WINTER– 18 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 judgement 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 :**	**12 M**
	a	**Define data structure. Enlist any two operations on it.**	2 M
	Ans	**Data structure:** A Data structure is logical and mathematical model used for storing and organizing data in particular way so that it can be accessed efficiently. **OR** **Data structure** is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data. Operations perform on data structure: **1.** Insertion **2.** Deletion **3.** Searching **4.** Sorting **5.** Traversing **6.** Merging	Definition: 1M, any two operations: 1M

	b	**Define sorting. Write its types.**	2 M
	Ans	**Sorting** is the process of placing elements from a collection in specific logical order. For example, a list of words could be sorted alphabetically or by length. **Types:** 1. Bubble sort 2. Selection sort 3. Insertion sort 4. Quick sort 5. Radix sort 6. Merge sort 7. Shell sort	Definition: 1M, any two types: 1M
	c	**Write any two applications of stack.**	2 M
	Ans	1. Reversing a list 2. Conversion of infix expression to postfix expression 3. Evaluation of postfix expression 4. Conversion of infix expression to prefix expression 5. Evaluation of prefix expression 6. Recursion	Each application: 1M
	d	**Write any four primitive operations on queue.**	2 M
	Ans	1. **enqueue** − add (store) an item to the queue from rear end 2. **dequeue** − remove an item from the queue from front end. 3. **peek** − Gets the element at the front of the queue without removing it. 4. **isfull** − Checks if the queue is full. 5. **isempty** − Checks if the queue is empty.	Listing each operation: ½ M
	e	**Define the terms NULL pointer and next pointer for linked list.**	2 M
	Ans	**NULL pointer:** It is an address field of the last node in the linked list which specifies end of the list by holding value as a NULL. **Next pointer:** It is one of field of node structure which contains an address of	Each 1M

		the next node.	
	f	**Define binary search tree.**	2 M
	Ans	A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties – • The left sub-tree of a node has a key less than or equal to its parent node's key. • The right sub-tree of a node has a key greater than to its parent node's key.	Correct definition: 2M
	g	**Write any two applications of graph.**	2 M
	Ans	**Applications of graphs:** 1. To represent road map 2. To represent circuit or networks 3. To represent program flow analysis 4. To represent transport network 5. To represent social network 6. Neural networks	Each 1M
	h	**Define the term recursion.**	2 M
	Ans	Recursion is the process of calling function by itself till terminating condition.	Correct definition: 2M
	B	**Attempt any TWO :**	**8 M**
	a	**Define the following terms with respect to tree:** **i) Leaf node** **ii) Degree of node** **iii) Height of tree** **iv) Descendant node**	4 M
	Ans	i. **Leaf node:** A node having no child or of degree zero is called a terminal node or leaf node. ii. **Degree of node**: It is defined as maximum number of child nodes of any node. OR Degree of node is the number of nodes connected to a particular node.	Each 1 M

		iii. **Height of tree:** The longest path from root node to the terminal node is known as Depth of tree or height of tree. iv. **Descendant node:** A descendant refers to any element that is connected lower down the hierarchy tree – no matter how many levels lower. A descendant is a child, grandchild, great-grandchild, and so on.	
	b	**Write a 'program in c' language for selection sort.**	4 M

	Ans	#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int a[100],n,i,j,min,temp;
		clrscr();
		printf("\n Enter the Number of Elements: ");
		scanf("%d",&n);
		printf("\n Enter %d Elements: ",n);
		for(i=0;i<n;i++)
		{
		scanf("%d",&a[i]);
		}
		for(i=0;i<n-1;i++)
		{
		min=i;
		for(j=i+1;j<n;j++)
		{
		if(a[min]>a[j])
		min=j;
		}
		if(min!=i)
		{
			Correct Logic 2M,Correct Syntax 2M
			any other relevant logic can be considered

```
            temp=a[i];
            a[i]=a[min];
            a[min]=temp;
          }
        }
        printf("\n The Sorted array in ascending order: ");
        for(i=0;i<n;i++)
        {
          printf("%d ",a[i]);
        }
        getch();
      }
```

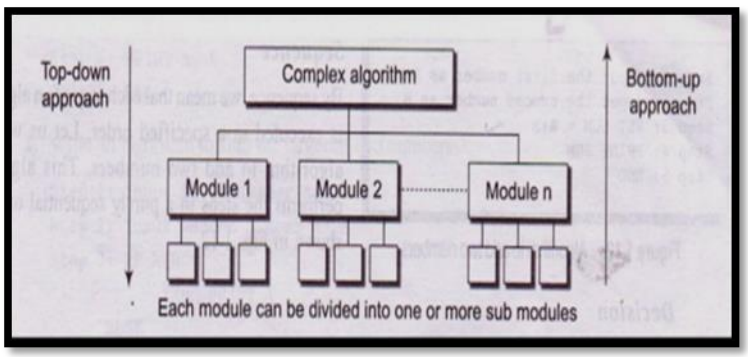
c **Convert the given infix expression into postfix using stack and write down step of conversion** — 4 M

a ↑ b * c – d + e.

Ans

Symbol Scanned	Stack	Expression
(	(	
a	(	a
↑	(↑	a
b	(↑	ab
*	(*	ab↑
c	(*	ab↑c
-	(-	ab↑c*
d	(-	ab↑c*d
+	(+	ab↑c*d-
e	(+	ab↑c*d-e
)	empty	ab↑c*d-e+

Correct answer: 4M, *give appropriate marks for steps

		Ans: a b ↑ c * d – e +	
2		**Attempt any FOUR :**	**16 M**
	a	**Define Algorithm. Describe different approaches for designing an algorithm.**	4 M
	Ans	**Algorithm:** It is sequence of steps/instructions that must be followed to solve a problem. In other words, an algorithm is a logical representation of the steps/instructions which should be executed to perform a meaningful task. 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 formed.  	Definition: 1M, Diagram: 1M, Explanation: 2M

b **Write difference between stack and queue (any 4 points).** — 4 M

Ans

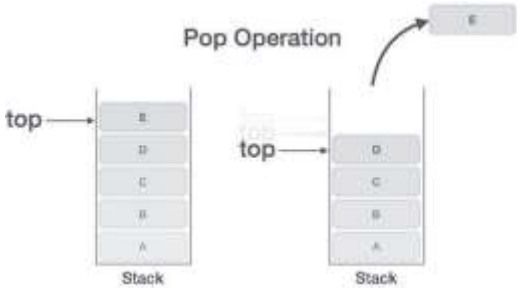
Stack	Queue
1. In Stack insertion and deletion operations are performed at same end.	**1.** In Queue insertion and deletion operations are performed at different end.
2. In stack the element which is inserted last is first to delete so it is called Last In First Out.	**2.** In Queue the element which is inserted first is first to delete so it is called First In First Out.
3. In stack only one pointer is used called as Top.	**3.** In Queue two pointers are used called as front and rear.
4. In Stack Memory is not wasted	**4.** In Queue memory can be wasted/ unusable in case of linear queue.
5. Stack of books is an example of stack	**5.** Students standing in a line at fees counter is an example of queue
6. Application: • Recursion • Polish notation	**6.** Application: • In computer system for organizing processes. • In mobile device for sending and receiving messages

Correct point 1 M

c **Write an algorithm to POP an element from stack.** — 4 M

Ans

POP: The process of deleting an element from top of the stack is called POP operation. As deletion takes place from the top of the stack. After every POP operation the top of the stack is decremented by one.



