Zeal Education Society's

ZEAL POLYTECHNIC, PUNE.

NARHE | PUNE -41 | INDIA

SECOND YEAR (SY)

DIPLOMA IN COMPUTER ENGINEERING

SCHEME: I **SEMESTER: III**

NAME OF SUBJECT: DATA STRUCTURE USING C

Subject Code: 22317

MSBTE QUESTION PAPERS & MODEL ANSWERS

1. MSBTE WINTER-18 EXAMINATION
2. MSBTE SUMMER-19 EXAMINATION
3. MSBTE WINTER-19 EXAMINATION

22317

11819

3 Hours / 70 Marks

Seat No. □□□□□□□□

Instructions : (1) All Questions are *compulsory.*

(2) Illustrate your answers with neat sketches wherever necessary.

(3) Figures to the right indicate full marks.

(4) Assume suitable data, if necessary.

Marks

1. Attempt any FIVE of the following : 10

(a) Define the term algorithm.

(b) List any 4 applications of queue.

(c) Describe following terms w.r.to tree :

(i) leaf node

(ii) level of node

(d) Differentiate between stack and queue. (any two points)

(e) Describe undirected graph with suitable example.

(f) Define the terms : linear data structure and non-linear data structure.

(g) Convert infix expression into prefix expression : (A + B)*(C / G) + F

2. Attempt any THREE of the following : 12

(a) Describe working of linear search with example.

(b) Describe the concept of linked list with the terminologies : node, next pointer, null pointer and empty list.

(c) Describe queue full and queue empty operation conditions on linear queue with suitable diagrams.

(d) Differentiate between general tree and binary tree. (any four points)

3. **Attempt any THREE of the following :** **12**

(a) Write ‘c’ program for deletion of an element from an array.

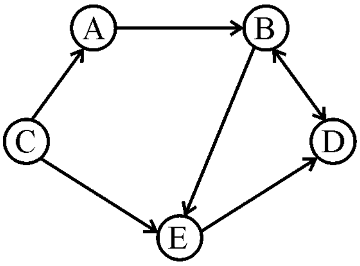
(b) Convert following expression into postfix form. Give stepwise procedure.

A + B ↑ C * (D / E) – F / G

(c) Find the position of element 29 using binary search method in an array ‘A’ given below. Show each step.

A = {11, 5, 21, 3, 29, 17, 2, 43}

(d) Give adjacency list and adjacency matrix for given graph :



4. **Attempt any THREE of the following :** **12**

(a) Describe working of bubble sort with example.

(b) Construct a binary search tree for following elements :

30, 100, 90, 15, 2, 25, 36, 72, 78, 10 show each step of construction of BST.

(c) Write an algorithm to count number of nodes in singly linked list.

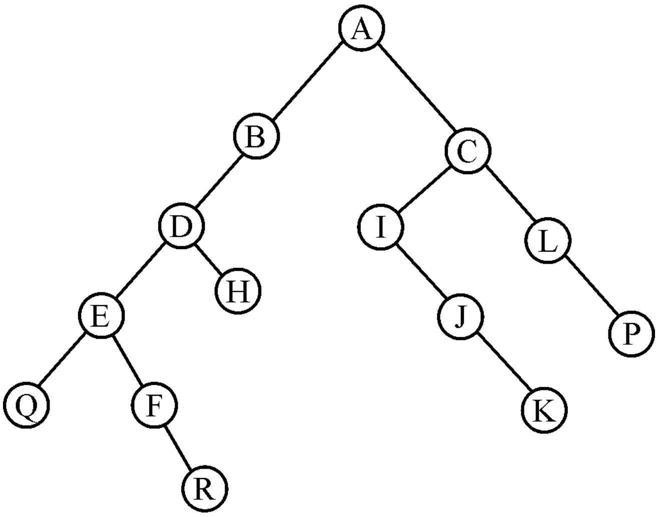
(d) Write a program in ‘C’ to insert an element in a linear queue.

(e) Describe circular linked list with suitable diagram. Also state advantage of circular linked list over linear linked list.

5. Attempt any TWO of following : 12

(a) Write algorithm for performing push and pop operations on stack.

(b) For given binary tree write in-order, pre-order and post-order traversal.



(c) Write an algorithm to insert an element at the beginning and at end of linked list.

6. Attempt any TWO of the following : 12

(a) Describe working of selection sort method. Also sort given input list in ascending order using selection sort input list – 55, 25, 5, 15, 35.

(b) Define the term recursion. Write a program in C to display factorial of a entered number using recursion.

(c) Describe procedure to delete an element from singly linked list using diagram.

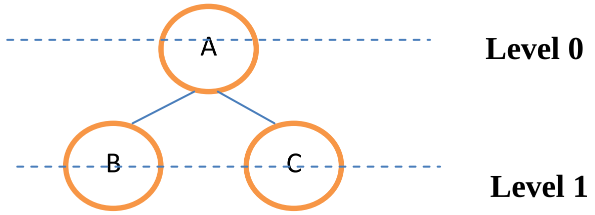
WINTER– 18 EXAMINATION

Subject Name: Data Structure using C **Model Answer** Subject Code: 22317

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		**Attempt any FIVE of the following :**	**10 M**
	a	**Define the term algorithm.**	2 M
	Ans	Algorithm is a stepwise set of instructions written to perform a specific task.	Correct definition 2M
	b	**List any 4 applications of queue.**	2 M
	Ans	• In computer system to maintain waiting list for single shared resources such as printer, disk, etc. • It is used as buffers on MP3 players, iPod playlist, etc. • Used for CPU scheduling in multiprogramming and time sharing systems. • In real life, Call Center phone systems will use Queues, to hold people calling them in an order, until a service representative is free. • Handling of interrupts in real-time systems. • Simulation	Any four apllications-1/2 M each
	c	**Describe following terms w.r.to tree:** **(i) Leaf node** **(ii) Level of node**	2 M
	Ans	**Example:**	Description of each term 1M



(i) Leaf node: A node without any child node is called as leaf node.

Nodes B and C are leaf node as shown in above example.

(ii) Level of node: Position of a node in the hierarchy of a tree is called as level of node.

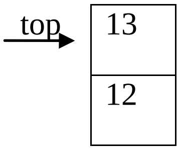
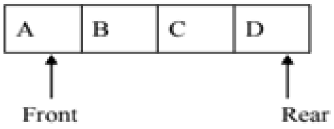
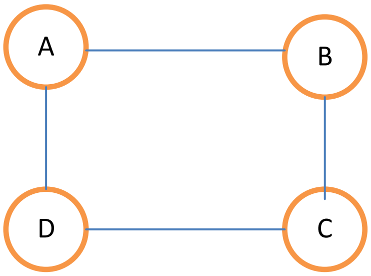
Level of node B is 1 as shown in above example.

d **Differentiate between stack and queue.(Any two points)** 2 M

Ans

Stack	**Queue**
1. Stack is a data structure in which insertion and deletion operations are performed at **same end**.	1. Queue is a data structure in which insertion and deletion operations are performed at **different ends**.
2. In stack an element inserted last is deleted first so it is called **Last In First Out list.**	2. In Queue an element inserted first is deleted first so it is called **First In First Out list.**
3.In stack only one pointer is used called as **stack top**	3.In Queue two pointers are used called as **front** and **rear**
4. **Example**: Stack of books	4. **Example**: Students standing in a line at fees counter
5.**Application**: • Recursion • Polish notation	5. **Application**: • In computer system for organizing processes. • In mobile device for sending receiving messages.

Any two correct differences-1M each

<table>
<tr><td></td><td></td><td><table><tr><td><b>6. Representation</b>: Using array

 </td><td><b>6. Representation:</b> Using array

 </td></tr></table></td><td></td></tr>
<tr><td></td><td><b>e</b></td><td><b>Describe undirected graph with suitable example.</b></td><td>2 M</td></tr>
<tr><td></td><td><b>Ans</b></td><td>Undirected graph: A graph in which the edges do not have any direction associated with them is known as undirected graph.
In undirected graph, if an edge exists between two nodes A and B then the nodes can traverse from A to B as well as from B to A. Each edge is bidirectional.
Example:-


In the above example, each edge is bidirectional.</td><td>Definition-1M, example-1M</td></tr>
<tr><td></td><td><b>f</b></td><td><b>Define the terms: Linear data structure and non-linear data structure.</b></td><td>2 M</td></tr>
<tr><td></td><td><b>Ans</b></td><td>Linear Data Structure: A data structure in which all data elements are stored in a particular sequence is known as linear data structure.
Example: stack, queue

Non-Linear data structure: A data structure in which all data elements are not stored in any particular sequence is known as nonlinear data structure.
Example: graph and tree.</td><td>Each term definition 1M</td></tr>
<tr><td></td><td><b>g</b></td><td><b>convert infix expression into prefix expression:</b>
<b>(A+B)*(C/G)+F</b></td><td>2 M</td></tr>
<tr><td></td><td><b>Ans</b></td><td><table><tr><th>Infix expression</th><th>Read Character</th><th>Stack contents</th><th>Prefix expression</th></tr><tr><td><b>(A+B)*(C/G)+F</b></td><td><b>F</b></td><td><b>-</b></td><td><b>F</b></td></tr><tr><td><b>(A+B)*(C/G)+</b></td><td>+</td><td>+</td><td><b>F</b></td></tr></table></td><td>Correct prefix expression - 2M</td></tr>
</table>

(A+B)*(C/G)	**)**	**+)**	**F**
(A+B)*(C/G	**G**	**+)**	**GF**
(A+B)*(C/	**/**	**+)/**	**GF**
(A+B)*(C	**C**	**+)/**	**CGF**
(A+B)*(	**(**	**+**	**/CGF**
(A+B)*	*****	**+***	**/CGF**
(A+B)	**)**	**+*)**	**/CGF**
(A+B	**B**	**+*)**	**B/CGF**
(A+	**+**	**+*)+**	**B/CGF**
(A	**A**	**+*)+**	**AB/CGF**
(	**(**	**+***	**+AB/CGF**
			***+AB/CGF**
			+*+AB/CGF

2		**Attempt any THREE of the following :**	**12 M**
	a	**Describe working of linear search with example.**	4 M

Ans

In linear search, search element is compared with each element from the list in a sequence. Comparison starts with first element from the list and continues till number is found or comparison reaches to the last element of the list.

As each element is checked with search element, the process of searching requires more time. Time complexity of linear search is O (n) where n indicates number of elements in list.

Linear search on sorted array:-On sorted array search takes place till element is found or comparison reaches to an element greater than search element.

Example:- Using array representation

Input list 10, 20, 30, 40, 50 and Search element 30, Index =0

Iteration 1

10	20	30	40	50

↑

10 ! = 30

Relevant description 2M, Any correct example-2M

Index = Index + 1

Iteration 2

10	20	30	40	50

↑

20 ! = 30

Index = Index + 1

Iteration 3

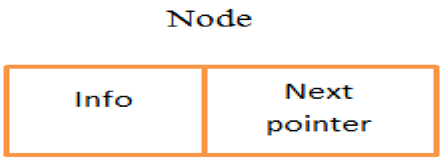
10	20	30	40	50

↑

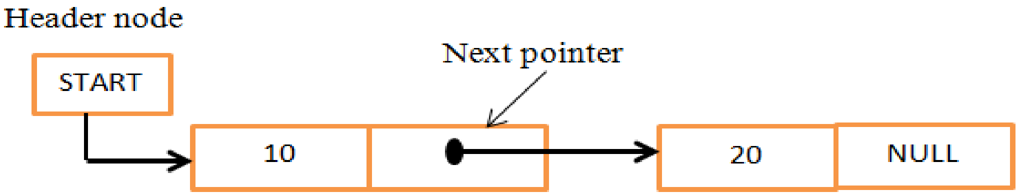
30 = 30

Number found

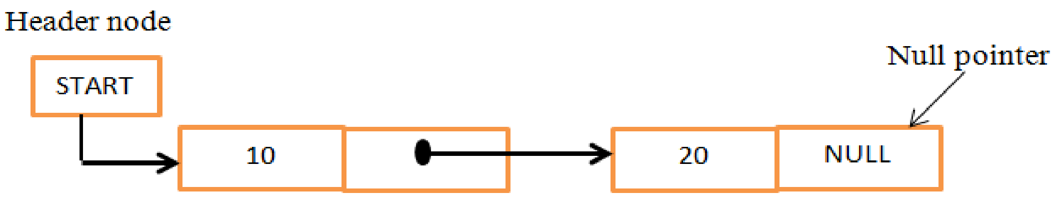
	b	**Describe the concept of linked list with the terminologies: node, next Pointer, null pointer and empty list.**	4 M
	Ans	**Node**: Each data element in a linked list is represented as a node. Node contains two parts-one is info (data) and other is next pointer (address). Info part stores data and next pointer stores address of next node.	Description of each terminology -1M



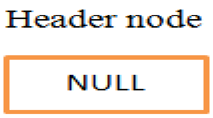
Next pointer: It is a pointer that holds address of next node in the list i.e. next pointer points to next node in the list



Null pointer: It is a pointer that does not hold any memory address i.e. it is pointing to nothing. It is used to specify end of the list. The last element of list contains NULL pointer to specify end of list.



Empty list: Each linked list has a header node. When header node contains NULL value, then that list is said to be empty list.



c **Describe queue full and queue empty operation conditions on linear queue with suitable diagrams.**

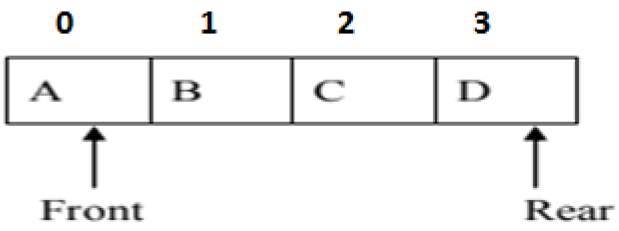
4 M

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. If queue is full then new element cannot be added to a queue.

