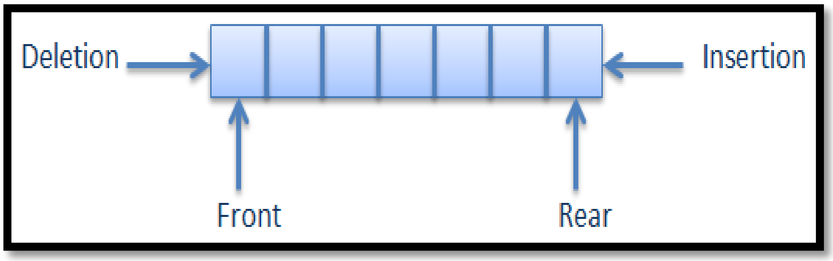
Output:- 003,008,045,087,234,320,359,379, 422,729

	b	**Write program to search an element in an array. Display position of element.**	**4 M**
	Ans	See program below	Correct logic : 2M Syntax 2M

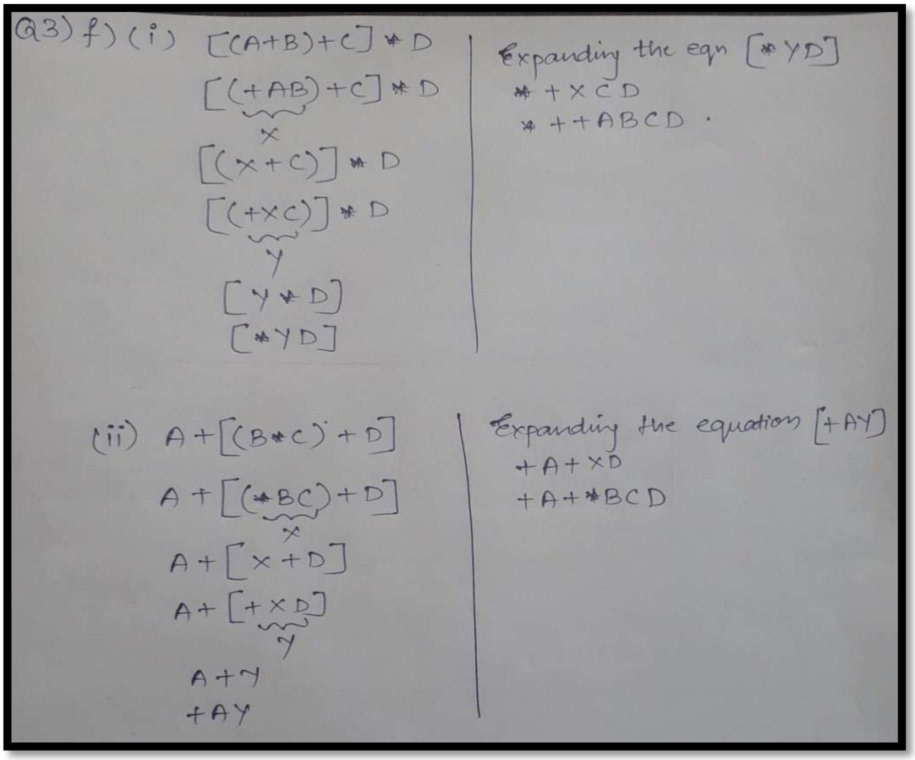
```
#include<stdio.h>
#include<conio.h>
void main()
{
int x,k,arr[100],i,j,high,low,mid,found=0;
clrscr();
printf("\nEnter number of array elements:");
scanf("%d",&x);
for(i=0;i<x;i++)
{
printf("Enter element number %d: \t",i+1);
scanf("%d",&arr[i]);
}
for(i=0;i<x;i++)
{
for(j=0;j<x-1;j++)
{
if(arr[j]>arr[j+1])
{
int t;
t=arr[j];
arr[j]=arr[j+1];
arr[j+1]=t;
}
}
}
printf("\n Enter element to be searched:");
scanf("%d",&k);
low=0;
high=x-1;
while(low<=high)
{
mid=(low+high)/2;
if(arr[mid]==k)
{
printf("\n Element %d found at position %d",k,mid+1);
found=1;
break;
}
else
{
if(arr[mid]>k)
high=mid-1;
else
low=mid+1;
}
}
 if(found!=1)
{
 printf("Not found");
}
getch();
}
```