Algorithm for pop operation:

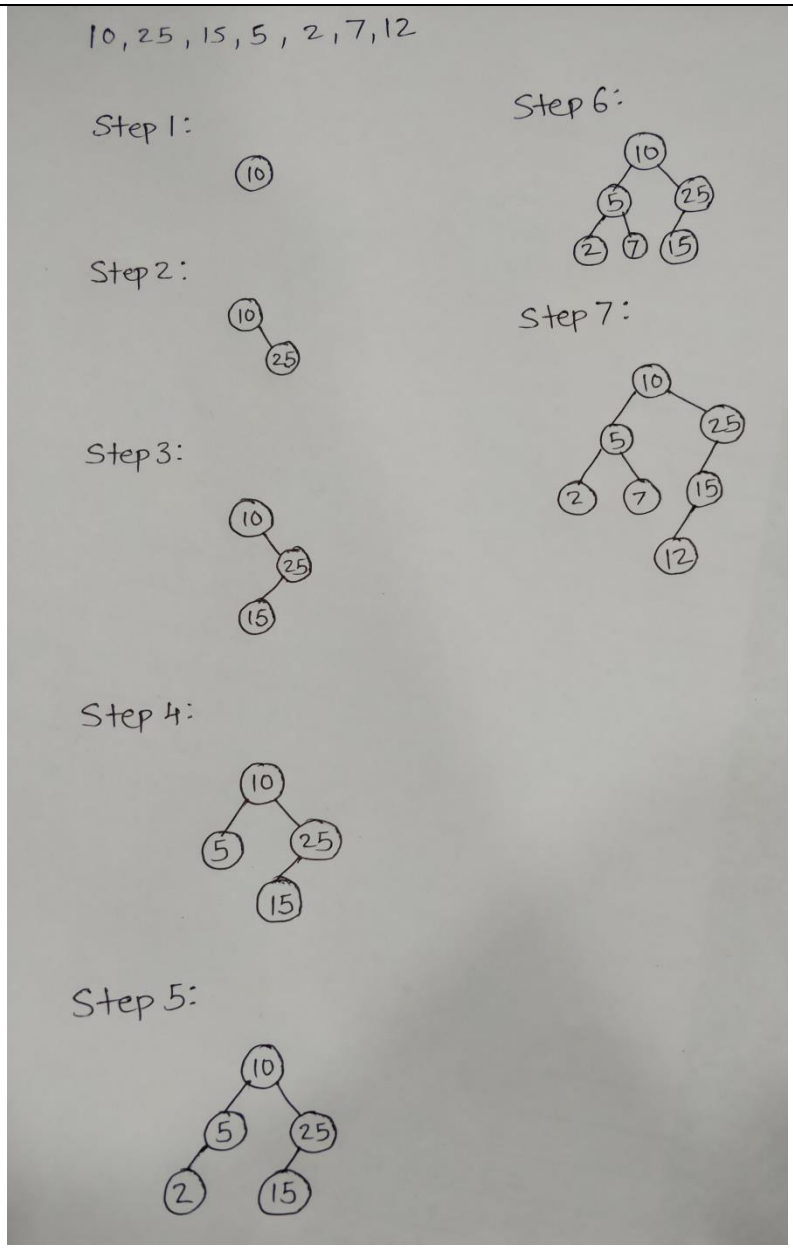
First check for underflow i.e. if top is -1 means there is no element in the stack then pop operation cannot be performed. If there is no underflow then decrement the top by 1. It removes an element placed at top of the stack.

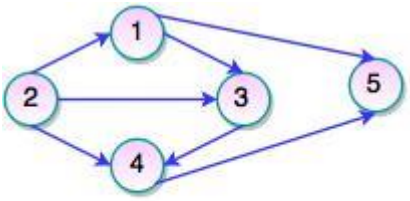
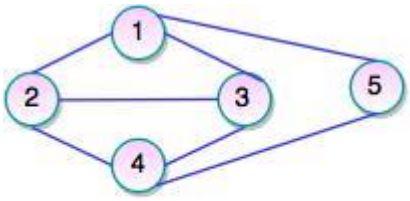
1. If TOP = =-1, then
2. Display "The Stack is empty"

(a) Exit

3. Else remove the Top most elements
4. DATA = STACK [TOP]

Correct algorithm: 4M

		5. TOP = TOP – 1 6. Exit.	
	d	**Create a binary search tree for the following data :** **10, 25, 15, 5, 2, 7, 12**	4 M
	Ans		Correct answer: 4M, give marks to right steps
	e	**Describe directed and undirected graph with suitable example.**	4 M
	Ans	**Directed Graph** If a graph contains ordered pair of vertices, is said to be a Directed Graph. If an edge is represented using a pair of vertices (V1, V2), the edge is said to be	Directed Graph Explanation: 2M Undirected Graph Explanation:

		directed from V1 to V2. The first element of the pair V1 is called the start vertex and the second element of the pair V2 is called the end vertex.  Fig. Directed Graph Set of Vertices V = {1, 2, 3, 4, 5, 5} Set of Edges W = {(1, 3), (1, 5), (2, 1), (2, 3), (2, 4), (3, 4), (4, 5)} **Undirected Graph** If a graph contains unordered pair of vertices, is said to be an Undirected Graph. In this graph, pair of vertices represents the same edge.  Fig. Undirected Graph Set of Vertices V = {1, 2, 3, 4, 5} Set of Edges E = {(1, 2), (1, 3), (1, 5), (2, 1), (2, 3), (2, 4), (3, 4), (4, 5)} In an undirected graph, the nodes are connected by undirected arcs. It is an edge that has no arrow. Both the ends of an undirected arc are equivalent; there is no head or tail.	2M
	f	**Describe binary search with example.**	4 M
	Ans	**Working:** 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 successful. 5. If the middle element is not equal to the input then a comparison is made to	Explanation: 2M, Example: 2M

determine whether the input is less than or greater than the middle element.

6. Accordingly given input array is divided into two subarrays, 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.

Search is successful

3 **Attempt any FOUR :** **16 M**

a **Describe the concept of time complexity with example.** 4 M

Ans

Time Complexity: Time complexity of program or algorithm is amount of computer time that it needs to run to completion. To measure time complexity of an algorithm we concentrate on developing only frequency count for key statements.