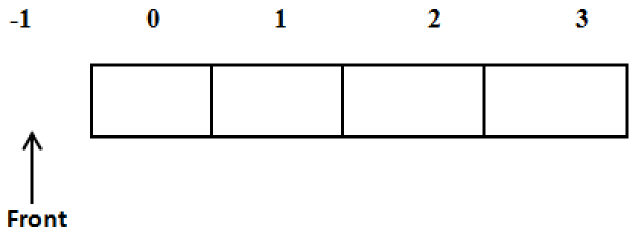
Example:-

Consider max=4. First element is stored at 0th position and last element is stored at 3rd position in queue. In the diagram given below rear pointer is pointing to max-1 (3) position so queue is full.



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.

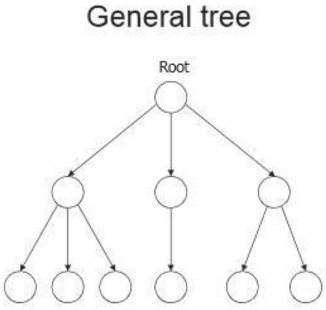
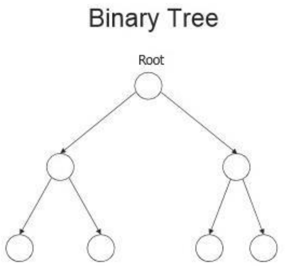
Example:-In the diagram given below front pointer points to -1 value i.e. it points no location inside queue so queue is empty.



Description of queue full-1M, diagram-1M, Description of queue empty-1M, diagram-1M

d **Differentiate between general tree and binary tree. (any four points)** 4 M

Ans

Sr. no	**General Tree**	**Binary Tree**
1	A general tree is a **data structure** in which each node can have infinite number of children	A Binary tree is a data structure in which each node has at most **two nodes** i.e. 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**.
6		

Any four relevant differences -1M each

3 **Attempt any THREE of the following :** **12 M**

a **Write a C program for deletion of an element from an array.** 4 M

Ans

```
#include <stdio.h>
int main()
{
  int array[100], position, c, n;
  printf("Enter number of elements in array\n");
  scanf("%d", &n);
  printf("Enter %d elements\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);
  printf("Enter the location where you wish to delete element\n");
  scanf("%d", &position);
  if (position >= n+1)
```

4M for correct logic & program code

```
            printf("Deletion not possible.\n");
          else
          {
            for (c = position - 1; c < n - 1; c++)
               array[c] = array[c+1];

            printf("Resultant array:\n");

            for (c = 0; c < n - 1; c++)
               printf("%d\n", array[c]);
          }
           return 0;
        }
```

b **Convert following expression into postfix form. Give stepwise procedure.**

A+B↑C*(D/E)-F/G.

4 M

Ans Consider input expression as (A+B↑C*(D/E)-F/G)

Scanned Symbol	Operation stack	Postfix Expression
(	(	
A	(	A
+	(+	A
B	(+	AB
↑	(+↑	AB
C	(+↑	ABC
*	(+*	ABC↑
(	(+*(	ABC↑
D	(+*(	ABC↑D
/	(+*(/	ABC↑D
E	(+*(/	ABC↑DE
)	(+*	ABC↑DE/
-	(-	ABC↑DE/*+
F	(-	ABC↑DE/*+F

Correct Postfix Expression 4M

/	(-/	ABC↑DE/*+F
G	(-/	ABC↑DE/*+FG
)	EMPTY	ABC↑DE/*+FG/-

POSTFIX EXPRESSION: ABC↑DE/*+FG/-

c **Find the position of element 29 using binary search method in an array ‘A’ given below. Show each step.**

A={11,5,21,3,29,17,2,43}

4 M

Ans An array which is given A[ ]= {11,5,21,3,29,17,2,43} is not in sorted manner, first we need to sort them in order;

1M for taking sorted input & 1M each for every iteration

So an array will be A[ ]={2,3,5,11,17,21,29,43} and the value to be searched is VAL = 29.

The binary search algorithm will proceed in the following manner.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
2	3	5	11	17	21	29	43

Iteration 1:

BEG = 0, END = 7, MID = (0 + 7)/2 = 3

Now, VAL = 29 and A[MID] = A[3] =11

A[3] is less than VAL, therefore, we now search for the value in the second half of the array.

So, we change the values of BEG and MID.

Iteration 2:

Now, BEG = MID + 1 = 4, END = 7, MID = (4 + 7)/2 =11/2 = 5; VAL = 29 and A [MID] = A [5] = 21

A[5] is less than VAL, therefore, we now search for the value in the second half of the segment.

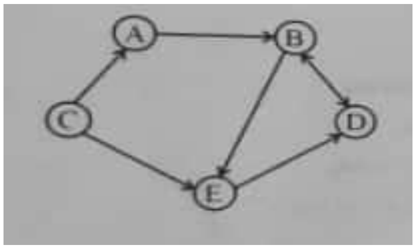
So, again we change the values of BEG and MID.

Iteration 3:

Now, BEG = MID + 1 = 6, END = 7, MID = (6 + 7)/2 = 6 Now, VAL = 29 and A [MID] = A [6]=29

So, Element 29 is found at 6th location in given array A[]={2,3,5,11,17,21,29,43}.

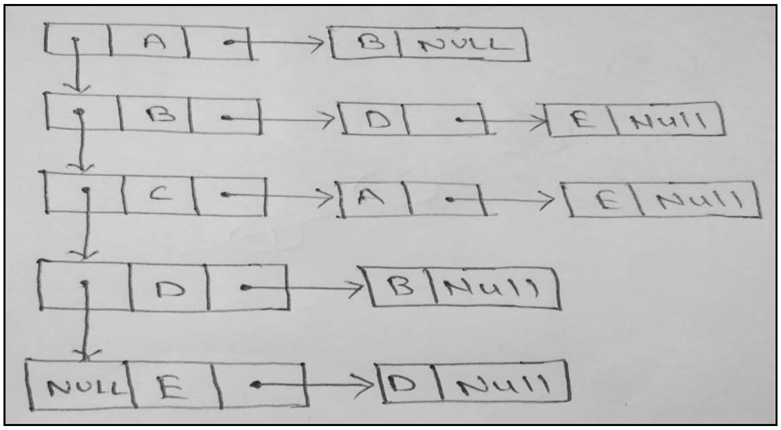
d give adjacency list and adjacency matrix for given graph: 4 M



Ans Adjacency List: (Using Linked List)

2M for Correct List and 2M for Correct matrix

Here, we use doubly linked list for storing header node list and singly linked list for storing respective adjacent node to it.



OR

Adjacency List

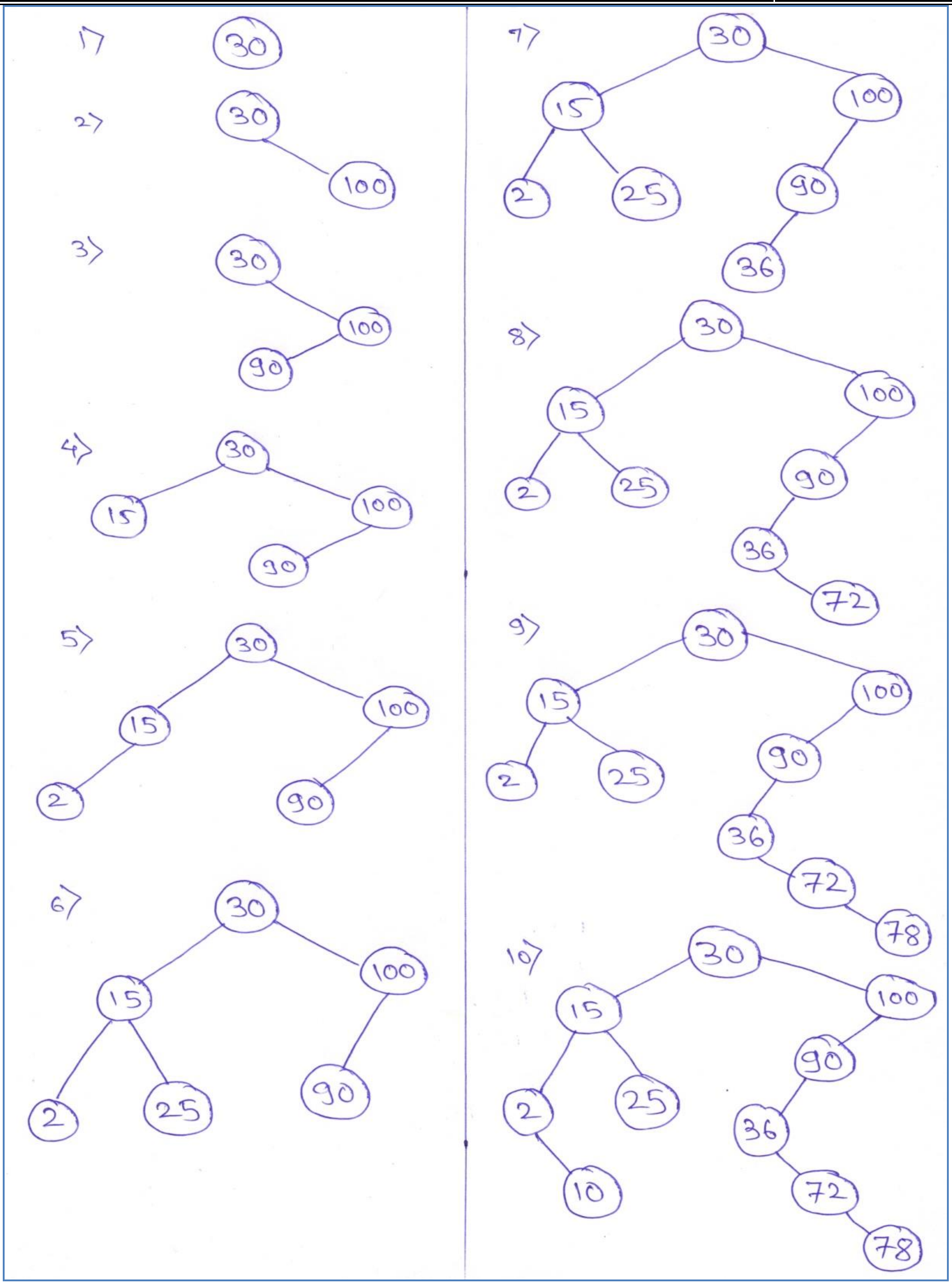
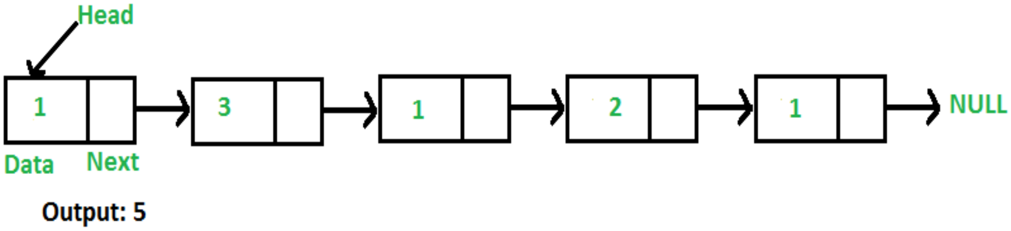
Nodes	**Adjacent Nodes**
A	**B**
B	**D,E**
C	**A,E**
D	**B**
E	**D**

Adjacency Matrix: (Using Array)

$$\begin{array}{c|ccccc|}
A & 0 & 1 & 0 & 0 & 0 \\
B & 0 & 0 & 0 & 1 & 1 \\
C & 1 & 0 & 0 & 0 & 1 \\
D & 0 & 1 & 0 & 0 & 0 \\
E & 0 & 0 & 0 & 1 & 0
\end{array}$$

4		**Attempt any THREE of the following :**	**12 M**
	a	**Describe working of bubble sort with example.**	4 M
	Ans	Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity is of O (n^2) where **n** is the number of items. **Bubble Sort Working:** We take an unsorted array for our example as A[]={19, 2, 27, 3, 7, 5, 31}. Bubble sort takes O(n^2) time so we're keeping it short and precise. {{**Note: Pass 4 onwards optional**}} Pass 1: 2,19,27,3,7,5,31 2,19,27,3,7,5,31 2,19,3,27,7,5,31 2,19,3,7,27,5,31 2,19,3,7,5,27,31 Pass 1 Completed Pass 2: 2,19,3,7,5,27,31 2,3,19,7,5,27,31 2,3,7,19,5,27,31	2M for description & 2M for example

		2,3,7,5,19,27,31 2,3,7,5,19,27,31 Pass 2 Completed Pass 3: 2,3,7,5,19,27,31 2,3,7,5,19,27,31 2,3,5,7,19,27,31 Pass 3 Completed Pass 4: 2,3,5,7,19,27,31 Pass 4 Completed Pass 5: 2,3,5,7,19,27,31 Pass 5 Completed Pass 6: 2,3,5,7,19,27,31 Pass 6 Completed	
	b	**Construct a binary search tree for following elements:** **30,100,90,15,2,25,36,72,78,10 show each step of construction of BST.**	4 M
	Ans	Stepwise construction of Binary search tree for following elements: 30,100,90,15,2,25,36,72,78,10 is as follows:	4M for all correct steps

			
	c	**Write an algorithm to count number of nodes in singly linked list.**	4 M
	Ans	Function to count number of nodes in a given singly linked list.  Head 1 3 1 2 1 NULL Data Next Output: 5	4M for correct algorithm

For example, the function should return 5 for linked list 1->3->1->2->1.