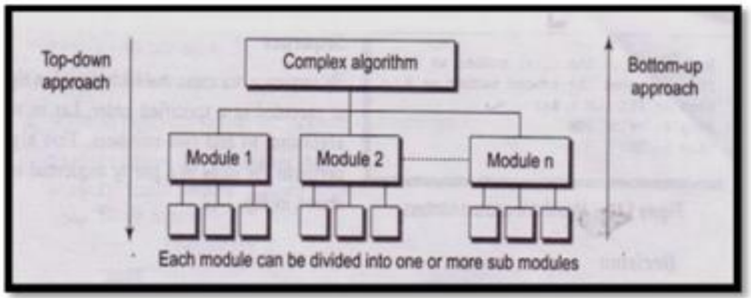
Output:

```
Enter number of array elements:10
Enter element number 1: 10
Enter element number 2:  0
Enter element number 3:  3
Enter element number 4:  2
Enter element number 5:  5
Enter element number 6:  6
Enter element number 7:  4
Enter element number 8:  1
Enter element number 9:  8
```

		Enter element number 10:7 Enter element to be searched: 2 Element 2 found at position 3	
	c	**Explain Queue as an abstract data type.**	**4 M**
	Ans	• Queue is a linear data structure in which the insertion and deletion operations are performed at two different ends. • In a queue data structure, adding and removing of elements are performed at two different positions. • The insertion is performed at one end and deletion is performed at other end. • In a queue data structure, the insertion operation is performed at a position which is known as 'rear' and the deletion operation is performed at a position which is known as 'front'. • In queue data structure, the insertion and deletion operations are performed based on FIFO (First In First Out) principle. 	Correct explanation : 4M
	d	**Explain Binary tree traversal. Write algorithm for following :** **(i) In-order traversal** **(ii) Pre-order traversal** **(iii) Post-order traversal**	**4 M**
	Ans	A binary tree is a tree in which each non-leaf node from the tree has maximum two child nodes. Each node can have one child, two child nodes or no child node in a binary tree. A child node on a left side of parent node is called as left child node. A child node on a right side of parent node is called as right child node. Often we wish to process a binary tree by "visiting" each of its nodes, each time performing a specific action such as printing the contents of the node. Any process for visiting all of the nodes in some order is called a traversal. Any traversal that lists every node in the tree exactly once is called an enumeration of the tree's nodes. **In-order traversal:** The algorithm works by: 1. Traversing the left sub-tree 2. Visiting the root node, and finally	Explanation of Binary tree traversal 1M Algorithm of each traversal 1M

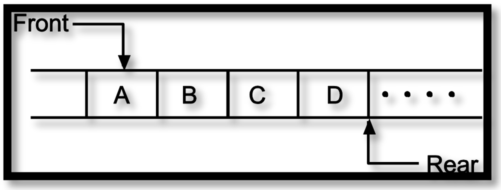
		3. Traversing the right sub-tree. **OR** INORDER(ROOT) Step 1: Repeat Steps 2 to 4 while (ROOT != NULL) Step 2: INORDER(ROOT-> LEFT) Step 3: Write ROOT->DATA Step 4: INORDER(ROOT-> RIGHT) [END OF LOOP] Step 5: END **Pre-order traversal:** The algorithm works by: 1. Visiting the root node, 2. Traversing the left sub-tree, and finally 3. Traversing the right sub-tree. **OR** Step 1: Repeat Steps 2 to 4 while TREE != NULL Step 2: Write TREE DATA Step 3: PREORDER(TREE LEFT) Step 4: PREORDER(TREE RIGHT) [END OF LOOP] Step 5: END **Post-order traversal:** The algorithm works by: 1. Traversing the left sub-tree, 2. Traversing the right sub-tree, and finally 3. Visiting the root node. **OR** Step 1: Repeat Steps 2 to 4 while TREE != NULL Step 2: POSTORDER(TREE LEFT) Step 3: POSTORDER(TREE RIGHT) Step 4: Write TREE DATA [END OF LOOP] Step 5: END	
	e	**Write an algorithm to search an element from single linked list.**	**4 M**
	Ans	Step 1: [INITIALIZE] SET PTR = START Step 2: Repeat Step 3 while PTR != NULL Step 3: IF VAL = PTR -> DATA SET POS = PTR Go To Step 5 ELSE SET PTR = PTR -> NEXT [END OF IF] [END OF LOOP]	Correct algorithm : 4M