Example:

```
#include<stdio.h>
void main ()
{
int i, n, sum, x;
sum=0;
printf("\n Enter no of data to be added");
scanf("% d", &n);
for(i=0 ; i<n; i++)
{
scanf("%d" , &x);
sum=sum+x;
}
printf("\n Sum = %d ", sum);
}
```

Calculation of Computational Time:

Statement	Frequency	Computational Time
sum=0	1	t_1
printf("\n Enter no of data to be added")	1	t_2
scanf("% d", &n)	1	t_3
for(i=0 ; i<n; i++)	n+1	$(n+1)t_4$
scanf("%d" , &x)	n	nt_5
sum=sum+x	n	nt_6
printf("\n Sum = %d ", sum)	1	t_7

Total computational time= t1+t2+t3+(n+1)t4 +nt6+nt5+t7

T= n(t4+t5+t6)+ (t1+t2+t3+t4+t7)

For large n , T can be approximated to

T= n(t4+t5+t6)= kn where k= t4+t5+t6

Thus T = kn or T

Time complexity concept:2M, Example: 2 M

**Note: Any Relevant example shall be considered

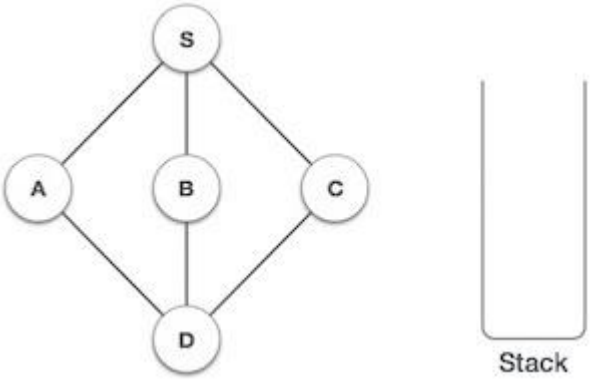
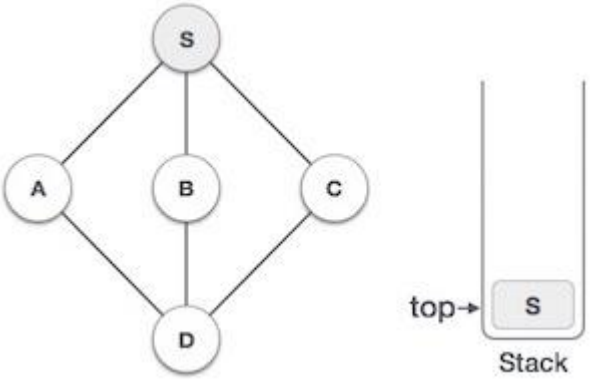
b **Write an algorithm for traversal of graph using DFS (Depth First Search) method.** 4 M

Ans **Depth First Search (DFS):** DFS concept: 1 mark,

The aim of DFS algorithm is to traverse the graph in such a way that it tries to go far from the root node. Stack is used in the implementation of the depth first search. Back tracking used in this algorithm.

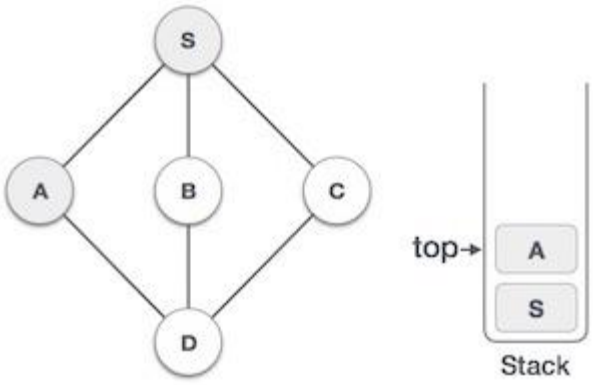
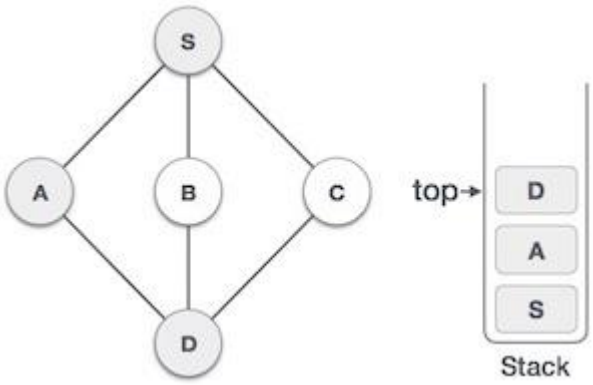
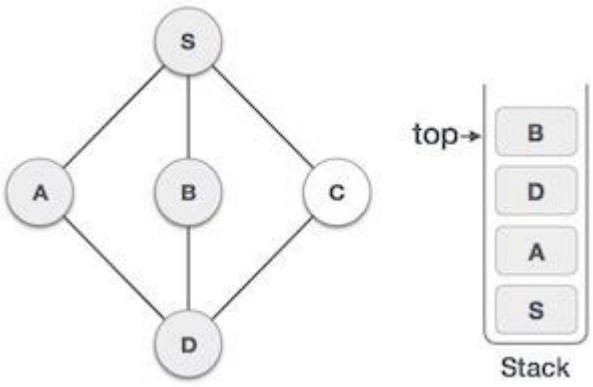
Algorithm

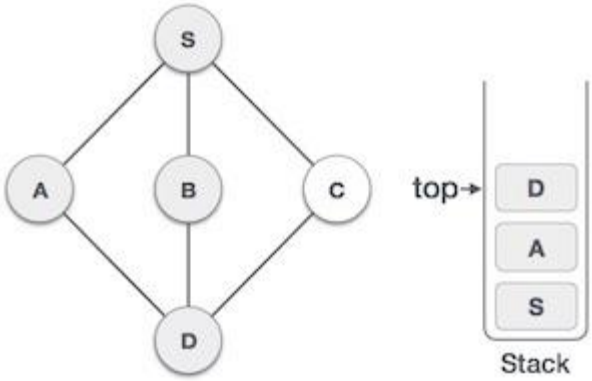
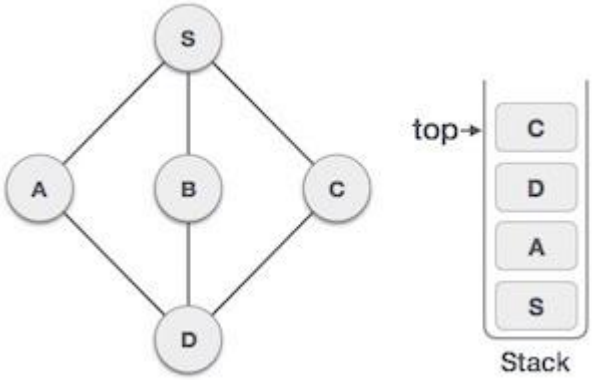
Step1: Start
Step2: Initialize all nodes as unvisited
Step3: Push the starting node onto the stack. Mark it as waiting.
Step4: Pop the top node from stack and mark is as visited. Push all its adjacent nodes into the stack &mark them as waiting.
Step 5 .Repeat step 4 until stack is empty.
Step 6: Stop