Algorithm: Using Iterative Solution

```
1) Initialize count as 0
2) Initialize a node pointer, current = head.
3) Do following while current is not NULL
     a) current = current -> next
     b) count++;
4) Return count
```

d **Write a program in 'C' to insert an element in a linear queue.** 4 M

Ans

```
// C program to insert an element in a linear queue using array
#include<stdio.h>
#include<conio.h>
#define n 5
void main()
{
   int queue[n],ch=1,front=0,rear=0,i,j=1,x=n;
   //clrscr();
   printf("Queue using Array");
   printf("\n1.Insertion \n2.Display \n3.Exit");
   while(ch)
   {
      printf("\nEnter the Choice:");
      scanf("%d",&ch);
      switch(ch)
      {
      case 1:
         if(rear==x)
            printf("\n Queue is Full");
         else
         {
            printf("\n Enter no %d:",j++);
            scanf("%d",&queue[rear++]);
         }
         break;
      case 2:
         printf("\n Queue Elements are:\n ");
         if(front==rear)
            printf("\n Queue is Empty");
```

4M for correct logic & program code

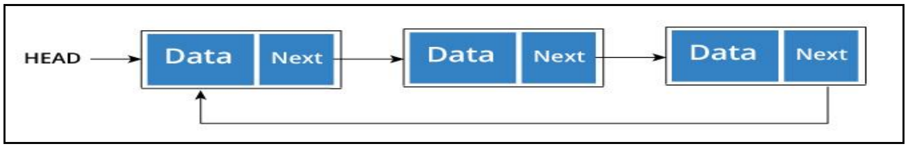
```
            else
            {
               for(i=front; i<rear; i++)
               {
                  printf("%d",queue[i]);
                  printf("\n");
               }
               break;
            case 3:
               exit(0);
            default:
               printf("Wrong Choice: please see the options");
            }
         }
      }
      getch();
   }
```

e **Describe circular linked list with suitable diagram. Also state advantage of circular linked list over linear linked list.**

4 M

Ans **Circular Linked List**

A circular linked list is a variation of linked list in which the last element is linked to the first element. This forms a circular loop.



A circular linked list can be either singly linked or doubly linked.

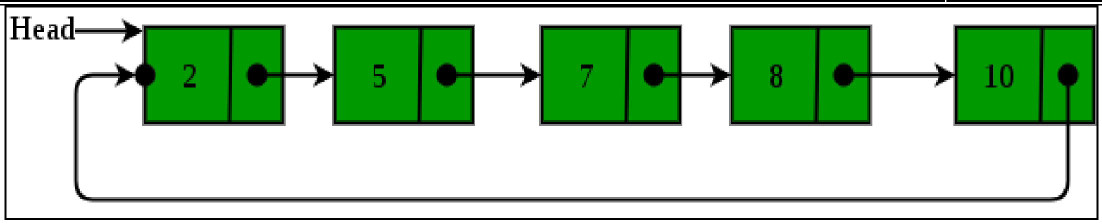
- for singly linked list, next pointer of last item points to the first item
- In doubly linked list, prev pointer of first item points to last item as well.

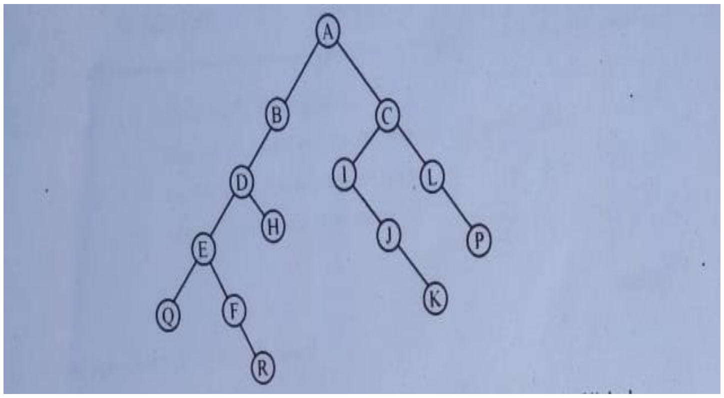
We declare the structure for the circular linked list in the same way as follows:

```
Struct node
{
Int data;
Struct node *next;
};
Typedef struct node *Node;
Node *start = null;
Node *last = null;
```

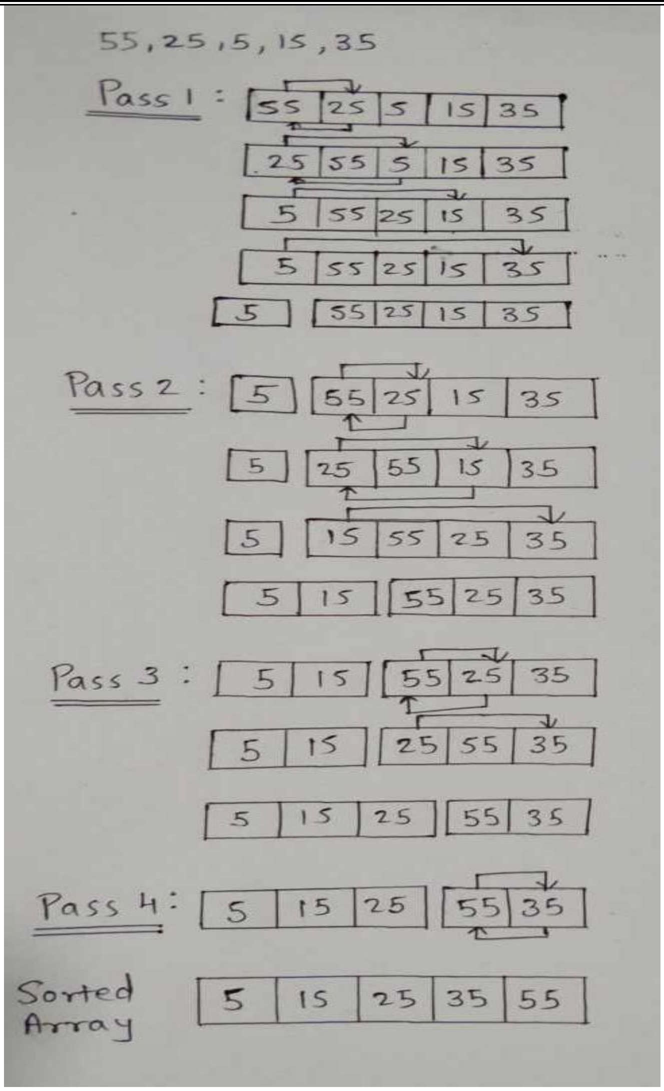
For example:

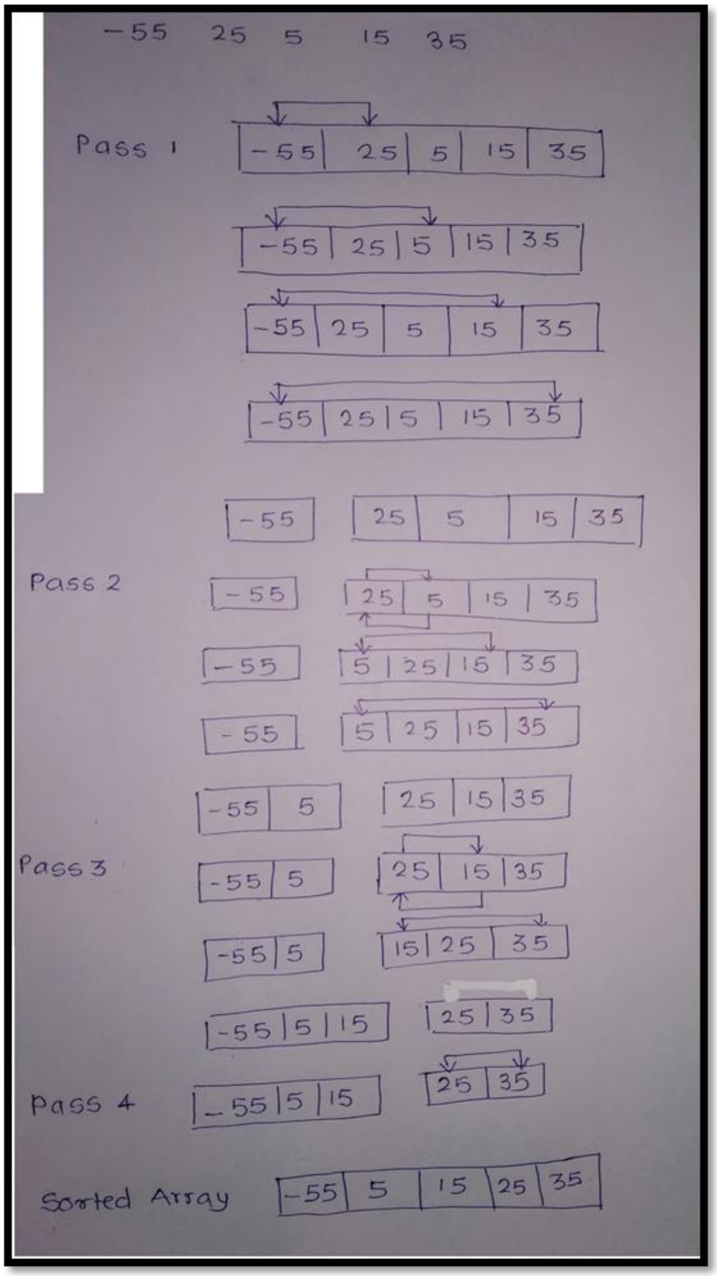
2M for description 1M for diagram and 1M for any one advantage

		 **Advantages of Circular Linked Lists:** **1)** Any node can be a starting point. We can traverse the whole list by starting from any point. We just need to stop when the first visited node is visited again. **2)** Useful for implementation of queue. Unlike this implementation, we don't need to maintain two pointers for front and rear if we use circular linked list. We can maintain a pointer to the last inserted node and front can always be obtained as next of last. **3)** Circular lists are useful in applications to repeatedly go around the list. For example, when multiple applications are running on a PC, it is common for the operating system to put the running applications on a list and then to cycle through them, giving each of them a slice of time to execute, and then making them wait while the CPU is given to another application. It is convenient for the operating system to use a circular list so that when it reaches the end of the list it can cycle around to the front of the list. **4)** Circular Doubly Linked Lists are used for implementation of advanced data structures like Fibonacci Heap.	
5		**Attempt any TWO of the following :**	**12 M**
	a	**Write algorithm for performing push and pop operations on stack.**	6 M
	Ans	**Push algorithm: -** Max is maximum size of stack. Step 1: [Check for stack full/ overflow] If stack_top is equal to max-1 then Display output as "Stack Overflow" and return to calling function Otherwise Go to step 2 Step 2: [Increment stack_top] Increment stack top pointer by one. stack_top=stack_top +1; Step 3: [Insert element] stack [stack_top] = item; Step 4: return to calling function **Pop algorithm: -** Max is maximum size of stack. Step 1: [Check for stack empty/underflow]	3marks for Push algorithm and 3marks for Pop operation

		If stack_top is equal to -1 then Display output as “Stack Underflow” and return to calling function Otherwise Go to step 2 Step 2: [delete element] stack [stack_top] = item; Step 3: [Decrement stack_top] Decrement stack top pointer by one. stack_top=stack_top -1; Step 4: return to calling function.	
	b	**For given binary tree write in-order, pre-order and post-order traversal.** 	6 M
	Ans	**Inorder Traversal: Q,E,F,R,D,H,B,A,I,J,K,C,L,P** **Preorder Traversal: A,B,D,E,Q,F,R,H,C,I,J,K,L,P** **Postorder Traversal: Q,R,F,E,H,D,B,K,J,I,P,L,C,A**	2marks for each traversal
	c	**Write an algorithm to insert an element at the beginning and end of linked list.**	6 M
	Ans	**Algorithm to insert an element at the beginning of linked list:** **1.** Start **2.** Create the node pointer *temp Struct node * temp **3.** Allocate address to temp using malloc temp = malloc(sizeof(struct node)); **4.** Check whether temp is null, if null then Display “Overflow” else	3marks for each algorithm

		temp-> info=data temp-> next=start **5.** Start=temp **6.** stop **Algorithm to insert an element at the end of linked list:** **1.** Start **2.** Create two node pointers *temp, *q struct node * temp, *q; **3.** q= start **4.** Allocate address to temp using malloc temp = malloc(sizeof(struct node)); **5.** Check whether temp is null, if null then Display “Overflow” else temp-> info=data temp-> next=null **6.** While(q->next!=null) q= q-> next **7.** q->next= temp **8.** stop	
6		**Attempt any TWO of the following :**	**12 M**
	a	**Describe working of selection sort method. Also sort given input list in ascending order using selection sort input list:- 55, 25, 5, 15, 35.**	6 M
	Ans	**Working of Selection sort**: Selection Sort algorithm is used to arrange a list of elements in a particular order (Ascending or Descending). In selection sort, the first element in the list is selected and it is compared repeatedly with remaining all the elements in the list. If any element is smaller than the selected element (for ascending order), then both are swapped. Then we select the element at second position in the list and it is compared with remaining all elements in the list. If any element is smaller than the selected element, then both are swapped. This procedure is repeated till the entire list is sorted.	3marks for description, 3marks for correct solution

		 55, 25, 5, 15, 35 Pass 1 : 55 | 25 | 5 | 15 | 35 25 | 55 | 5 | 15 | 35 5 | 55 | 25 | 15 | 35 5 | 55 | 25 | 15 | 35 5 55 | 25 | 15 | 35 Pass 2 : 5 55 | 25 | 15 | 35 5 25 | 55 | 15 | 35 5 15 | 55 | 25 | 35 5 | 15 55 | 25 | 35 Pass 3 : 5 | 15 55 | 25 | 35 5 | 15 25 | 55 | 35 5 | 15 | 25 55 | 35 Pass 4 : 5 | 15 | 25 55 | 35 Sorted Array 5 | 15 | 25 | 35 | 55 **OR**	

		 -55 25 5 15 35 Pass 1 -55 25 5 15 35 -55 25 5 15 35 -55 25 5 15 35 -55 25 5 15 35 -55 25 5 15 35 Pass 2 -55 25 5 15 35 -55 5 25 15 35 -55 5 25 15 35 -55 5 25 15 35 Pass 3 -55 5 25 15 35 -55 5 15 25 35 -55 5 15 25 35 Pass 4 -55 5 15 25 35 Sorted Array -55 5 15 25 35	
	b	**Define the term recursion. Write a program in C to display factorial of an entered number using recursion.**	6 M

	Ans	**Definition:** Recursion is the process of calling function by itself. A recursive function body contains function call statement that calls itself repeatedly.
		Program:
		#include<stdio.h>
		#include<conio.h>
		int fact(int n);
		void main()
			2marks for definition, 4marks for correct program

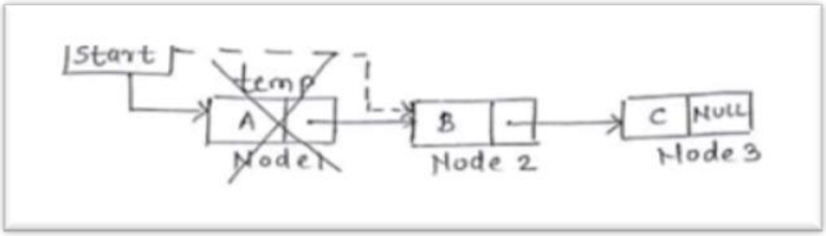
```
{
int n;
 clrscr();
 printf("\nThe factorial of % is = %d",n,fact(n));
getch();
}
 int fact(int n)
{
if(n==1)
return 1;
else
return(n*fact(n-1));
}
```

c **Describe procedure to delete an element from singly linked list using diagram.** 6 M

Ans

In a linear linked list, a node can be deleted from the beginning of list, from in between positions and from end of the list.