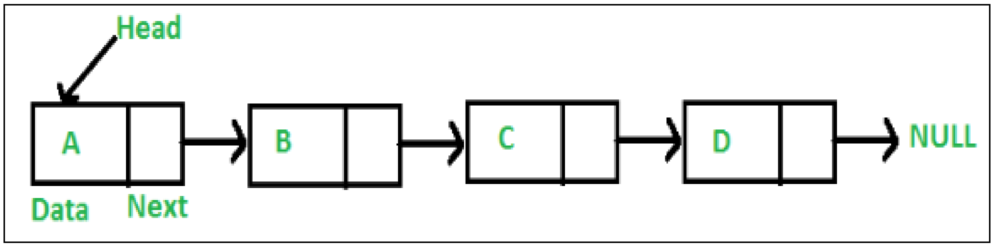
		Step 4: SET POS = NULL Step 5: EXIT	
	f	**Find out prefix equivalent of following expressions :** **(i) [(A + B) + C] * D** **(ii) A + [(B * C) + D]**	**4 M**
	Ans	 Q3) f) (i) $[(A+B)+C] * D$ $[(\underbrace{+AB}_{x})+C] * D$ $[(x+C)] * D$ $[(\underbrace{+xC}_{y})] * D$ $[y * D]$ $[* y D]$ Expanding the eqn $[* y D]$ $* + x C D$ $* + + A B C D$. (ii) $A+[(B*C)+D]$ $A+[(\underbrace{*BC}_{x})+D]$ $A+[x+D]$ $A+[\underbrace{+xD}_{y}]$ $A+y$ $+Ay$ Expanding the equation $[+Ay]$ $+A+xD$ $+A+*BCD$	Correct output : 2M each
4		**Attempt any FOUR :**	**16 M**
	a	**Explain different approaches for designing an algorithm.**	**4 M**
	Ans	Approaches to design an algorithm: 1. Top-down approach 2. Bottom-up approach **Top-down approach:** a. A top-down design approach starts by dividing complex algorithm into one or more modules or subsystems b. Each subsystem is then refined in yet greater detail, sometimes in many additional sub system levels, until the entire specification is reduced to base elements. c. Top-down design method is a form stepwise refinement where we begin with the Top most modules and incrementally add modules that it calls. **2. Bottom-up approach:** a. In this approach the individual base elements of the system are first specified in detail. b. These elements are then linked together to form larger subsystems, which then in turn are clubbed in many levels, until a complete top-level system is	For Top down Approach Explanation 2M For Bottom Up Approach explanation 2M

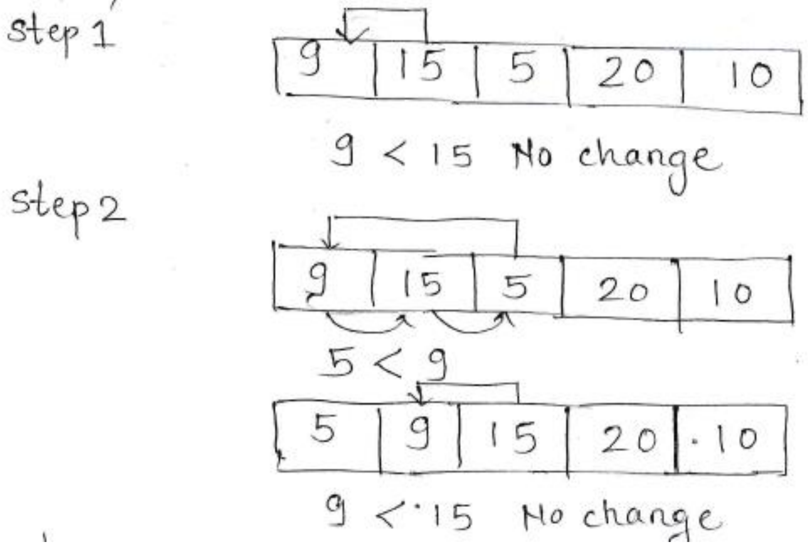
		formed.  Top-down approach Complex algorithm Bottom-up approach Module 1 Module 2 Module n Each module can be divided into one or more sub modules	
	b	**Write a program to find factorial of a number using recursion.**	**4 M**