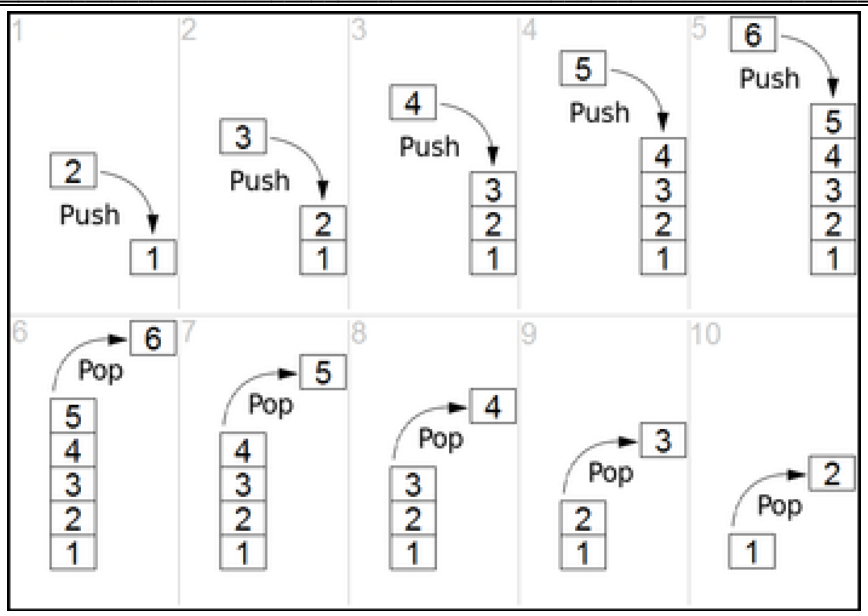
Step	Traversal	Description
1		Initialize the stack.
2		Mark **S** as visited and put it onto the stack. Explore any unvisited adjacent node from **S**. We have three nodes and we can pick any of them. For this example, we shall take the node in an alphabetical order.

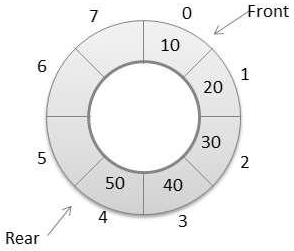
Algorithm: 2 marks

Example 2Marks

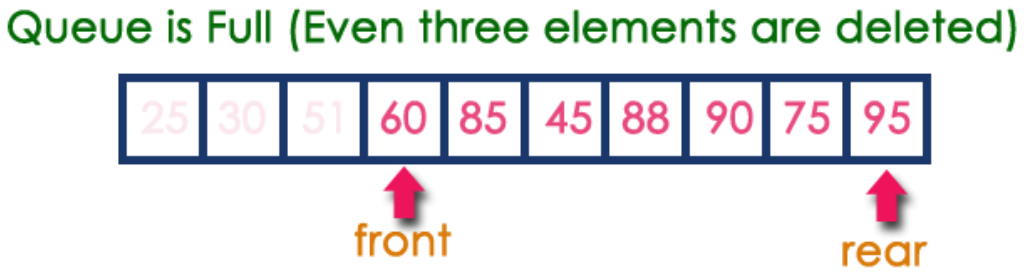
		3		Mark **A** as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both **S**and **D** are adjacent to **A** but we are concerned for unvisited nodes only.	
		4		Visit **D** and mark it as visited and put onto the stack. Here, we have **B** and **C** nodes, which are adjacent to **D** and both are unvisited. However, we shall again choose in an alphabetical order.	
		5		We choose **B**, mark it as visited and put onto the stack. Here **B** does not have any unvisited adjacent node. So, we pop **B** from the stack.	

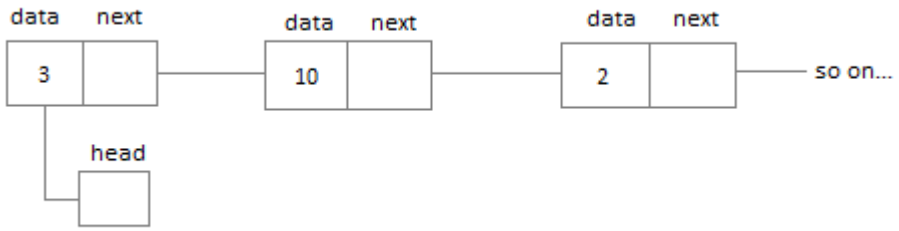
		6		We check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find **D** to be on the top of the stack.
		7		Only unvisited adjacent node is from **D** is **C** now. So we visit **C**, mark it as visited and put it onto the stack.
	c	**Describe with example, use of stack in reversing a list.**		4 M
	Ans	**Reversing a list** To reverse a list, the elements of list are pushed onto the stack one by one. Once all elements are pushed on the stack they are popped one by one. Since the element last pushed in comes out first, hence reversal of string occurs. Example: a list contains elements as {1, 2, 3, 4, 5, 6}. Every push operator will push an element on top of stack. Once all elements are pushed one can pop all elements and save it which results in to reversing of list as {6, 5, 4, 3, 2, 1}.		Explanation : 2 marks, Example: 2 marks

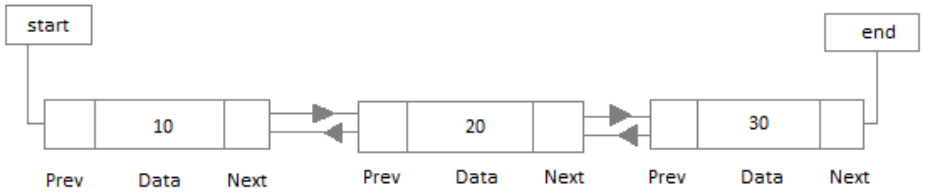


	d	**Explain the concept of circular queue with example.**	4 M
	Ans	**Definition:** A queue, in which the last node is connected back to the first node to form a cycle, is called as circular queue. • It has rear pointer to insert an element and front pointer to delete an element. • It works in FIFO manner where first inserted element is deleted first. **Representation:**  **Example:** In a normal Queue Data Structure, we can insert elements until queue becomes full. But once if queue becomes full, we cannot insert the next element until all the elements are deleted from the queue. For example consider the queue below... After inserting all the elements into the queue.	Definition: 1 mark, Representation: 1 mark, Example: 2 marks

Now consider the following situation after deleting three elements from the queue...