Delete a node from the beginning:-



Node to be deleted is node1.Create a temporary node as 'temp'. Set 'temp' node with the address of first node. Store address of node 2 in header pointer 'start' and then delete 'temp' pointer with free function. Deleting temp pointer deletes the first node from the list.

OR

Step 1: Create temporary node 'temp'.

Step 2: Assign address of first node to 'temp' pointer.

Step 3: Store address of second node (temp->next) in header pointer 'start'.

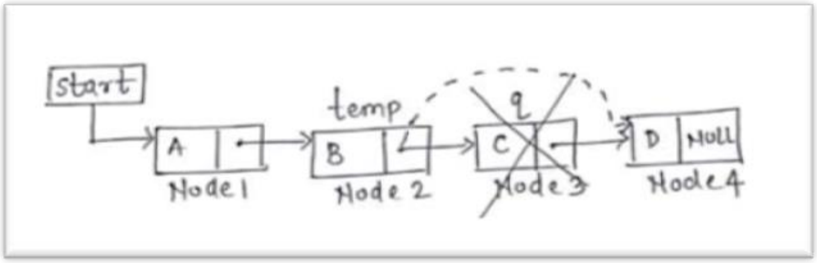
Step 4: Free temp.

Delete a node from in between position:-

**Note: Correct algorithm or program shall be considered.

Any two deletions shall be considered

3marks each



Node to be deleted is node3.Create a temporary node as ‘temp’ and ‘q’. Set ‘temp’ node with the address of first node. Traverse the list up to the previous node of node 3 and mark the next node (node3) as ‘q’. Store address from node ‘q’ into address field of ‘temp’ node. Then delete ‘q’ pointer with free function. Deleting ‘q’ pointer deletes the node 3 from the list.

OR

Step 1: Create temporary node ‘temp’, ’q’.

Step 2: Assign address of first node to ‘temp’ pointer.

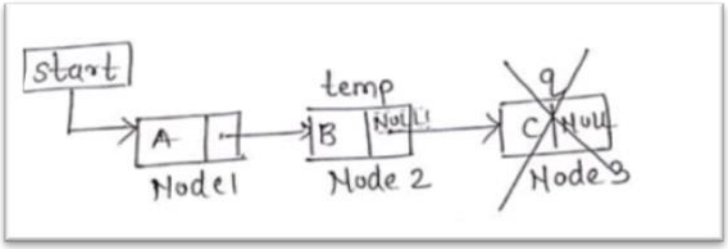
Step 3: Traverse list up to previous node of node to be deleted.

Step 4: Mark the node to be deleted ‘q’.

Step 5: Store address from node ‘q’ in address field of ‘temp’ node (temp- >next=q->next).

Step 6: Free q.

Delete a node from the end:-



Node to be deleted is node 3.Create a temporary node as ‘temp’ and ‘q’. Set ‘temp’ node with the address of first node. Traverse the list up to the second last node and mark the last node as ‘q’. Store NULL value in address field of ‘temp’ node and then delete ‘q’ pointer with free function. Deleting q pointer deletes the last node from the list.

OR

Step 1: Create temporary node ‘temp’,’q’.
Step 2: Assign address of first node to ‘temp’ pointer.
Step 3: Traverse list upto second last node.
Step 4: Mark last node’s address in node ‘q’.
Step 5: store NULL value in address field of second last node (temp->next).
Step 6: Free q

22317

21819

3 Hours / 70 Marks

Seat No. □□□□□□□□

Instructions : (1) All Questions are *compulsory.*

(2) Illustrate your answers with neat sketches wherever necessary.

(3) Figures to the right indicate full marks.

(4) Assume suitable data, if necessary.

(5) Mobile Phone, Pager and any other Electronic Communication devices are not permissible in Examination Hall.

Marks

1. Attempt any FIVE of the following : 10

(a) List any four operations on data structure.

(b) Enlist queue operations condition.

(c) Define :

(i) Binary tree (ii) Binary search tree

(d) Show the memory representation of stack using array with the help of a diagram.

(e) Define given two types of graph and give example.

(i) Direct graph (ii) Undirected graph

(f) Differentiate between linear and non-linear data structures on any two parameters.

(g) Convert the following infix expression to its prefix form using stack

A + B – C ∗ D / E + F

2. Attempt any THREE of the following : 12

(a) Explain the working of Binary search with an example.

(b) Write a program to traverse a linked list.

(c) Draw and explain construction of circular queue.

(d) Explain indegree and outdegree of a graph with example.

3. Attempt any THREE of the following : 12

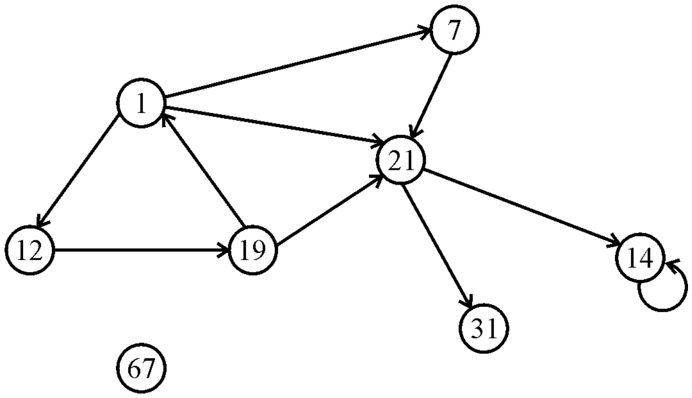
(a) Write C program for performing following operations on array : insertion, display.

(b) Evaluate the following postfix expression :

5, 6, 2, +, ∗, 12, 4, /, – Show diagrammatically each step of evolution using stack.

(c) Sort the following numbers in ascending order using quick sort. Given numbers 50, 2, 6, 22, 3, 39, 49, 25, 18, 5.

(d) From the following graph, complete the answers :



(i) Indegree of node 21

(ii) Adjacent node of 19

(iii) Path of 31

(iv) Successor of node 67

4. Attempt any THREE of the following : 12

(a) Differentiate between binary search and sequential search (linear search).

(b) Draw the tree structure of the following expressions :

(i) $(2a + 5b)^3 * (x - 7y)^4$ (ii) $(a - 3b) * (2x - y)^3$

(c) Create a singly linked list using data fields 15, 20, 22, 58, 60. Search a node 22 from the SLL and show procedure step-by-step with the help of diagram from start to end.

(d) Evaluate the following prefix expression :

– ∗ + 4 3 2 5 show diagrammatically each step of evaluation using stack.

(e) Write an algorithm to delete a node from the beginning of a circular linked list.

5. Attempt any TWO of the following : 12

(a) Show the effect of PUSH and POP operation on to the stack of size 10. The stack contains 40, 30, 52, 86, 39, 45, 50 with 50 being at top of the stack. Show diagrammatically the effect of :

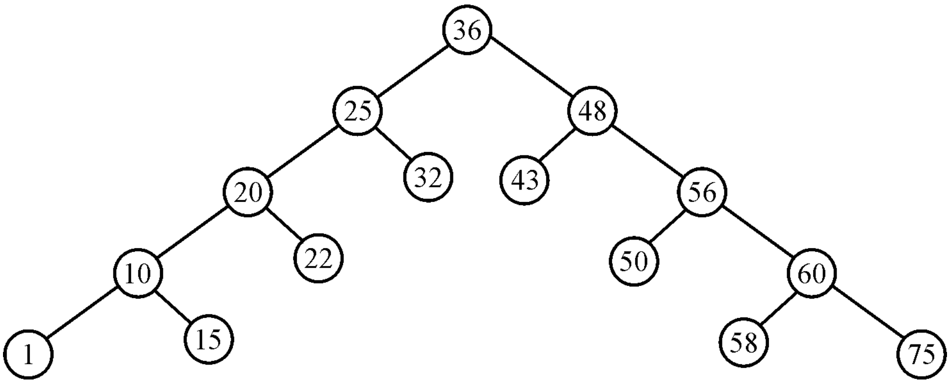
(i) PUSH 59 (ii) PUSH 85

(iii) POP (iv) POP

(v) PUSH 59 (vi) POP

Sketch the final structure of stack after performing the above said operations.

(b) Traverse the following tree by the in-order, pre-order and post-order methods :



(c) Write an algorithm to count number of nodes in singly linked list.

6. Attempt any TWO of the following : 12

(a) Sort the following numbers in ascending order using Bubble sort. Given numbers : 29, 35, 3, 8, 11, 15, 56, 12, 1, 4, 85, 5 & write the output after each interaction.

(b) Evaluate the following postfix expression :

57 + 62 – *

(c) Create a singly linked list using data fields 90, 25, 46, 39, 56. Search a node 40 from the SLL and show procedure step-by-step with the help of diagram from start to end.

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)

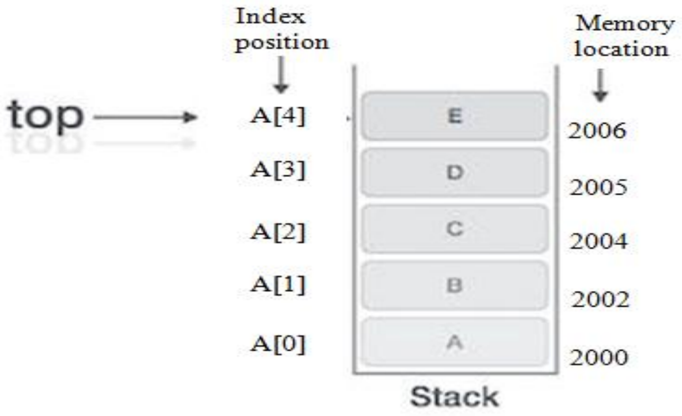
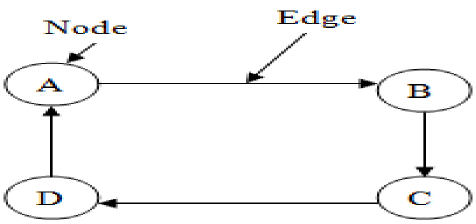
SUMMER – 2019 EXAMINATION
MODEL ANSWER

Subject: Data Structure Using 'C' **Subject Code: 22317**

Important Instructions to examiners:

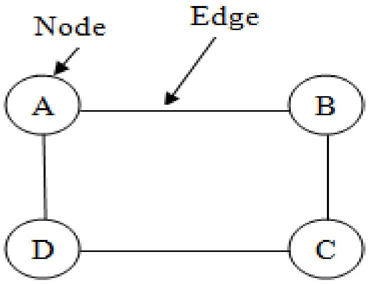
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling 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)** **Ans.**	**Attempt any FIVE of the following:** **List any four operations on data structure.** **Operations on data structure:** • Insertion • Deletion • Searching • Sorting • Traversing • Merging	**10** **2M** ***Any four operations $^{1/2}$M each***
	(b) **Ans.**	**Enlist queue operation condition.** 1. Queue Full 2. Queue Empty	**2M** ***Two operational conditions 1M each***

	(c) **Ans.**	**Define:** **(i) Binary tree (ii) Binary search tree** **(i) Binary tree**: It is a nonlinear data structure in which each non-leaf node can have maximum two child nodes as left child ad right child. **(ii)Binary search tree**: It is a nonlinear data structure in which left child of root node is less than root and right child of root node is greater than root.	**2M** ***Each correct definition 1M***
	(d) **Ans.**	**Show the memory representation of stack using array with the help of a diagram.** Consider stack contains five integer elements represented with an array A in which each element occupy 2 bytes memory. Array starts with base address of 2000.  Index position Memory location top A[4] A[3] A[2] A[1] A[0] E D C B A 2006 2005 2004 2002 2000 Stack	**2M** ***Correct representation 2M***
	(e) **Ans.**	**Define given two types of graph and give example.** **(i) Direct graph (ii) Undirected graph** **(i) Direct graph:** A graph in which direction is associated with each edge is known as directed graph. Example:  	**2M** ***Definition with example of each1M***

(ii) Undirected graph: A graph in which the edges do not have any direction associated with them is known as undirected graph.

Example:-



(f) **Differentiate between linear and non-linear data structures on any two parameters.** **2M**

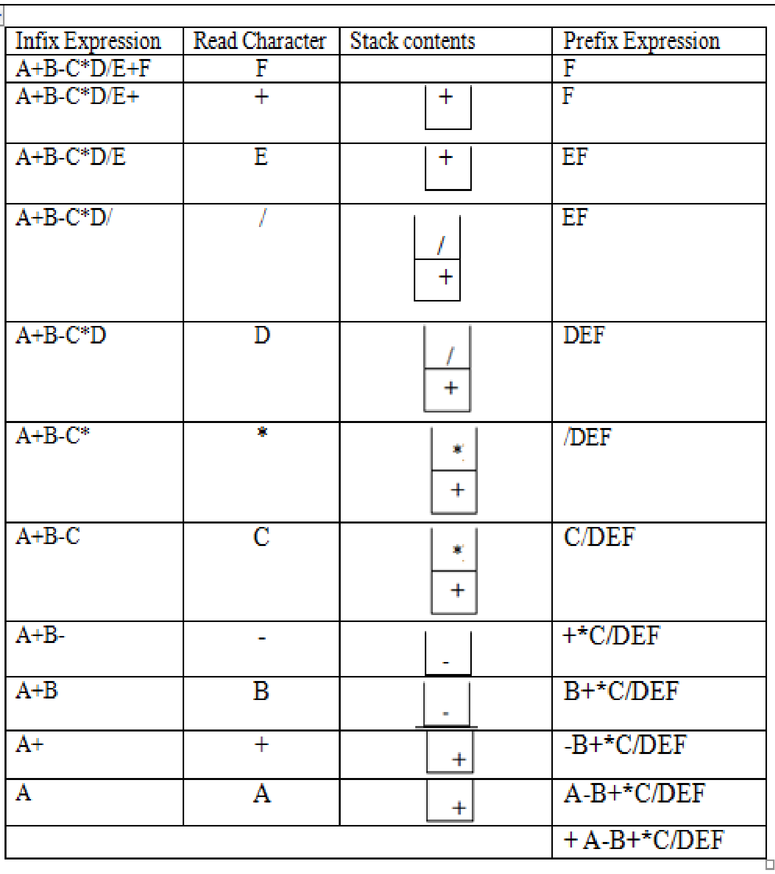
Ans.