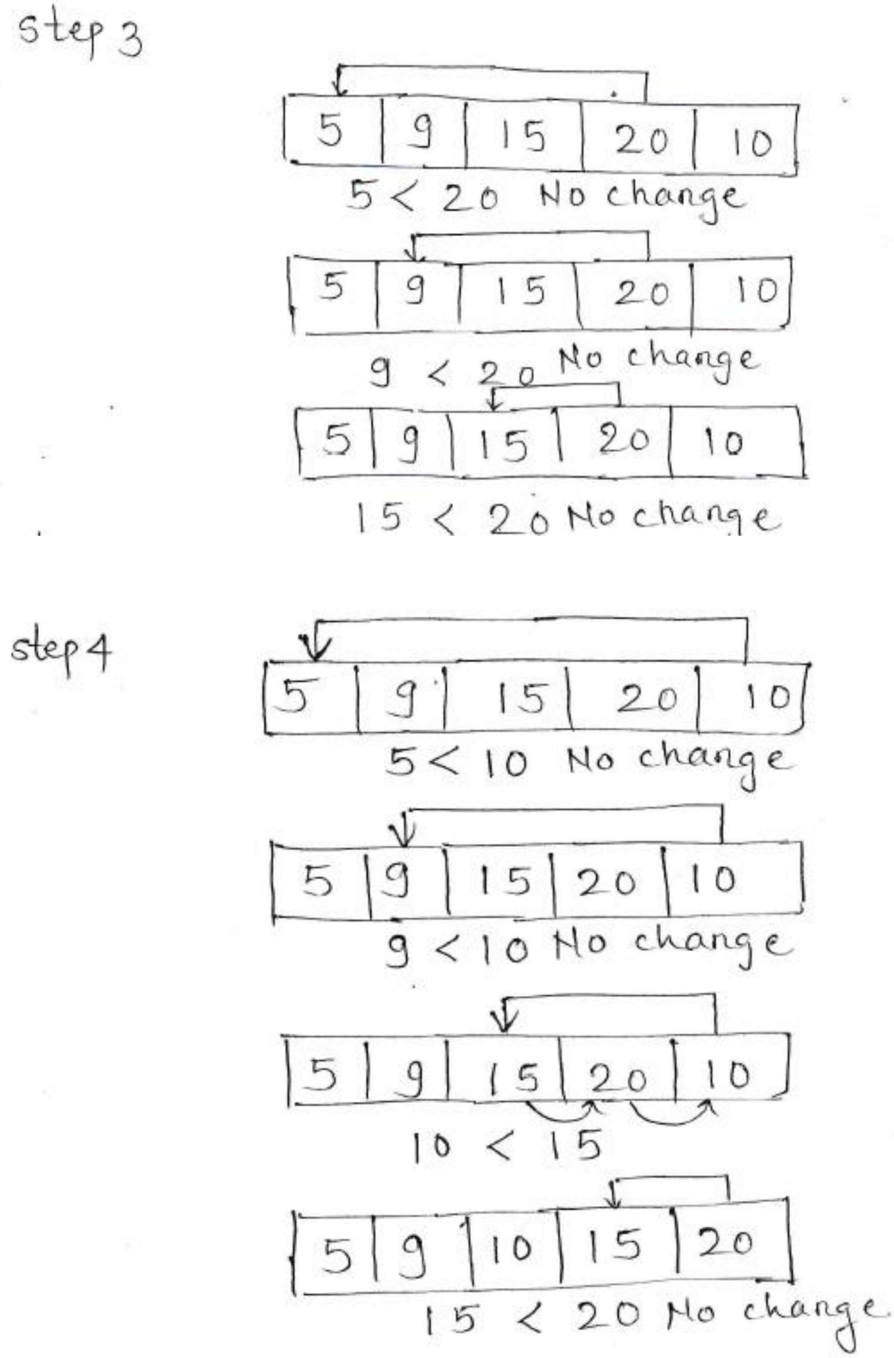
	Ans	#include <stdio.h>
		#include <conio.h>
		intrec(int x);
		void main()
		{
		inta, fact;
		printf("Enter a number to calculate its factorial:");
		scanf("%d", &a);
		fact = rec(a);
		printf("Factorial of %d = %d\n", fact);
		}
		int rec(int x)
		{
		int f;
		if(x = = 1)
		return 1;
		else
		f = x*rec(x-1);
		return(f);
		}
		
		Output:
		Enter a number to calculate its factorial: 5
		Factorial of 5 = 120
			Correct logic : 2M
			Syntax 2 M

	c	**Define Queue. Write an algorithm to delete an element from a queue.**	**4 M**
	Ans	A queue is an ordered collection of items from which items may be deleted at one end (called the front of the queue) and into which items may be inserted at the other end (called the rear of the queue). A queue is logically a First in First Out (FIFO) type of list. Queue means a line, for example, at railway reservation booth; we have to get into the reservation queue.	Definition : 2M, Algorithm : 2M