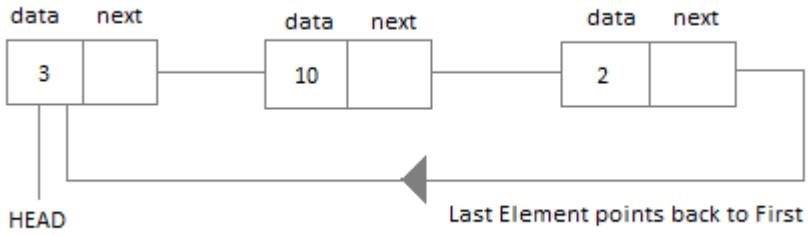


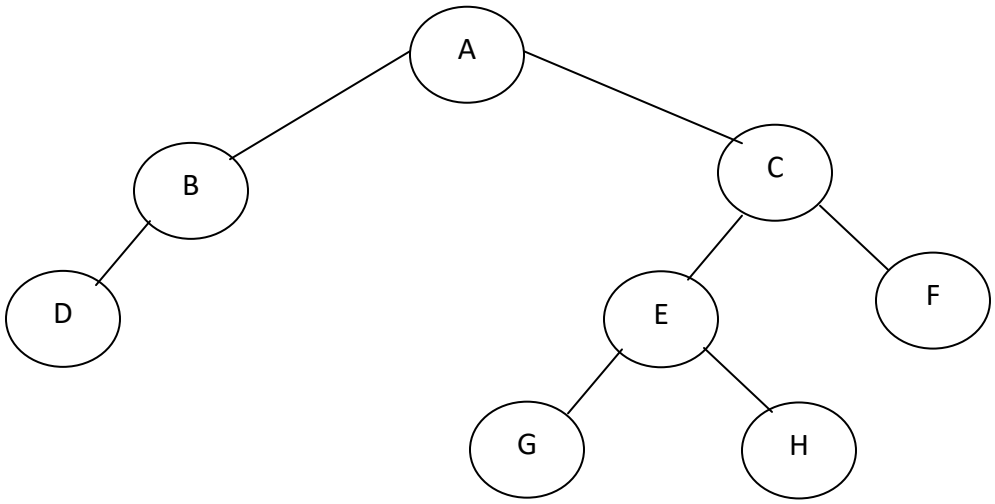
	e	**State and describe three types of linked list with suitable diagram.**	4 M
	Ans	**Types of Linked Lists** There are 3 different implementations of Linked List available, they are: 1. Singly Linked List 2. Doubly Linked List 3. Circular Linked List **Singly Linked List** Singly linked lists contain nodes which have a **data** part as well as an **address part** i.e. next, which points to the next node in the sequence of nodes. The operations we can perform on singly linked lists are **insertion**, **deletion** and **traversal**.  **Doubly Linked List** In a doubly linked list, each node contains a **data** part and two addresses, one for the **previous** node and one for the **next** node.	Types: 1 mark, Explanation of types: 1 mark each

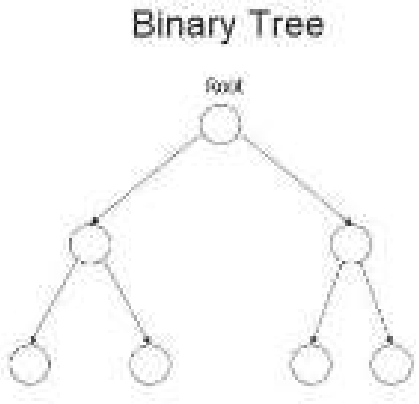
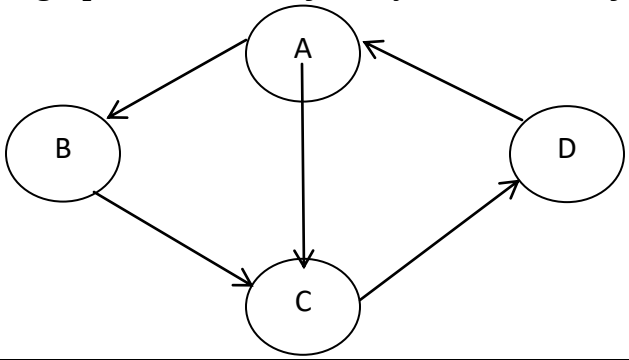


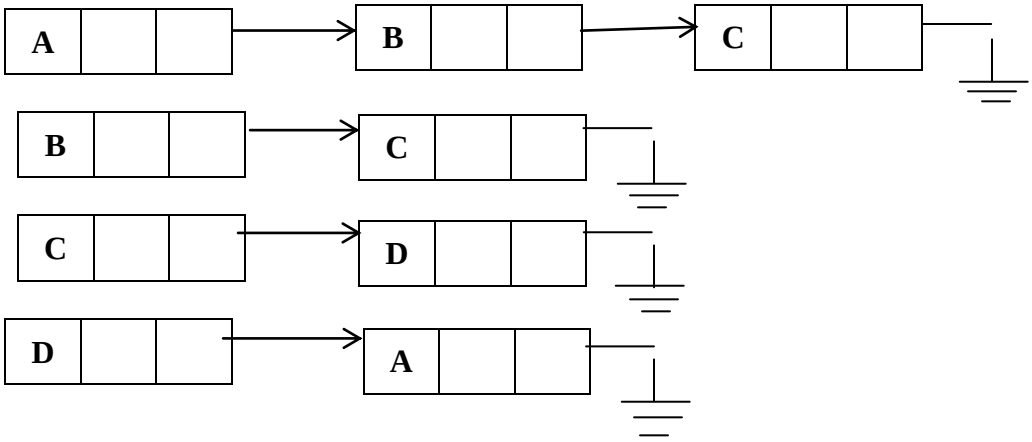
Circular Linked List

In circular linked list the last node of the list holds the address of the first node hence forming a circular chain.



	f	**Define the term binary tree. Write down preorder, inorder, postorder traversal for following tree** 	4 M
	Ans	**Binary tree:** A Binary tree is a nonlinear data structure in which each non-leaf node can have maximum two child nodes as left child ad right child.	Definition:1 mark, Each Traversal: 1 mark

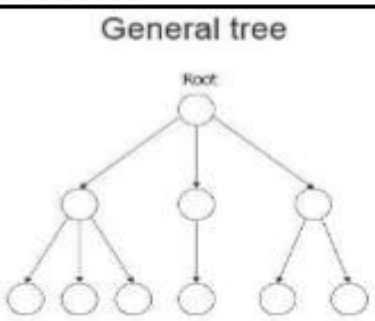
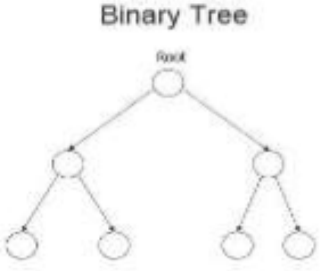
		 **Traversal:** **Inorder:** D – B – A – G – E – H – C – F **Preorder:** A – B – D – C – E – G – H – F **Postorder:** D – B – G – H – E – F – C – A	
4		**Attempt any FOUR :**	**16 M**
	a	**Consider the given graph. Write the adjacency matrix and adjacency list for it.** 	4 M

	Ans	**Adjacency matrix A for the above graph** **Adjacency list representation of above graph**  **Adjacency List**	Adjacency Matrix: 2 Marks, Adjacency list: 2 Marks
	b	**Differentiate between general tree and binary tree (any 4 points).**	4 M

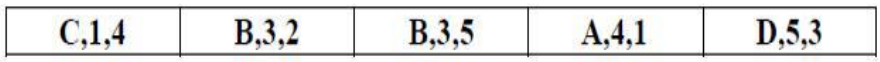
	A	**B**	**C**	**D**
A	0	1	1	0
B	0	0	1	0
C	0	0	0	1
D	1	0	0	0

Node	**Adjacency List**
A	B,C
B	C
C	D
D	A

	Ans		Any four correct points: 1 mark each

Sr.no	General Tree	Binary Tree
1	A general tree is a **data structure** in that each node can have infinite number of children	A Binary tree is a data structure in that each node has at most **two nodes** left and right
2	In general tree, root has **in-degree 0** and maximum **out-degree n**.	In binary tree, root has **in-degree 0** and maximum **out-degree 2**.
3	In general tree, each **node** have in-degree **one** and maximum out-degree **n**.	In binary tree, each node have in-degree **one** and maximum out-degree **2**.
4	**Height** of a general tree is the length of longest path from root to the leaf of tree. Height(T) = {**max**(height(child1) , height(child2) , … height(child-n)) +1}	Height of a binary tree is : Height(T) = { **max** (Height(Left Child) , Height(Right Child) + 1}
5	Subtree of general tree are **not ordered**	Subtree of binary tree is **ordered**.
		