Sr. No.	**Linear data structure**	**Non-linear data structure**
1	A data structure in which all data elements are stored in a sequence is known as linear data structure.	A data structure in which all data elements are not stored in a sequence is known as non-linear data structure.
2	All elements are stored in contiguous memory locations inside memory.	All elements may stored in non-contiguous memory locations inside memory.
3	Example:- stack, queue	Example:- tree, graph

Any two differences 1M each

(g) **Convert the following infix expression to its prefix form using stack A + B – C * D/E + F** **2M**

Ans.



Infix Expression	Read Character	Stack contents	Prefix Expression
A+B-C*D/E+F	F		F
A+B-C*D/E+	+	+	F
A+B-C*D/E	E	+	EF
A+B-C*D/	/	/ +	EF
A+B-C*D	D	/ +	DEF
A+B-C*	*	* +	/DEF
A+B-C	C	* +	C/DEF
A+B-	-	-	+*C/DEF
A+B	B	-	B+*C/DEF
A+	+	+	-B+*C/DEF
A	A	+	A-B+*C/DEF
			+A-B+*C/DEF

Correct prefix expression2M

2. **Attempt any THREE of the following:** **12**

(a) **Explain the working of Binary search with an example.** **4M**

Ans. Binary search is performed only on sorted array. Search method starts with calculating mid position from an array and compare the mid position element with the search element. If a match is found thenthe search process ends otherwise divide the i/p list into 2 parts. First part contains all thenumbers less than mid position element and second part contains all the numbers greater than mid position element.Then select one of the part depending on search element is less or greater than mid position element and calculate mid position for selected part.Again compare mid position element with search element. The binary search performs comparison and division task the element is found or division of list gives one element for comparison.

To calculate mid element perform (lower + upper) / 2.

lower-lower index position of an array(initially 0)

upper-upper index position of an array(initially size-1)

Explanation 2M

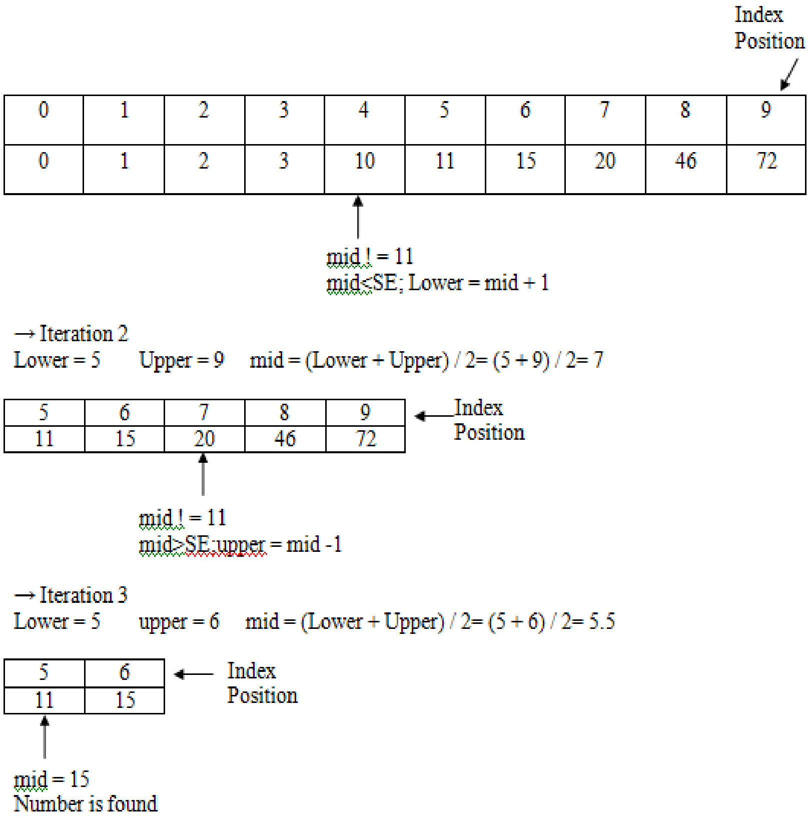
Example:

Consider Input list 0, 1, 2, 9, 10, 11, 15, 20, 46, 72

Search element:11

→ Iteration 1

Lower = 0 Upper = 9mid = (lower + upper) / 2= (0 + 9/2)= 4.5


Index Position

0	1	2	3	4	5	6	7	8	9
0	1	2	3	10	11	15	20	46	72

mid ! = 11
mid<SE; Lower = mid + 1

→ Iteration 2
Lower = 5 Upper = 9 mid = (Lower + Upper) / 2= (5 + 9) / 2= 7

5	6	7	8	9
11	15	20	46	72

Index Position

mid ! = 11
mid>SE;upper = mid -1

→ Iteration 3
Lower = 5 upper = 6 mid = (Lower + Upper) / 2= (5 + 6) / 2= 5.5

5	6
11	15

Index Position

mid = 15
Number is found

Example 2M

(b) **Write a program to traverse a linked list.** **4M**

(Note: create_list and addatbeg are optional)

Ans.

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>

void create_list(int);
void addatbeg(int);
void display();
struct node
```

Correct logic 2M

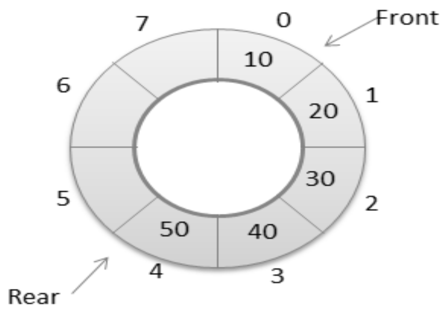
Correct syntax 2M

```
{
int info;
struct node *next;
}*start=NULL;

void main()
{
int m;
clrscr();
printf("enter data value\n");
scanf("%d",&m);
create_list(m);
printf("enter data value\n");
scanf("%d",&m);
addatbeg(m);
  display();
getch();
}

void create_list(int data)
{
struct node *tmp,*q;
tmp=malloc(sizeof(struct node));
tmp->info=data;
tmp->next=NULL;
 start=tmp;
}

void addatbeg(int data)
{
struct node *tmp;
tmp=malloc(sizeof(struct node));
tmp->info=data;
tmp->next=start;
 start=tmp;
}