		 **Algorithm to delete an element from a queue:** Step 1: IF FRONT = -1 OR FRONT > REAR Write UNDERFLOW ELSE SET VAL = QUEUE[FRONT] SET FRONT = FRONT+1 [END OF IF] Step 2: EXIT	
	d	**Write an algorithm to insert and delete element from circular linked list.**	**4 M**
	Ans	**Inserting element in circular linked list:** Step 1: IF AVAIL = NULL Write OVERFLOW Go to Step 11 [END OF IF] Step 2: SET NEW_NODE = AVAIL Step 3: SET AVAIL = AVAIL -> NEXT Step 4: SET NEW_NODE -> DATA = VAL Step 5: SET PTR = START Step 6: Repeat Step 7 while PTR NEXT != START Step 7: PTR = PTR -> NEXT [END OF LOOP] Step 8: SET NEW_NODE -> NEXT = START Step 9: SET PTR -> NEXT = NEW_NODE Step 10: SET START = NEW_NODE Step 11: EXIT **Deleting element from circular linked list:** Step 1: IF START = NULL Write UNDERFLOW Go to Step 8 [END OF IF] Step 2: SET PTR = START Step 3: Repeat Step 4 while PTR -> NEXT != START Step 4: SET PTR = PTR -> NEXT [END OF LOOP] Step 5: SET PTR -> NEXT = START -> NEXT Step 6: FREE START Step 7: SET START = PTR -> NEXT Step 8: EXIT	Insertion : 2M, Deletion : 2M OR Any other correct algorithm can be considered

	e	**Explain linked list as an abstract data structure.**	**4 M**
	Ans	**Linked List:** A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers.  **Operations performed on Linked List are:** Create () – Creation of node Destroy () – Deletion of node Add (element) – Adding element in the list Remove (element) – Removing element from the list Traverse () – Changing the position of the element from the list Is Empty () – Checking if the node is empty. Search(element) – Searching for the intended element	Diagram: 1 M, Explanation: 3 M
	f	**Describe general tree and binary tree.**	**4 M**
	Ans	**General Tree:** • General trees are data structures that store elements hierarchically. • The top node of a tree is the root node and each node, except the root, has a parent. • A node in a general tree (except the leaf nodes) may have zero or more sub-trees. • General trees which have 3 sub-trees per node are called ternary trees. • However, the number of sub-trees for any node may be variable. • For example, a node can have 1 sub-tree, whereas some other node can have 3 sub-trees. **Binary Tree:** • A binary tree is a data structure that is defined as a collection of elements called nodes. In a binary tree, the topmost element is called the root node, and each node has 0, 1, or at the most 2 children. • A node that has zero children is called a leaf node or a terminal node. • Every node contains a data element, a left pointer which points to the left child, and a right pointer which points to the right child. • The root element is pointed by a 'root' pointer.	General tree : 2M, Binary tree : 2M

		• If root = NULL, then it means the tree is empty.	
5		**Attempt any TWO :**	**16 M**
	a	**Write an algorithm for insertion sort and arrange given numbers in ascending order using insertion sort :9, 15, 5, 20, 10**	**8 M**
	Ans	**Algorithm for insertion sort:** **Step 1**:Start **Step 2**: Input array elements **Step 3**: Compare 1st index element with 0th index element. If 0th element is greater than 1st element then store 1st element in temporary variable and shift 0th element to its right by one and then insert temp variable data into 0th index position. **Step 4**: Compare 2nd index element with 0th index element. If 0th element is greater than 2nd element then store 2nd element in temporary variable and shift 1st and 0th elements to their right by one and then insert temp variable data into 0th index position. Then compare 2nd index element with 1st index element and if required perform shift insert operation. **Step 5**: Continue the process of comparison of next index position elements with all numbers stored before it in an array and if required perform shift insert operation. If an array contains N number of elements then this procedure is repeated for N-1 times to get sorted list. step 6:stop **Sorting:-** **Input list: 9,15,5,20,10**  Step 1 9 | 15 | 5 | 20 | 10 9 < 15 No change Step 2 9 | 15 | 5 | 20 | 10 5 < 9 5 | 9 | 15 | 20 | 10 9 < 15 No change	Correct algorithm : 4M, Sorting steps: 4M Note: Any correct logical sequence of steps shall be considered as an algorithm