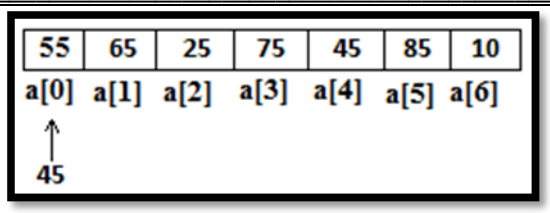
	c	**Explain the procedure for deleting first node from a singly linked list.**	4 M
	Ans	**Deleting first node from singly linked list** We can use the following steps to delete a first node from singly linked list... • **Step 1:** Check whether list is **Empty** (**head** == **NULL**) • **Step 2:** If it is **Empty** then, display **'List is Empty!!! Deletion is not possible'** and terminate the function. • **Step 3:** If it is **Not Empty** then, define a Node pointer **'temp'** and initialize with **head**. • **Step 4:** Check whether list is having only one node (**temp → next** == **NULL**) • **Step 5:** If it is **TRUE** then set **head** = **NULL** and delete **temp** (Setting **Empty** list conditions) **Step 6:** If it is **FALSE** then set **head** = **temp → next**, and delete **temp**.	4 marks for correct procedures
	d	**Describe priority queue with suitable example.**	4 M

	Ans	A priority Queue is a collection of elements where each element is assigned a priority and the order in which elements are added into the queue. The rules for processing the elements of priority queue are: 1) An element with higher priority is processed before any element of lower priority. 2) Two elements with the same priority are processed according to the order in which they are added to the queue (FCFS). One of the examples of priority queue is a queue used in operating system. The operating system has to handle a large number of jobs. These jobs have to be properly scheduled. The operating system assigns priorities to each type of job. The jobs are placed in a queue and the job with the highest priority will be executed first. Example: **Array representation:** Array element of priority queue has a structure with data, priority and order. Priority queue with 5 elements is as shown below:-  In the above diagram, each structure element has three members as information, priority and order in which element is arrived in the list.	Explanation :2M Example :2M
	e	**Perform bubble sort on following data to sort all elements in ascending order.** **15, 10, 02, 35, 08 (show all steps).**	4 M

	Ans	Pass 1 : 15 10 02 35 08 10 15 02 35 08 10 02 15 35 08 10 02 15 35 08 10 02 15 08 35 Pass 2 : 10 02 15 08 35. 02 10 15 08 35 02 10 15 08 35 02 10 08 15 35 Pass 3 : 02 10 08 15 35 02 10 08 15 35 02 08 10 15 35 Pass 4 : 02 08 10 15 35 The final sorted array in ascending order is 02, 08, 10, 15, 35.	Correct answer 4 M ** Give Stepwise Marks**
	f	**Write a C program to calculate the factorial of a number using recursion.**	4 M
	Ans	#include<stdio.h> #include<conio.h>	syntax: 2 marks, Logic: 2 marks