void display()
{
```

```
struct node *q;
 if(start==NULL)
 {
printf("list is empty\n");
  }
 q=start;
printf("list is:\n");
 while(q!=NULL)
 {
printf("%d\t",q->info);
  q=q->next;
 }
}
```

(c) Draw and explain construction of circular queue. **4M**

Ans. A queue, in which the last node is connected back to the first node to form a cycle, is called as circular queue.



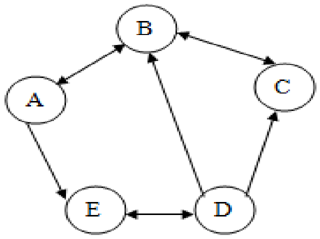
Draw 1M

The above diagram represents a circular queue using array.

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.

Initially front and rear both are initialized to -1 to represent queue empty. First element inserted in circular queue is stored at 0th index position pointed by rear pointer. For the very first element, front pointer is also set to 0th position. Whenever a new element is inserted in a queue rear pointer is incremented by one. If rear is pointing to max-1 and no element is present at 0th position then rear is set to 0th position to continue cycle. Before inserting an element, queue full condition is checked. If rear is set to max-1 position and front is set to 0 then queue is full. Otherwise if rear =front+1 then also queue is full.

Explanation 3M

		If queue is full then new element cannot be added in a queue. For deletion, front pointer position is checked and queue empty condition is checked. If front pointer is pointing to -1 then queue is empty and deletion operation cannot be performed. If queue contains any element then front pointer is incremented by one to remove an element. If front pointer is pointing to max-1 and element is present at 0th position then front pointer is initialize to 0th position to continue cycle. Circular queue has advantage of utilization of space**.** Circular queue is full only when there is no empty position in a queue. Before inserting an element in circular queue front and rear both the pointers are checked. So if it indicates any empty space anywhere in a queue then insertion takes place.	
	(d) **Ans.**	**Explain indegree and outdegree of a graph with example.** **Indegree of node:** It is number of edges coming towards a specified node i.e. number of edges that have that specified node as the head is known as indegree of a node. **Outdegree of node:** It is number of edged going out from a specified node i.e. number of edges that have that specified node as the tail is known as outdegree of a node In undirected graph each edge is bidirectional so each edge coming towards node is also going out of that node. Due to this indegree and outdegree of a node is same number. In indirected graph, each edge is having direction associated with it, so indegree and outdegree depends on the direction. ***Example:-***  Indegree of node A= 1 Outdegree of node A=2	**4M** ***Each term-explanation 1M*** ***Each example 1M***

		Indegree of node B= 3 Outdegree of node B=2 Indegree of node C= 2 Outdegree of node C=1 Indegree of node D= 1 Outdegree of node D=3 Indegree of node E= 2 Outdegree of node E=1	

3.	**(a)**
	Ans.
		Attempt any THREE of the following:
		Write C program for performing following operations on array: insertion, display.
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		inta[10],x,i,n,pos;
		clrscr();
		printf(“Enter the number of array element\n”);
		scanf(“%d”,&n);
		printf(“Enter the array with %d element\n”, n);
		for(i=0;i<n;i++)
		scanf(“%d”,&a[i]);
		printf(“Enter the key value and its position\n”);
		scanf(“%d%d” ,&x,&pos);
		for(i=n; i >= pos; i--)
		{
		a[i]=a[i-1];
		}
		a[pos-1]=x;
		printf(“Array element\n “);
		for(i=0;i<n+1;i++)
		printf(“%d\t”,a[i]);
		getch();
		}
			12
			4M
			Correct program 4M

	(b) **Ans.**	**Evaluate the following postfix expression:** **5, 6, 2, +, *, 12, 4, /, - Show diagrammatically each step of evolution using stack.**	**4M**

Scanned Symbol	Operand 1	Operand 2	Value	Stack Content
5				5
6				5,6
2				5,6,2
+	6	2	8	5,8
*	5	8	40	40
12				40,12
4				40,12,4
/	12	4	3	40,3
-	40	3	37	37

Correct answer 4M

Result of above postfix expression evaluation- 37

(c) **Sort the following numbers in ascending order using quick sort. Given numbers 50, 2, 6, 22, 3, 39, 49, 25, 18, 5.** **4M**

Ans. Given array

Array elements	50	2	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

Correct solve example 4M

Set l=0 , h=9 ,pivot= a[h]=5
Initialize index of smaller element, i= l-1 =-1
Traverse elements from j=l to j=h-1

1. j=0 i=-1 since a[j] > pivot do nothing array will remain same

Array elements	50	2	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

2. j=1 since a[j]<=pivot, do i++ and swap(a[i], a[j])
   i=0

Array elements	**2**	**50**	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

3. j=2 ,i=0 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**50**	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

4. j=3 ,i=0 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**50**	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

5. j=4, since a[j]<=pivot do, i++ and swap(a[i],a[j])
i=1

Array elements	**2**	**3**	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

6. j=5 , i=1 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**3**	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

7. j=6, i=1 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**3**	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

8. j=7 ,i-1 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**3**	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

9. j=8 ,i-1 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**3**	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

We come out of loop because j is now equal to high-1.
Finally we place pivot at correct position by swapping a[i+1] and a[h] (or pivot)

a[] = {2,3,**5**,22,50,39,49,25,18,6} // 6 and 5 Swapped

Now, **5**is at its correct place. All elements smaller than 5 are before it and all elements greater than 5 are afterit.
Similarly rest of the passes will be executed and will provide the following output
Output of pass1

Array elements	**2**	**3**	**5**	22	50	39	49	25	18	6
indexes	0	1	2	3	4	5	6	7	8	9

Pass2

A[]={2,3} pivot=3

Array elements	**2**	**3**	**5**
indexes	0	1	2

a[]={22,50,39,49,25,18,6}pivot=6

Array elements	6	50	39	49	25	18	22
indexes	3	4	5	6	7	8	9

a[]={50,39,49,25,18,22}pivot=22

Array elements	18	22	49	25	50	39
indexes	4	5	6	7	8	9

a[]={18}pivot=18

Array elements	18	22
indexes	4	5

a[]={49,25,50,39},pivot=39

Array elements	25	39	50	49
indexes	6	7	8	9

a[]={25}, pivot=25

Array elements	25	39
indexes	6	7

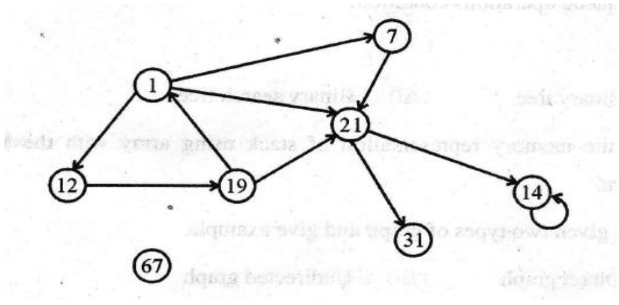
a[]={50,49},pivot=49

Array elements	49	50
indexes	8	9

Final sorted array using quick sort will be

Array elements	**2**	**3**	**5**	**6**	**18**	**22**	**25**	**39**	**49**	**50**
indexes	0	1	2	3	4	5	6	7	8	9

(d) **From the following graph, complete the answers:** **4M**



(i) Indegree of node 21
(ii) Adjacent node of 19

(iii) Path of 31
(iv) Successor of node 67

Ans.

(i) Indegree of node 21:
node 1, 7, 19

(i1) Adjacent node of 19:
node 1,21

(iii) Path of 3l:
Path1: 1-21-31
Path2: 1-7-21-31
Path3: 1-7-21-31

(iv) Successor of node 67: No Successor of node 67 since it is isolated node or not connected node in node.

Each correct answer 1M

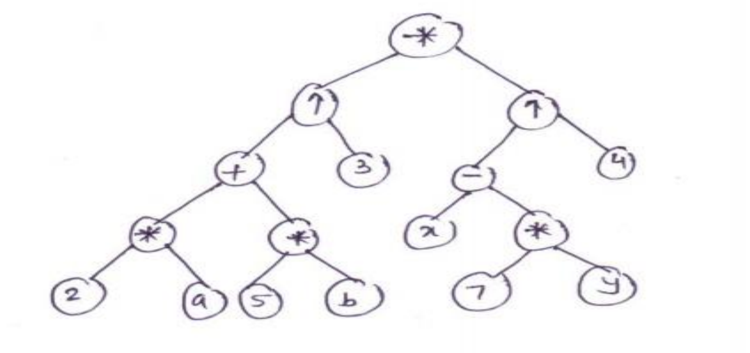
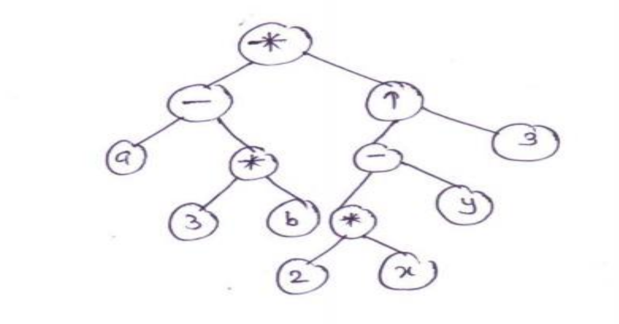
4. **Attempt any THREE of the following:** **12**

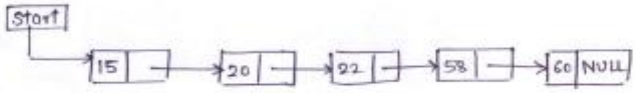
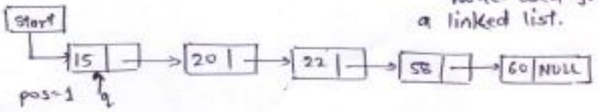
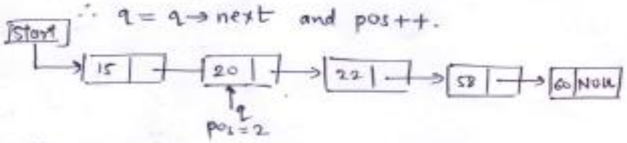
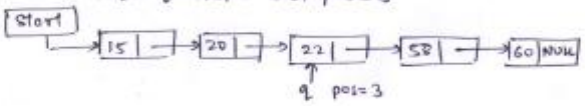
(a) **Differentiate between binary search and sequential search (linear search).** **4M**

Ans.

Sr. No.	Binary Search	Sequential search (linear search)
1	Input data needs to be sorted in Binary Search	Input data need not to be sorted in Linear Search.
2	In contrast, binary search compares key value with the middle element of an array and if comparison is unsuccessful then cuts down search to half.	A linear search scans one item at a time, without jumping to any item.
3	Binary search implements divide and conquer approach.	Linear search uses sequential approach.
4	In binary search the worst case complexity is O(log n) comparisons.	In linear search, the worst case complexity is O(n), comparisons.
5	Binary search is efficient for the larger array.	Linear search is efficient for the smaller array.

Any four points 1M each

	(b) **Ans.**	**Draw the tree structure of the following expressions:** **(i) $(2a+5b)^3 * (x-7y)^4$ (ii) $(a-3b) * (2x-y)^3$** **(i) $(2a+5b)^3 * (x-7y)^4$**  **(ii) $(a-3b) * (2x-y)^3$**  	**4M** ***Each correct tree structur e 2M***
	(c) **Ans.**	**Create a singly linked list using data fields 15, 20, 22, 58, 60. Search a node 22 from the SLL and show procedure step-by-step with the help of diagram from start to end.**	**4M**

		① With given data fields, singly linked list is created as follows  ② Operation – Search a node 22 from the above SLL a] Initially q = start where q is a pointer of type struct node used for traversing a linked list.  b] q != NULL and pos = 1 q → data ≠ key value i.e 15 ≠ 22 ∴ q = q → next and pos++.  c] q != NULL and pos = 2 q → data ≠ key value i.e 20 ≠ 22 ∴ q = q → next and pos = 3  q != NULL and pos = 3 q → data = = key value i.e 22 = = 22 ∴ node 22 is located at position 3 search is successful.	*Create linked list 1M* *Searching node procedure with diagram 3M*
	(d) Ans.	**Evaluate the following prefix expression:** **- * + 4 3 2 5 show diagrammatically each step of evaluation using stack.**	**4M**

Scanned Symbol	Operand 1	Operand 2	Value	Stack Content
5				5
2				5,2
3				5,2,3
4				5,2,3,4
+	4	3	12	5,2,12
*	12	2	24	5,24
-	24	5	19	19

Each correct step 1M

Result of above prefix expression evaluation - 19

(e) **Write an algorithm to delete a node from the beginning of a circular linked list.** **4M**

Ans.

Algorithm to delete a node from the beginning of a circular linked list

Consider the function delatbeg()

Correct algorithm 4M

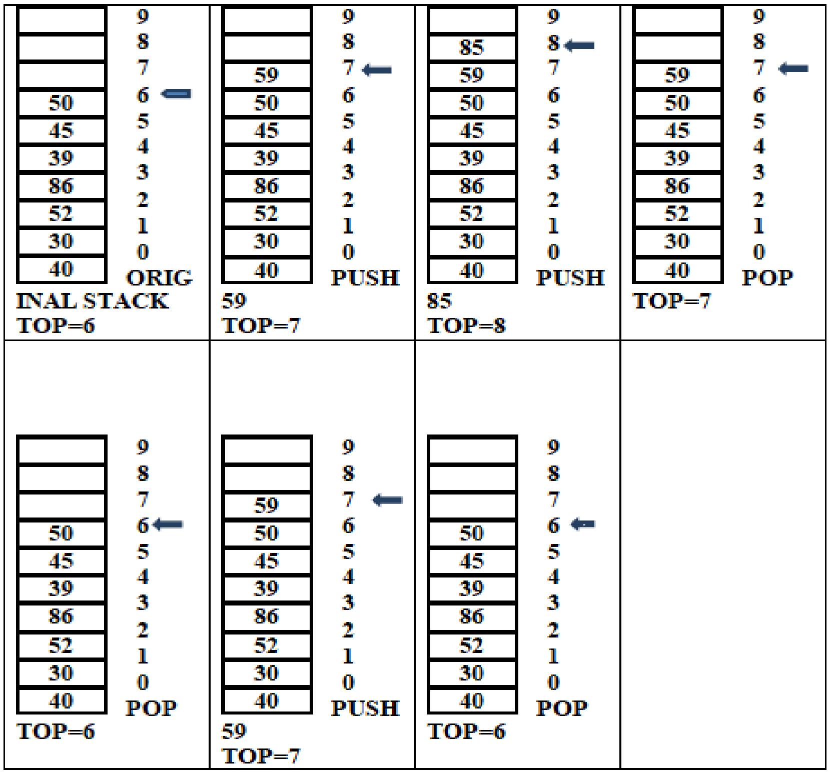
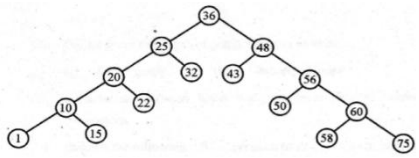
```
1. Start
2. Declare struct node *tmp,*q;
3. Set q=last->link;
4. While (q! = last)
   Do
   tmp = q;    // Identifies beginning node of Circular Linked List
   last->link=q->link; // Set the address field before deleting identified node
   free(tmp);           // Delete the beginning node
   End of While
5. last=NULL; // Set last= NULL if only one node is present in the Circular Linked List
6. End of function
```

5. **Attempt any TWO of the following:** **12**

(a) **Show the effect of PUSH and POP operation on to the stack of size 10. The stack contains 40, 30, 52, 86, 39, 45, 50 with 50 being at top of the stack. Show diagrammatically the effect of:** **6M**

(i) PUSH 59 (ii) PUSH 85
(iii) POP (iv) POP
(v) PUSH 59 (vi) POP

Sketch the final structure of stack after performing the above

	Ans.	**said operations.**  ORIGINAL STACK TOP=6 PUSH 59 TOP=7 PUSH 85 TOP=8 POP TOP=7 POP TOP=6 PUSH 59 TOP=7 POP TOP=6 40 30 52 86 39 45 50 59 85 0 1 2 3 4 5 6 7 8 9	***Each correct push/pop operation diagrammatically 1M***
	(b)	**Traverse the following tree by the in-order, pre-order and post-order methods:**  36 25 48 20 32 43 56 10 22 50 60 1 15 58 75	**6M**
	Ans.	INORDER (LVR) 1,10,15,20,22,25,32,36,43,48,50,56,58,60,75 PREORDER (VLR) 36,25,20,10,1,15,22,32,48,43,56,50,60,58,75 POST ORDER (LRV) 1,15,10,22,20,32,25,43,50,58,75,60,56,48,36	***in-order 2M*** ***pre-order2M*** ***post-order2M***

	(c) **Ans.**	**Write an algorithm to count number of nodes in singly linked list.** Let start is pointer variable which always stores address of first node in single linked list. If single linked list is empty then start will point to NULL. q is pointer variable used to store address of nodes in single linked list. Step 1: Start Step 2: [Assign starting address of single linked list to pointer q] q=start Step 3: [Initially set count of nodes in Linked list as zero] count=0 Step 4: [Check if Linked list empty or not] if start==NULL Display “Empty Linked List” go to step 6. Step 5: [Count number of nodes in single linked list] while q!=NULL count++ and q=q->next; Step 6: Display count (total number of nodes in single linked list) Step 7: stop	**6M** ***Correct algorith m 6M***
6.	 **(a)** **Ans.**	**Attempt any TWO of the following:** **Sort the following numbers in ascending order using Bubble sort. Given numbers: 29, 35, 3, 8, 11, 15, 56, 12, 1, 4, 85, 5 & write the output after each interaction.** Pass 1 Enter no of elements :12 Enter array elements :29 35 3 8 11 15 56 12 1 4 85 5 Unsorted Data: 29 35 3 8 11 15 56 12 1 4 85 5	**12** **6M**