		 step 3 5 | 9 | 15 | 20 | 10 5< 20 No change 5 | 9 | 15 | 20 | 10 9 < 20 No change 5 | 9 | 15 | 20 | 10 15 < 20 No change step 4 5 | 9 | 15 | 20 | 10 5< 10 No change 5 | 9 | 15 | 20 | 10 9 < 10 No change 5 | 9 | 15 | 20 | 10 10 < 15 5 | 9 | 10 | 15 | 20 15 < 20 No change **Sorted list: 5,9,10,15,20**	
	b	**Convert given infix expression to postfix expression using stack** **(A + B) * D + E / (F + A * D) + C**	**8 M**
	Ans		Correct postfix expression : 8M

Infix expression	Read character	Stack contents	Postfix expression
(A+B)*D+E/(F+A*D)+C	(	(	-
A+B)*D+E/(F+A*D)+C	A	(	A
+B)*D+E/(F+A*D)+C	+	(+	A
B)*D+E/(F+A*D)+C	B	(+	AB
)*D+E/(F+A*D)+C	)		AB+
*D+E/(F+A*D)+C	*	*	AB+
D+E/(F+A*D)+C	D	*	AB+D
+E/(F+A*D)+C	+	+	AB+D*
E/(F+A*D)+C	E	+	AB+D*E
/(F+A*D)+C	/	+/	AB+D*E
(F+A*D)+C	(	+/(	AB+D*E
F+A*D)+C	F	+/(	AB+D*EF
+A*D)+C	+	+/(+	AB+D*EF
A*D)+C	A	+/(+	AB+D*EFA
*D)+C	*	+/(+*	AB+D*EFA
D)+C	D	+/(+*	AB+D*EFAD
)+C	)	+/	AB+D*EFAD*+
+C	+	+	AB+D*EFAD*+/+
C	C	+	AB+D*EFAD*+/+C
			AB+D*EFAD*+/+C+

	c	**Explain BFS with suitable example.**	**8 M**
	Ans	**Algorithm for BFS**: Step 1. Initialize all nodes to ready state. Step 2. Insert starting node in a queue and change its state to waiting state.	Explanation algorithm

Step 3. Repeat steps 4 to 6 till the queue becomes empty.

Step 4. Remove front node N from queue and change its status to visited. Add this node N to list of reachable nodes and add its origin to origin list.

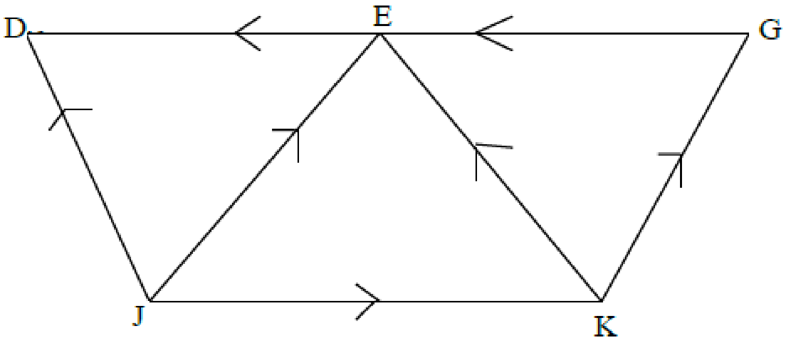
Step 5. Insert all adjacent nodes of N at the rear end of the queue and change their status to waiting state.

Step 6. From the origin find path from source node to destination node or from the queue element list.

Step 7. Stop.

: 4M, Example : 4M

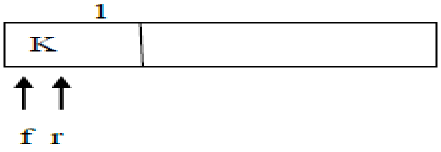
Example: - Consider following graph G. Find all nodes reachable from node K to D.



All nodes are in ready state.

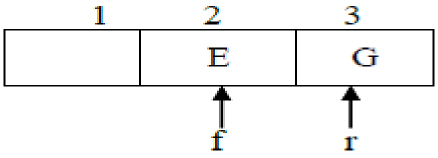
Step 1- Initially insert node K in a queue and add NULL to origin

Front = 1 queue = K
Rear = 1 origin = O



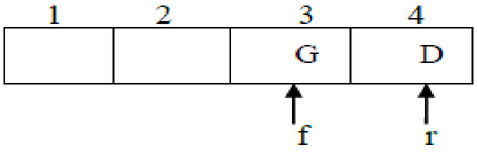
Step 2 – Remove front element K and change its states to visited. Find its adjacent nodes and insert them into queue if their state is ready.