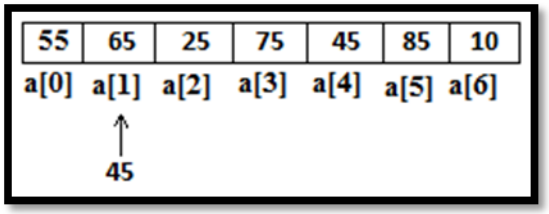
```
int Fibonacci(int);
int main()
{
int n, i = 0, c;
scanf("%d",&n);
printf("Fibonacci series\n");
for ( c = 1 ; c <= n ; c++ )
{
printf("%d\n", Fibonacci(i));
i++;
}
getch();
return 0;
}
int Fibonacci(int n)
{
if ( n == 0 )
return 0;
else if ( n == 1 )
return 1;
else
return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

5		**Attempt any FOUR :**	**16 M**
	a	**Consider the following array:** **55 65 25 75 45 85 10** **Write stepwise procedure to find 45 using linear search.**	4 M
	Ans	**Step 1:** Compare element 45 with a [0].	Correct answer 4M



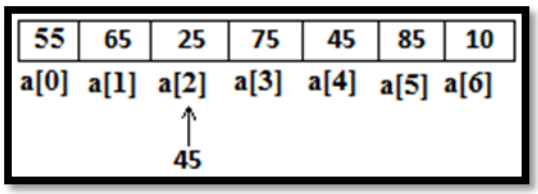
Since, 45 is not equal to a[0], compare next element.

Step 2: Compare element 45 with a[1].



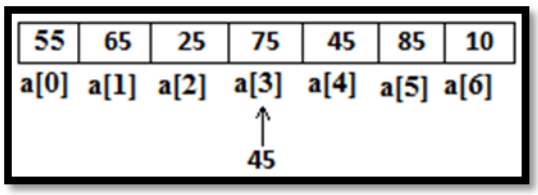
Since, 45 is not equal to a[1], compare next element.

Step 3: Compare element 45 with a[2].



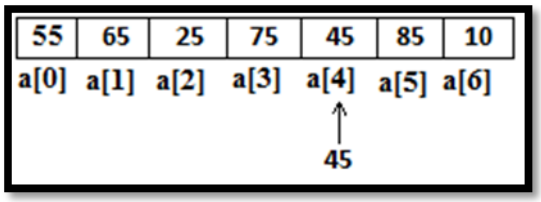
Since, 45 is not equal to a[2], compare next element.

Step 4: Compare element 45 with a[3].

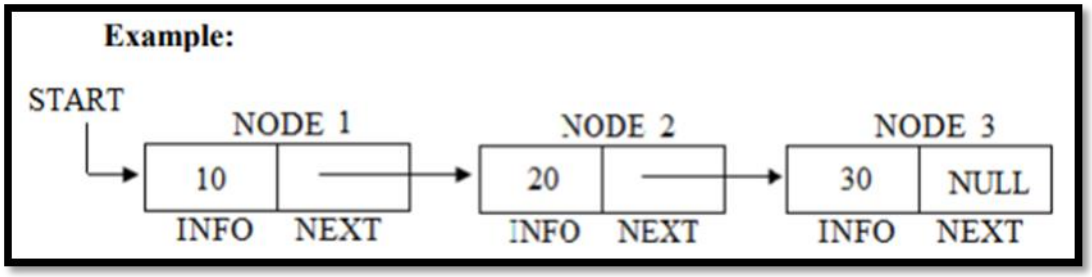


Since, 45 is not equal to a[3], compare next element.

Step 4: Compare element 45 with a[4].



Since, 45 is equal to a[4]. Therefore we conclude that key element 45 is found.

	b	**Define linked list. Write its two advantages and disadvantages.**	4 M
	Ans	**Linked list:** It is linear collection of data elements. Each element in linked list is called as 'node'. Each node contains two fields. First is INFO which stores data & second is NEXT which is linked with address of next node in a list.  **Advantages:** 1. Linked list is a dynamic memory allocation 2. In Linked list Insertions and deletions can be done easily just by storing an address of new node into previously existing node ie without movement of elements for insertion and deletion. **Disadvantages:** 1. It requires more space as pointers are also stored along with information. 2. Different amount of time is required to access each element. 3. If we have to go to a particular element then we have to go through all those elements that come before that element. 4. We cannot traverse it from last and only from the beginning. 5. It is not easy to sort the elements stored in the linear linked list.	Define 2M, Advantages 1M and Disadvantages 1M)
	c	**Write algorithm for inorder traversal of binary tree.**	4 M
	Ans	The algorithm works by: 1. Traversing the left sub-tree 2. Visiting the root node, and finally 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	4M for correct algorithm

Step 4: INORDER(ROOT-> RIGHT) [END OF LOOP]

Step 5: END

d **Describe Big 'O' notation. Also give example.** 4 M

Ans

1.Big O notation is used in Computer Science to describe the performance or complexity of an algorithm.

2.Big O specifically describes the worst-case scenario, and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.

3.Example:

Since Big O notation is a f(n) where n = no. of inputs

Consider f(n)= $10n^2$+ 4n+2 , g(n) = n^2 c=11 where c = constant ≥0 and n≥1

n	1	2	3	4	5
f(n)	16	50	104	178	272
g(n)	1	4	9	16	25
c. g(n)	11	44	99	176	275

From above table it is observed that f(n) > g(n) when n= 1,2,3,4

At n =5

F(n) ≤ c. g(n) for large value of n

Therefore O(n)=g(n)= n^2

2M for explanation and 2M for correct example

e **Define Hash function. Explain any one method of hashing.** 4 M

Ans

1. A **hash function** is any function that can be used to map data of arbitrary size to data of a fixed size. The values returned by a hash function are called **hash values**, **hash codes**, **digests**, or simply **hashes**.
2. Hash functions are often used in combination with a hash table, a common data structure used in computer software for rapid data lookup.
3. Hash functions accelerate table or database lookup by detecting duplicated records in a large file.
4. Hashing methods:-
    a. Division method
    b. Mid square method

2M definition, 2M for explanation of method.

c. Folding method

5. **Division method:** In this method hash address is calculated by dividing the key value by a prime number or a number without small divisor.

   Formula: H(K)=K(mod m) or H(K)=K(mod m)+1 K- Specify unique key value. m- Specify is a prime number or number without small divisors. Example:- H(3205)= 3205 mod 97=4 2.

6. **Middle square method:** In this method hash address is calculated by taking two digits from middle of square of key value.

7. Formula: H(K)=I I-specify digits after deleting digits from both ends of K2 Example:- H(3205)=(3205) 2 =10272015=72

8. **Folding method:** In this method hash address is calculated by partitioning key into multiple parts and performing addition. Formula: H(K)=K1+K2+...Kn Example: - H(3205)=32+05=37

Given a hash table of 100 locations, calculate the hash value using folding method for keys 5678, 321, and 34567

Solution

Since there are 100 memory locations to address, we will break the key into parts where each part (except the last) will contain two digits. The hash values can be obtained as shown below:

Key	**5678**	**321**	**34567**
Parts	56 and 78	32 and 1	34, 56 and 7
Sum	134	33	97
Hash Value	34 (Ignore the last carry)	33	97

f **Describe stack as ADT.** 4 M

Ans The stack , an abstract data type is defined as an ordered collection of items where items are added to and removed from the end called the “top.” 4M for correct explanation

Stacks are ordered LIFO.

The stack operations are given below.