		After pass 1 : 29 35 3 8 11 15 56 12 1 4 85 5 After pass 1 : 29 3 **35** 8 11 15 56 12 1 4 85 5 After pass 1 : 29 3 8 **35** 11 15 56 12 1 4 85 5 After pass 1 : 29 3 8 11 **35** 15 56 12 1 4 85 5 After pass 1 : 29 3 8 11 15 **35** 56 12 1 4 85 5 After pass 1 : 29 3 8 11 15 35 **56** 12 1 4 85 5 After pass 1 : 29 3 8 11 15 35 12 **56** 1 4 85 5 After pass 1 : 29 3 8 11 15 35 12 1 **56** 4 85 5 After pass 1 : 29 3 8 11 15 35 12 1 4 **56** 85 5 After pass 1 : 29 3 8 11 15 35 12 1 4 56 **85** 5 After pass 1 : 29 3 8 11 15 35 12 1 4 56 5 **85** Pass 2 After pass 2 : 3 29 8 11 15 35 12 1 4 56 5 85 After pass 2 : 3 8 **29** 11 15 35 12 1 4 56 5 85 After pass 2 : 3 8 11 **29** 15 35 12 1 4 56 5 85 After pass 2 : 3 8 11 15 **29** 35 12 1 4 56 5 85 After pass 2 : 3 8 11 15 29 **35** 12 1 4 56 5 85 After pass 2 : 3 8 11 15 29 12 **35** 1 4 56 5 85 After pass 2 : 3 8 11 15 29 12 1 **35** 4 56 5 85 After pass 2 : 3 8 11 15 29 12 1 4 **35** 56 5 85 After pass 2 : 3 8 11 15 29 12 1 4 35 **56** 5 85 After pass 2 : 3 8 11 15 29 12 1 4 35 5 **56** 85 Pass 3 After pass 3 : 3 8 11 15 29 12 1 4 35 5 56 85 After pass 3 : 3 8 11 15 29 12 1 4 35 5 56 85 After pass 3 : 3 8 11 15 29 12 1 4 35 5 56 85 After pass 3 : 3 8 11 15 29 12 1 4 35 5 56 85 After pass 3 : 3 8 11 15 12 **29** 1 4 35 5 56 85 After pass 3 : 3 8 11 15 12 1 **29** 4 35 5 56 85 After pass 3 : 3 8 11 15 12 1 4 **29** 35 5 56 85 After pass 3 : 3 8 11 15 12 1 4 29 35 5 56 85 After pass 3 : 3 8 11 15 12 1 4 29 5 **35** 56 85 Pass 4 After pass 4 : 3 8 11 15 12 1 4 29 5 35 56 85 After pass 4 : 3 8 11 15 12 1 4 29 5 35 56 85 After pass 4 : 3 8 11 **15** 12 1 4 29 5 35 56 85 After pass 4 : 3 8 11 12 **15** 1 4 29 5 35 56 85	***Correct passes 6M (For 4 passes 3M shall be awarded)***

		After pass 4 : 3 8 11 12 1 **15** 4 29 5 35 56 85 After pass 4 : 3 8 11 12 1 4 **15** 29 5 35 56 85 After pass 4 : 3 8 11 12 1 4 15 **29** 5 35 56 85 After pass 4 : 3 8 11 12 1 4 15 5 **29** 35 56 85 Pass 5 After pass 5 : 3 8 11 12 1 4 15 5 29 35 56 85 After pass 5 : 3 8 11 12 1 4 15 5 29 35 56 85 After pass 5 : 3 8 11 **12** 1 4 15 5 29 35 56 85 After pass 5 : 3 8 11 1 **12** 4 15 5 29 35 56 85 After pass 5 : 3 8 11 1 4 **12** 15 5 29 35 56 85 After pass 5 : 3 8 11 1 4 12 **15** 5 29 35 56 85 After pass 5 : 3 8 11 1 4 12 5 **15** 29 35 56 85 Pass 6 After pass 6 : 3 8 11 1 4 12 5 15 29 35 56 85 After pass 6 : 3 8 **11** 1 4 12 5 15 29 35 56 85 After pass 6 : 3 8 1 **11** 4 12 5 15 29 35 56 85 After pass 6 : 3 8 1 4 **11** 12 5 15 29 35 56 85 After pass 6 : 3 8 1 4 11 **12** 5 15 29 35 56 85 After pass 6 : 3 8 1 4 11 5 **12** 15 29 35 56 85 Pass 7 After pass 7 : 3 8 1 4 11 5 12 15 29 35 56 85 After pass 7 : 3 1 8 4 11 5 12 15 29 35 56 85 After pass 7 : 3 1 4 8 11 5 12 15 29 35 56 85 After pass 7 : 3 1 4 8 **11** 5 12 15 29 35 56 85 After pass 7 : 3 1 4 8 5 **11** 12 15 29 35 56 85 Pass 8 After pass 12 : **1** 3 4 8 5 11 12 15 29 35 56 85 **Sorted elements are** 1 3 4 8 5 11 12 15 29 35 56 85	
	(b) **Ans.**	**Evaluate the following postfix expression:** **5 7 + 6 2 - ***	**6M**

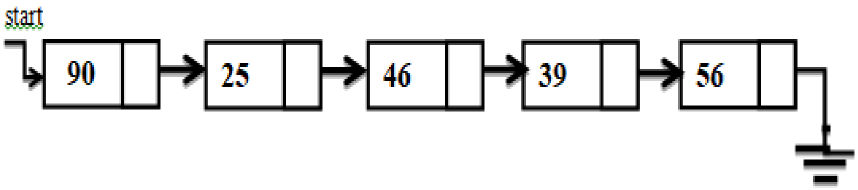
Symbols to be scanned	STACK					Expression Evaluation and Result
	4	3	2	1	0	
5					5	----
7				7	5	----
+					12	7+5=12
6				6	12	----
2			2	6	12	6-2=4
-				4	12	----
*					48	12*4

Correct evaluative 6M

(c) **Create a singly linked list using data fields 90, 25, 46, 39, 56. Search a node 40 from the SLL and show procedure step-by-step with the help of diagram from start to end.** **6M**

Ans. To Search a data field in singly linked list, need to start searching the data field from first node of singly linked list.

ORIGINAL LIST:



List creation 1M

SEARCHING A NODE

STEP 1:

Compare 40 with 90

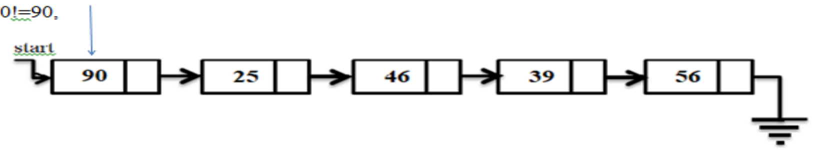
40!=90,

SEARCHING A NODE

STEP 1:

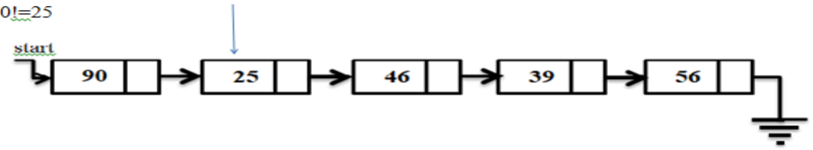
Compare 40 with 90

40!=90,



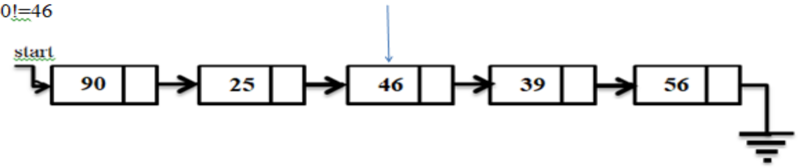
STEP 2:

40!=25



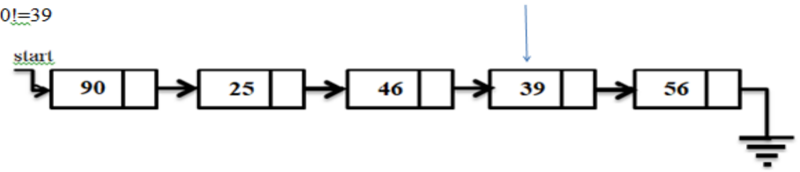
STEP 3:

40!=46



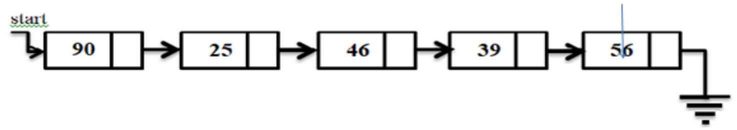
STEP 4:

40!=39



Step 5:

40!=56



Node not found. Search unsuccessful

Comparison with each node diagrammatically 1M

22317

11920

3 Hours / 70 Marks Seat No. □□□□□□□□

Instructions : (1) All Questions are *compulsory.*

(2) Answer each next main Question on a new page.

(3) Illustrate your answers with neat sketches wherever necessary.

(4) Figures to the right indicate full marks.

(5) Assume suitable data, if necessary.

(6) Mobile Phone, Pager and any other Electronic Communication devices are not permissible in Examination Hall.

Marks

1. Attempt any FIVE of the following : 10

(a) Write any four operations that can be performed on data structure.

(b) Define the terms 'overflow' and 'underflow' with respect to stack.

(c) Define the following terms w.r.t. tree : (i) In-degree, (2) Out-degree

(d) Evaluate the following arithmetic expression P written in postfix notation :

P : 4, 2, ^, 3, *, 3, -, 8, 4, /, +

(e) Describe directed and undirected graph.

(f) Give classification of data structure.

(g) Define queue. State any two applications where queue is used.

2. Attempt any THREE of the following : 12

(a) Sort the given numbers in ascending order using Radix sort :

348, 14, 641, 3851, 74

(b) Write an algorithm to insert a new node at the beginning and end of the singly linked list.

(c) Explain the concept of circular Queue along with its need.

(d) Draw a binary search tree for the given numbers :

50, 33, 44, 22, 77, 35, 60, 40.

3. Attempt any THREE of the following : 12

(a) Explain time and space complexity with an example.

(b) Convert the following infix expression to postfix expression using stack and show the details of stack in each step.

((A+B)*D)^(E-F)

(c) Implement a 'C' program to search a particular data from the given array using Linear Search.

(d) Draw an expression tree for the following expression :

$(a - 2b + 5c)^2 * (4d = 6e)^5$

4. Attempt any THREE of the following : 12

(a) Find the position of element 21 using Binary Search method in Array 'A' given below :

A = {11, 5, 21, 3, 29, 17, 2, 45}

(b) Differentiate between tree and graph. (Any 4 points)

(c) Construct a singly linked list using data fields 21, 25, 96, 58, 74 and show procedure step-by-step with the help of diagram start to end.

(d) Show the effect of PUSH and POP operations on the stack of size 10.

PUSH(10)

PUSH(20)

POP

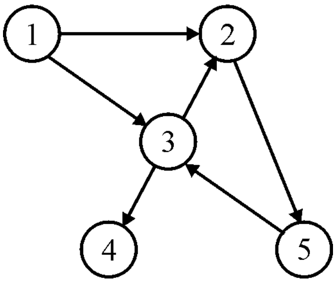
PUSH(30)

(e) Compare Linked List and Array. (any 4 points)

5. Attempt the following any TWO of the following : 12

(a) Implement a 'C' program to insert element into the queue and delete the element from the queue.

(b) Consider the graph given in following figure and answer given questions.



(1) All simple path from 1 to 5

(2) In-degree of and out-degree of 4

(3) Give adjacency matrix for the given graph.

(4) Give adjacency list representation of the given graph.

(c) Write an algorithm to search a particular node in the given linked list.

6. Attempt any TWO of the following : 12

(a) Elaborate the steps for performing selection sort for given elements of array.

A = {37, 12, 4, 90, 49, 23, –19}

(b) Explain the concept of recursion using stack.

(c) Show with suitable diagrams how to delete a node from singly linked list at the beginning, in between and at the end of the list.

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

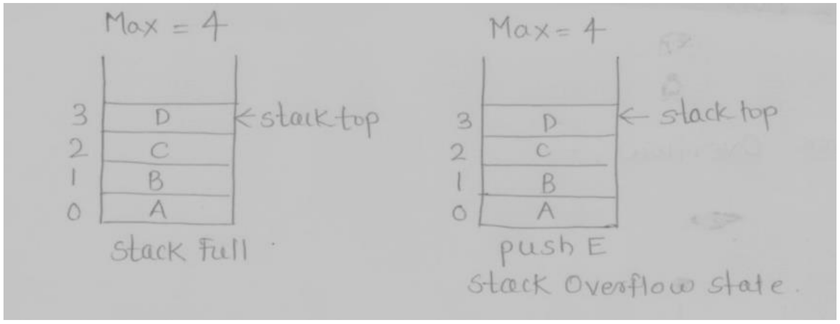
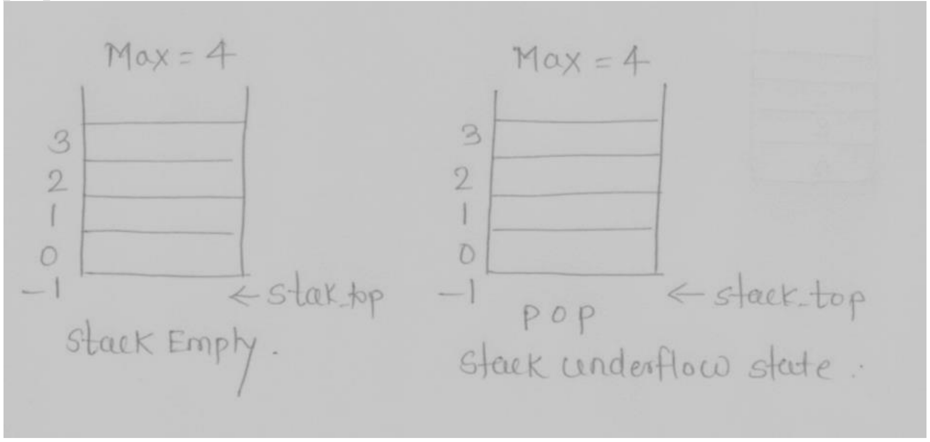
Winter – 19 EXAMINATION

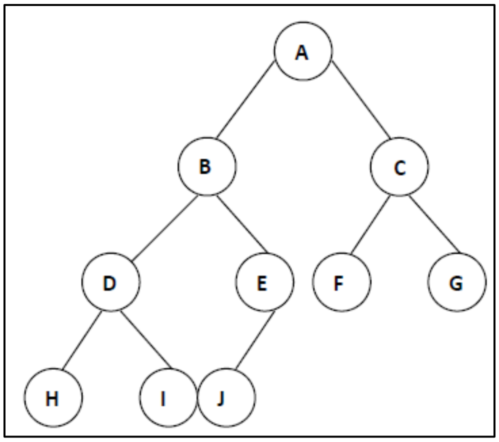
Subject Name: Data Structure Using 'C' **Model Answer** **Subject Code: 22317**

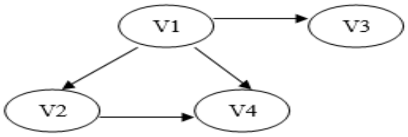
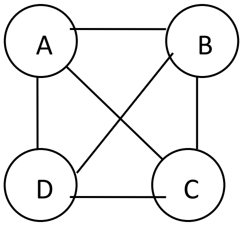
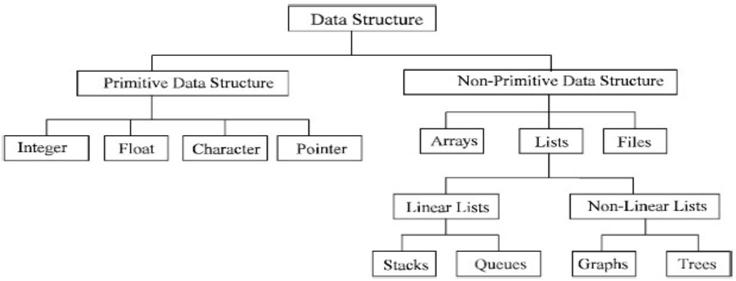
Important Instructions to examiners:

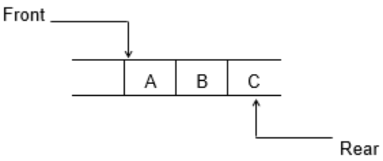
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling 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.		**Attempt any Five of the following:**	10M
	a	**Write any four operations that can be performed on data structure.**	2M
	Ans	1. **Data structure operations (Non Primitive)** 2. **Inserting:** Adding a new data in the data structure is referred as insertion. 3. **Deleting:** Removing a data from the data structure is referred as deletion. 4. **Sorting:** Arranging the data in some logical order (ascending or descending, numerically or alphabetically). 5. **Searching:** Finding the location of data within the data structure which satisfy the searching condition. 6. **Traversing:** Accessing each data exactly once in the data structure so that each data item is traversed or visited. 7. **Merging:** Combining the data of two different sorted files into a single sorted file. 8. **Copying:** Copying the contents of one data structure to another. 9. **Concatenation:** Combining the data from two or more data structure. **OR**	2 M for any 4 Operation

		Data structure operations (Primitive) 1. Creation: To create new Data Structure 2. Destroy: To delete Data Structure 3. Selection: To access (select) data from the data structure 4. Updating: To edit or change the data within the data structure.	
	b	**Define the term overflow and underflow with respect to stack.**	**2M**
	Ans	**Stack overflow**: When a stack is full and push operation is performed to insert a new element, stack is said to be in overflow state.  Max = 4 Max = 4 3 2 1 0 D C B A ←stack top Stack Full push E Stack Overflow state. **Stack underflow**: When there is no element in a stack (stack empty) and pop operation is called then stack is said to underflow state.  Max = 4 Max = 4 3 2 1 0 −1 ←stak top Stack Empty. pop ←stack top Stack underflow state.	1 M for stack overflow and 1M for stack underflow
	c	**Define the following term w.r.t. tree: (i) In-degree (ii) out-degree.**	**2M**
	Ans	**In -degree:** Number of edges coming towards node is in-degree of node. For e.g. : In degree of node B is 1 **Out -degree:** Number of edges going out from node is out -degree of node. For e.g. Out Degree of is node D is 2	1 M for each correct definition

<table>
<tr><td></td><td></td><td>
</td><td></td></tr>
<tr><td></td><td>d</td><td><b>Evaluate the following arithmetic expression P written in postfix notation: P : 4, 2, ^, 3, *,3,-,8,4 ,/,+</b></td><td>2M</td></tr>
<tr><td></td><td>Ans</td><td>
<table>
<tr><th>Sr. No.</th><th>Symbol Scanner</th><th>STACK</th></tr>
<tr><td>1</td><td>4</td><td>4</td></tr>
<tr><td>2</td><td>2</td><td>4, 2</td></tr>
<tr><td>3</td><td>^</td><td>16</td></tr>
<tr><td>4</td><td>3</td><td>16, 3</td></tr>
<tr><td>5</td><td>*</td><td>48</td></tr>
<tr><td>6</td><td>3</td><td>48,3</td></tr>
<tr><td>7</td><td>-</td><td>45</td></tr>
<tr><td>8</td><td>8</td><td>45,8</td></tr>
<tr><td>9</td><td>4</td><td>45,8,4</td></tr>
<tr><td>10</td><td>/</td><td>45,2</td></tr>
<tr><td>11</td><td>+</td><td>47</td></tr>
</table>
</td><td>2 M for correct answer</td></tr>
</table>

	e	**Describe directed and undirected graph.**	**2M**
	Ans	**Direct Graph:** A directed graph is defined as the set of ordered pair of vertices and edges where each connected edge has **assigned a direction.**  **Undirected Graph :** An undirected graph G is a graph in which each edge e is not assigned a direction.  	1M for each definition with diagram
	f	**Give classification of data structure.**	**2M**
	Ans	 Data Structure Primitive Data Structure Non-Primitive Data Structure Integer Float Character Pointer Arrays Lists Files Linear Lists Non-Linear Lists Stacks Queues Graphs Trees	2 M for diagram
	g	**Define queue. State any two applications where queue is used.**	**2M**
	Ans	A **Queue** is an ordered collection of items. It has two ends, front and rear. Front end is used to delete element from queue. Rear end is used to insert an element in queue. Queue has two ends; the element entered first in the queue is removed first from the queue. So it is called as FIFO list.	1M for definition, 1M for applications (any two)



APPLICATIONS OF QUEUES

1. Round Robin Technique for processor scheduling is implemented using queues.

2. All types of customer service (like railway ticket reservation) center software's are designed using queues to store customer's information.

3. Printer server routines are designed using queues. A number of users share a printer using printer server (a dedicated computer to which a printer is connected), the printer server then spools all the jobs from all the users, to the server's hard disk in a queue. From here jobs are printed one-by-one according to their number in the queue.

2.		**Attempt any Three of the following:**	**12M**
	a	**Sort the given number in ascending order using Radix sort:** **348, 14, 641, 3851, 74.**	**4M**
	Ans	**Pass 1:**	4 M for correct answer

	0	1	2	3	4	5	6	7	8	9
0348									0348	
0014					0014					
0641		0641								
3851		3851								
0074					0074					

0641,3851,0014,0074,0348

Pass 2:

	0	1	2	3	4	5	6	7	8	9
0641					0641					
3851						3851				
0014		0014								
0074								0074		
0348					0348					

0014,0641,0348,3851,0074

Pass 3:

	0	1	2	3	4	5	6	7	8	9
0014	0014									
0641							0641			
0348				0348						
3851									3851	
0074	0074									

0014,0074,0348,0641,3851

Pass 4:

	0	1	2	3	4	5	6	7	8	9
0014	0014									
0074	0074									
0348	0348									
0641	0641									
3851					3851					

Sorted Elements are: 14, 74, 348, 641, 3851

b **Write an algorithm to insert a new node at the beginning and end of the singly linked list.** **4M**

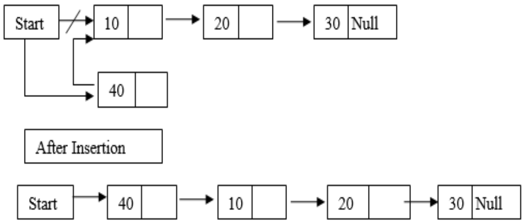
Ans

1. Algorithm for inserting a node at the beginning