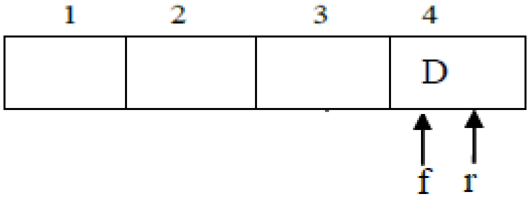
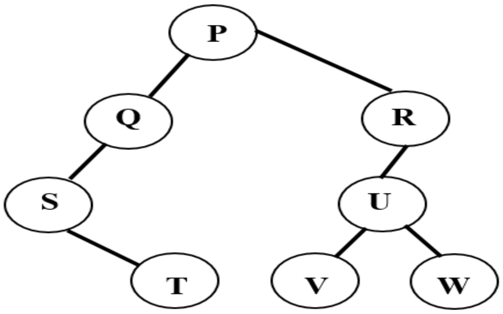
Front = 2 queue = K, E, G
Rear = 3 origin = O, K, K



Step 3 – Remove front element E and change its states to visited. Find its adjacent nodes and insert them into queue if their state is ready.

Front = 3 queue = K, E, G, D
Rear = 4 origin = O, K, K, E



		Step 4 – Remove front element G and change its states to visit. Find its adjacent nodes and insert them into queue if their state is ready. E is already visited. Front = 4 queue = K, E, G, D Rear = 4 origin = O, K, K, E  **Step 5 -** Remove front element D and change its states to visited. Find its adjacent nodes and insert them into queue if their state is ready. Node D does not have any adjacent node. **Path between K to D is K -> E -> D**	
6		**Attempt any TWO :**	**16 M**
	a	**Describe the process of pre-order traversal and post-order traversal of a binary tree with suitable example.**	**8 M**
	Ans	**Preorder traversal**: - In preorder traversal method, first visit root element, then visit left sub tree and then visit right sub tree. It uses recursive function call to perform preorder traversal while visiting left and right tree. Procedure:- Step 1: Visit root node Step 2: Visit left sub tree in preorder Step 3: Visit right sub tree in preorder **Postorder traversal**:-In postorder traversal method, first visit process left sub tree, then visit right sub tree and then visit the root element. It uses recursive function call to perform postorder traversal while visiting left and right tree. Procedure:- Step 1: Visit left sub tree in postorder Step 2: Visit right sub tree in postorder Step 3: Visit root node **Example:-** 	Each traversal-description : 2M, Example : 2M

Tree traversal:-

Preorder:- P Q S T R U V W

Postorder:- T S Q V W U R P

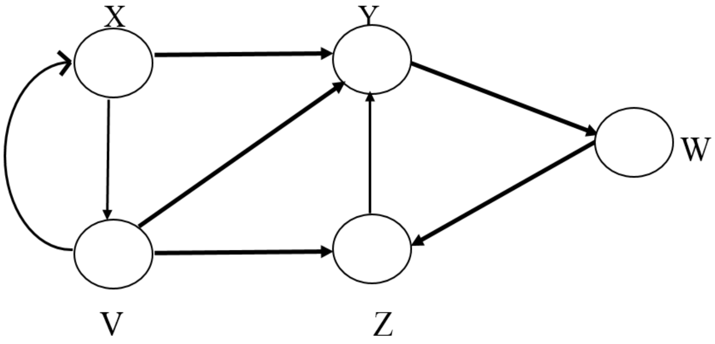
b **Explain sequential and linked representation of graph with suitable example.** **8 M**

Ans **1. Sequential representation:** It is represented using adjacency Matrix. It is a 2D array of size V x V where V is the number of vertices in a graph. It is also called as bit matrix as it contains only two values i.e. 1 and 0.Value 1 indicates that there is an edge from vertex i to vertex j. Value 0 indicates that there is no edge from vertex i to vertex j or vice versa. Adjacency matrix for undirected graph is always symmetric.

Each representation- explanation : 2M, Example : 2M

2. Linked representation: It is represented with adjacency list. Adjacency list contains two columns as vertex name and adjacent vertices. It shows who all adjacent vertices of each vertex in a graph are. With the help this list linked representation is done. Each node from a graph is represented as a node with three fields, one is the data, second is pointer to next node in list and third pointer is used to point to its adjacent node.