Stack () creates a new stack that is empty. It needs no parameters and returns an empty stack.

push (item) adds a new item to the top of the stack. It needs the item and returns nothing.

Pop () removes the top item from the stack. It needs no parameters and returns the item. The stack is modified.

Peek () returns the top item from the stack but does not remove it. It needs no parameters. The stack is not modified.

isEmpty() tests to see whether the stack is empty. It needs no parameters and returns a boolean value.

size() returns the number of items on the stack. It needs no parameters and returns an integer.

6		**Attempt any FOUR :**	**16 M**
	a	**Describe working of radix sort with example.**	4 M
	Ans	1. In this method, ten buckets (0-9) are used to sort elements of an input list. All the elements are sorted according to their digit position from each element. 2. In pass 1, each element is placed inside the bucket with respect its unit position digit. After placing all elements inside the buckets, read those from 0th bucket to 9th bucket. 3. In pass 2, elements are placed in buckets with respect to 10th position digit from each element. In each pass one position is considered to arrange all the elements in bucket. 4. At the end of each pass elements are collected from buckets and given as input to the next pass. Total number of passes required for sorting is equal to maximum number of digits present in the largest number from the input list. 5. Last pass gives sorted list after reading all elements from 0th bucket to 9th bucket. 6. Example: 18,253, 1000,2 ,80,75,58 Pass1: In this pass arrange the element according to unit place.	2M for description and 2M for example

	0	1	2	3	4	5	6	7	8	9
18									18	
253				253						
1000	1000									
2			2							
80	80									
75						75				
58									58	

Output of 1st pass: 1000,80,2,253,75,18,58

Pass 2: In this pass arrange the element according to tens place.

	0	1	2	3	4	5	6	7	8	9
1000	1000									
80									80	
2	2									
253						253				
75								75		
18		18								
58						58				

Output of 2nd pass: 1000, 2, 18,253, 58, 75, 80

Pass 3: In this pass arrange the element according to hundred's place.

	0	1	2	3	4	5	6	7	8	9
1000	1000									
2	2									
18	18									
253			253							
58	58									
75	75									
80	80									

Output of 3rd pass: 1000, 2, 18, 58, 75, 80,253

Pass 4: In this pass arrange the element according to thousand's place

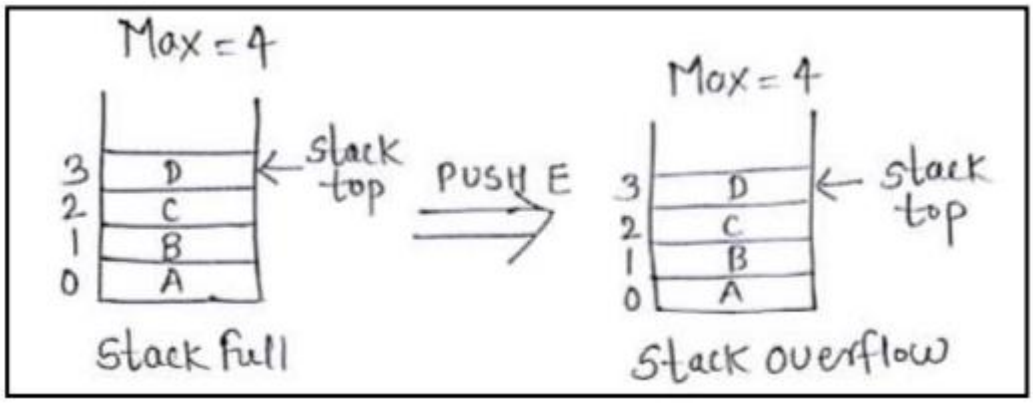
	0	1	2	3	4	5	6	7	8	9
1000		1000								
2	2									
18	18									
58	58									
75	75									
80	80									
253	253									

Output of 4th pass: 2,18,58,75,80,253,1000

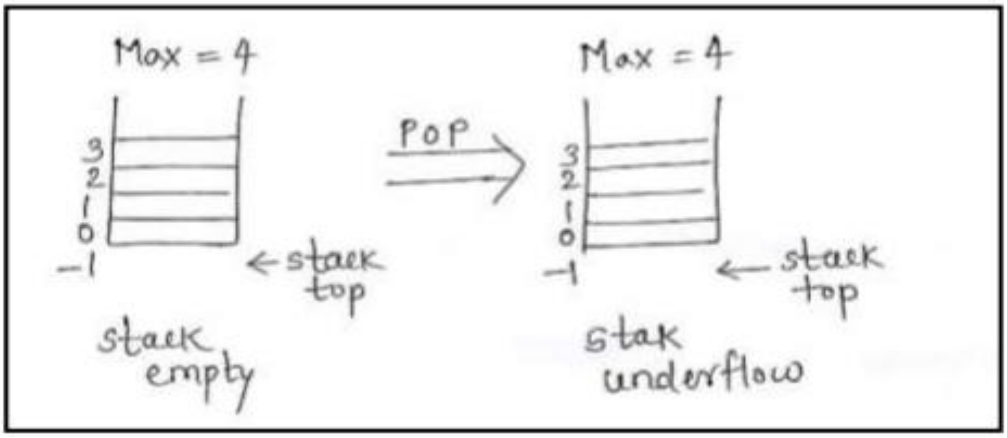
Elements in ascending orders: 2,18,58,75,80,253,1000

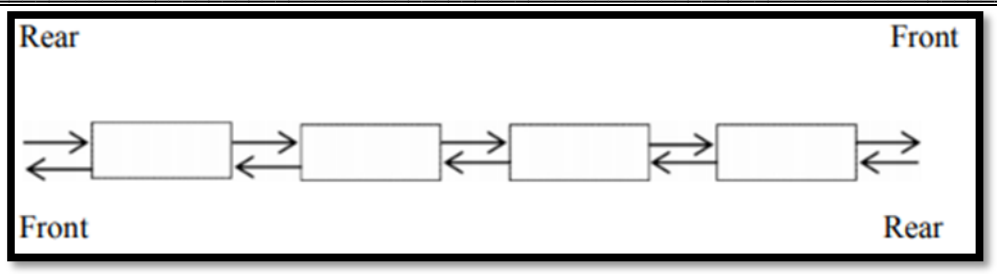
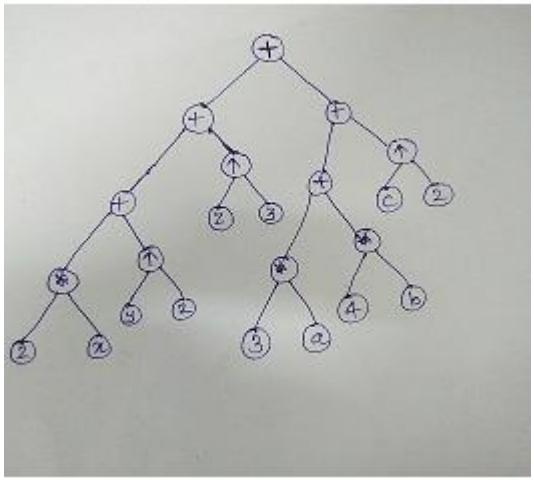
b **Explain underflow and overflow of stack with suitable diagram.** 4 M

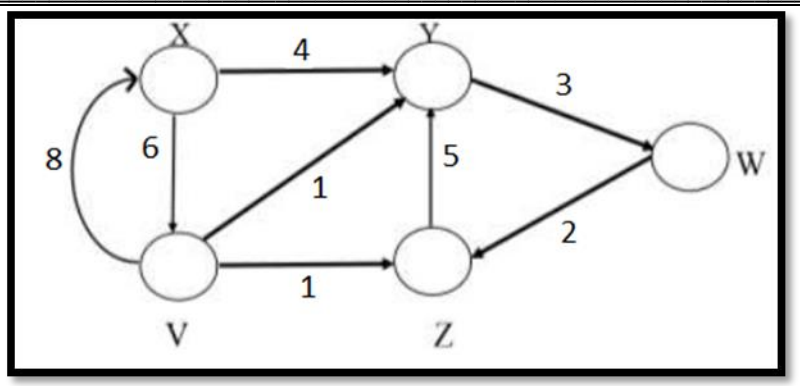
Ans **Stack overflow:** When stack contains maximum number of elements i.e. stack is full and push operation is called to insert a new element then stack is said to be in overflow state. 2M for stack overflow and 2M for stack underflow



Stack underflow: When there is no elements in a stack i.e. stack is empty and pop operation is called then stack is said to be in underflow state.

			
	c	**Write an algorithm for traversing in linked list.**	4 M
	Ans	Step 1: [INITIALIZE] SET PTR = START Step 2: Repeat Steps 3 and 4 while PTR != NULL Step 3: Apply Process to PTR DATA Step 4: SET PTR = PTR NEXT [END OF LOOP] Step 5: EXIT	4M for correct algorithm
	d	**Explain Double Ended Queue with suitable diagram.**	4 M
	Ans	1. A double-ended queue or dequeue is an abstract data structure that implements a queue for which elements can only be added to or removed from the front (head) or back (tail). 2. It is also often called a head-tail linked list. 3. Dequeue is a special type of data structure in which insertions and deletions will be done either at the front end or at the rear end of the queue. 4. The operations that can be performed on dequeues are: a. Insert an item from front end b. Insert an item from rear end c. Delete an item from front end d. Delete an item from rear end e. Display the contents of queue	2M for explanation and 2M for diagram

			
	e	**Draw tree structure for following expression: $(2x + y^2 + z^3) + (3a + 4b + c^2)$.**	4 M
	Ans		4M for correct structure
	f	**Define the following terms with respect to graph:** **a) Successor** **b) Indegree** **c) Path** **d) Weighted graph.**	4 M
	Ans	**a) Successor:** Successor is a node which comes after a particular node. In the below given diagram, successor of node X is V and Y. b) **Indegree:** It is number of edges coming towards a specified node i.e. number of edges that have that specified node as the head. The Indegree of node Y is 3. c) **Path:** Path is a sequence of alternating vertices and edges such that each successive vertex is connected by the edge. Frequently only the vertices are listed especially if there are no parallel edges. Suppose if we want to find the path to Z from X. The solution is X – Y – W – Z and another path is X – V – Z. d) **Weighted graph:** A graph whose edges are assigned non-negative numerical values is called as weighted graph. The diagram below given shows the eights for the edges	Definition each 1M


X
Y
W
V
Z
4
3
6
8
1
5
2
1