Example:

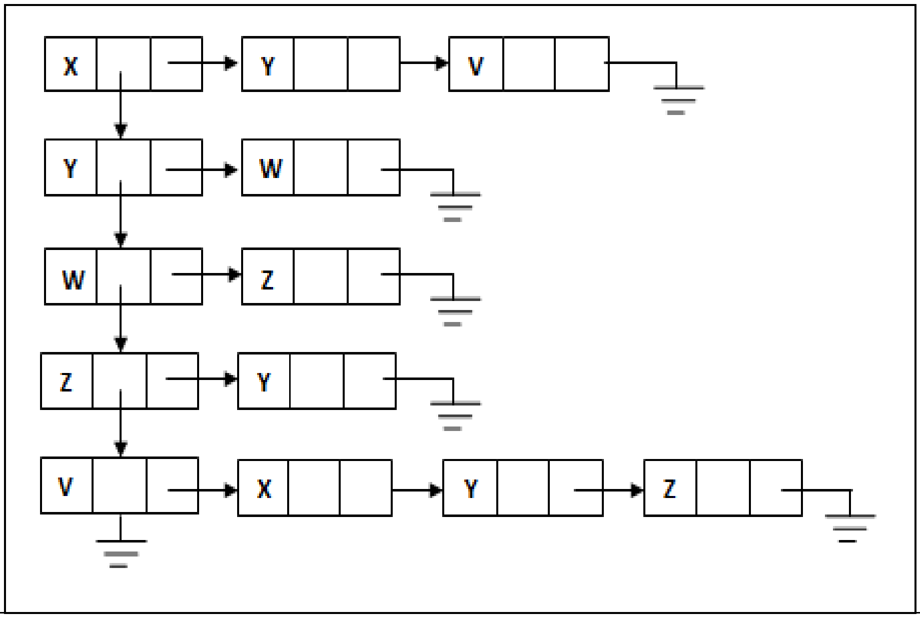


Sequential representation- Adjacency Matrix

	X	Y	W	Z	V
X	0	1	0	0	1
Y	0	0	1	0	0
W	0	0	0	1	0
Z	0	1	0	0	0
V	1	1	0	1	0

Linked representation: Adjacency list

Node	Adjacency List
X	Y,V
Y	W
W	Z
Z	Y
V	X,Y,Z



c **State principle of stack and evaluate following post-fix expression using stack.**
6, 2, 3, +, -, 3, 8, 2, 1, +, *, 2∅3, +

8 M

Ans **Principle:-**

Stack is a linear data structure in which all the elements are stored in sequence.

Stack works on LIFO principle.**LIFO is Last In First Outie**. An element inserted at last in a stack is removed first from it.

As stack has only one end i.e. stack top it inserts and removes elements only from top so it works on LIFO principle

Principle : 2M, Correct answer for evaluation : 6M

Evaluation:-

Step No	**Postfix expression**	**Read char acter**	**Stack contents**	**Evaluation**
1	**6,2,3,+,-,3,8,2,1,+,*,2,∅,3,+**	**6**	**6**	
2	**2,3,+,-,3,8,2,1,+,*,2,∅,3,+**	**2**	**6,2**	
3	**3,+,-,3,8,2,1,+,*,2,∅,3,+**	**3**	**6,2,3**	
4	**,+,-,3,8,2,1,+,*,2,∅,3,+**	**+**	**6,5**	**2+3=5**
5	**,-,3,8,2,1,+,*,2,∅,3,+**	**-**	**1**	**6-5=1**
6	**3,8,2,1,+,*,2,∅,3,+**	**3**	**1,3**	
7	**8,2,1,+,*,2,∅,3,+**	**8**	**1,3,8**	
8	**2,1,+,*,2,∅,3,+**	**2**	**1,3,8,2**	
9	**1,+,*,2,∅,3,+**	**1**	**1,3,8,2,1**	
10	**+,*,2,∅,3,+**	**+**	**1,3,8,3**	**2+1=3**
11	***,2,∅,3,+**	*****	**1,3,24**	**8*3=24**
12	**2,∅,3,+**	**2**	**1,3,24,2**	
13	**∅,3,+**	**∅**		
14	**3,+**	**3**		
15	**+**	**+**		

Note: Partial Marks shall be awarded as per the evaluation