```
Insert first(start, item)

1.  [check the overflow]
        if Ptr=NULL then print 'Overflow'

        exit

        else

        Ptr=(node *) malloc (size of (node))

//create new node from memory and assign its address to ptr
```

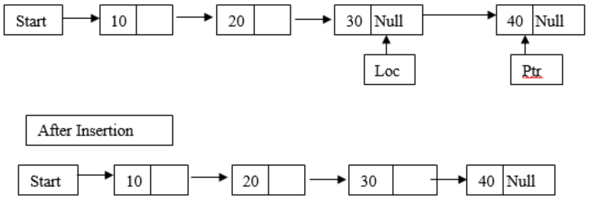
2M for Algorithm for inserting a node at the beginning
2M for Algorithm for Inserting A Node at the End

```
        End if
    2. set Ptr->num = item
    3. set Ptr->next=start
    4. set start=Ptr
```



2. Algorithm for Inserting A Node at the End

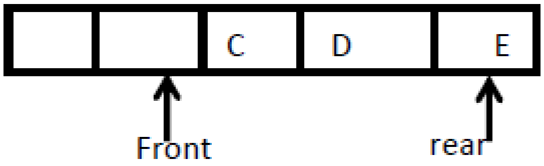
```
insert last (start, item)
1. [check for overflow]
   If Ptr=NULL, then print 'Overflow'
   exit
   else
   Ptr=(node * ) malloc (sizeof (node));
   end if
2. set Ptr->info=item
3. set Ptr->next=NULL
4. if start=NULL and if then set start=P
5. set loc=start
6. repeat step 7 until loc->next != NULL
7. set loc=loc->next
8. set loc->next=P
```



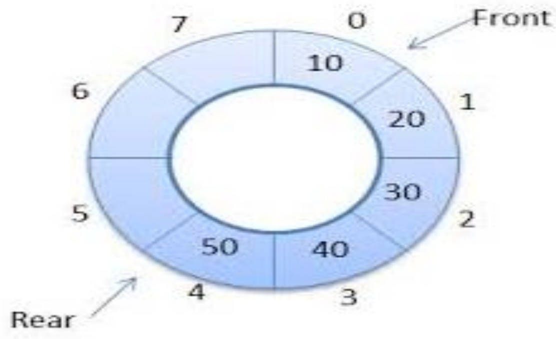
	c	**Explain the concept of circular Queue along with its need.**	**4M**
	Ans	- Circular queue are the queues implemented in circular form rather than in a straight line. - Circular queues overcome the problem of unutilized space in linear queue implemented as an array. - The main disadvantage of linear queue using array is that when	3 M for explanation and & 1M for need

elements are deleted from the queue, new elements cannot be added in their place in the queue, i.e. the position cannot be reused. After rear reaches the last position, i.e. MAX-1 in order to reuse the vacant positions, we can bring rear back to the 0th position, if it is empty, and continue incrementing rear in same manner as earlier.

- Thus rear will have to be incremented circularly. For deletion, front will also have to be incremented circularly.
- Rear can be incremented circularly by the following code. If ((rear == MAX-1) and (front !=0) Rear =0; Else Rear= rear +1; Example: Assuming that the queue contains three elements.



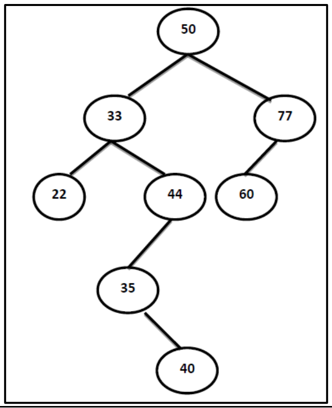
- Now we insert an element F at the beginning by bringing rear to the first position in the queue. this can be represented circularly as shown.



Need of Circular Queue:

- Circular queues overcome the problem of unutilized space in linear queue implemented as an array.
- The element can be stored efficiently in an array so as to wrap around so that the end of queue is followed by front of the queue.

	d	**Draw a binary search tree for the given number. 50, 33, 44, 22, 77, 35, 60, 40.**	**4M**
	Ans		4 M for correct answer



3. **Attempt any Three of the following:** **12M**

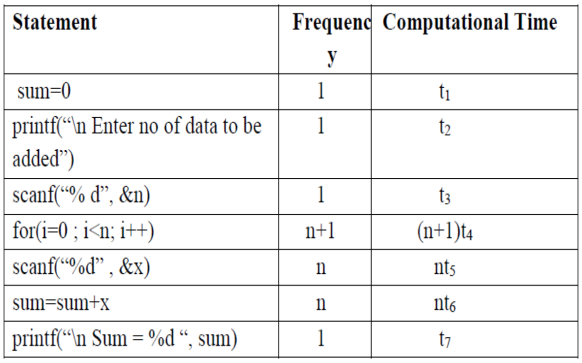
a **Explain time and space complexity with an example.** **4M**

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.

2M for Time Complexity and
2M for space complexity

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++)
```



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

Space Complexity: Total amount of computer memory required by an algorithm to complete its execution is called as space complexity of that algorithm. 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.

If the amount of space required by an algorithm is increased with the increase of input value, then that space complexity is said to be Linear Space Complexity.

Example:

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

In the 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 total amount of memory required depends on the value of 'n'. As 'n' value increases the space required also increases proportionately. This type of space complexity is said to be **Linear Space Complexity.**

OR

Time complexity:- Time complexity of a program/algorithm is the amount of computer time that it needs to run to completion. While calculating time complexity, we develop frequency count for all key statements which are important and basic instructions of an algorithm.

Example: Consider three algorithms given below:-

		Algorithm A: - a=a+1 Algorithm B: - for x = 1 to n step 1 a=a+1 Loop Algorithm C:- for x=1 to n step 1 for y=1 to n step 1 a=a+1 Loop Frequency count for algorithm A is 1 as a=a+1 statement will execute only once. Frequency count for algorithm B is n as a=a+1 is key statement executes n time as the loop runs n times. Frequency count for algorithm C is n as a=a+1 is key statement executes n2 time as the inner loop runs n times, each time the outer loop runs and the outer loop also runs for n times. **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:- **Fixed space requirements**: - It includes space for instructions, for simple variables, fixed size structured variables and constants. **Variable time requirements**: - It consists of space needed by structured variables whose size depends on particular instance of variables. Example: - additional space required when function uses recursion.	
	b	**Convert the following infix expression to postfix expression using stack and show the details of stack in each step.((A+B)*D)^(E-F)**	**4M**
	Ans	**infix expression:** **(((A+B)*D)^(E-F))**	Correct answer-4M

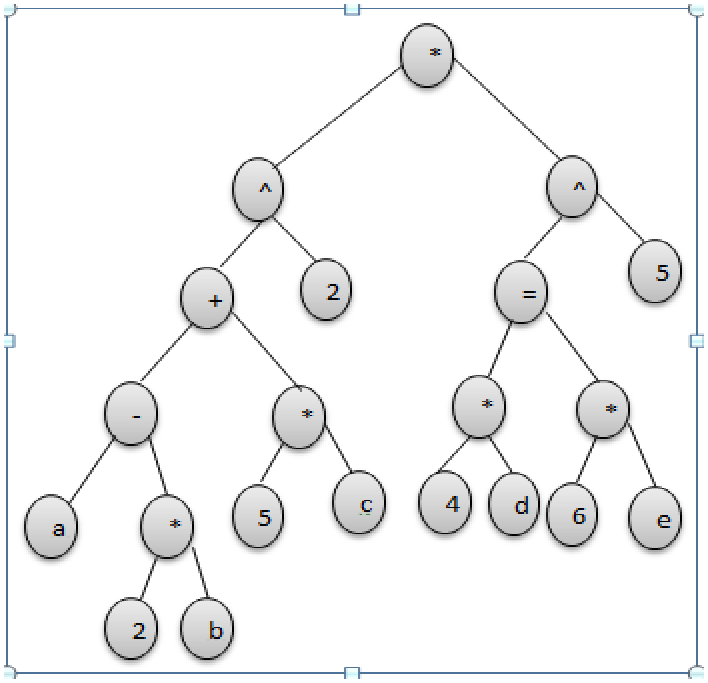
Current Symbol	Operator Stack	Postfix array
(	(	Empty
(	((	Empty
(	(((	Empty
A	(((	A
+	(((+	A
B	(((+	AB
)	((	AB+
*	((*	AB+
D	((*	AB+D
)	(	AB+D*
^	(^	AB+D*
(	(^(	AB+D*
E	(^(	AB+D*E
-	(^(-	AB+D*E
F	(^(-	AB+D*EF
)	(^	AB+D*EF-
)	EMPTY STACK	AB+D*EF-^

Postfix expression: AB+D*EF-^

	c	**Implement a ‘C’ program to search a particular data from the given array using Linear Search.**	4M
	Ans	**Program:-**	

		# include<stdio.h>
		#include <conio.h>
		void main ()
		{
		int a[10], n, key,i,c=0;
		clrscr( );
		printf (“Enter number of array elements\n”);
		scanf (“%d”, &n);
		printf (“Enter array elements\n”);
		for (i=0; i< n; i++)
		scanf (“%d”, &a[i]);
		prinntf (“Enter key value\n”);
		scanf (“%d”, &key);
		
		for(i=0;i<n-1;i++)
		{
		
		if (key == a[i])
		{
		c=1;
		printf (“%d is found at location %d\n”, key, i+1);
		break;
		}
		
		}
		if (c==0)
		printf (“%d not present in the list\n”,key);
		getch();
		}
			2M for logic And 2 M for syntax

	d	**Draw an expression tree for the following expression:** **(a-2b+5e) ^{2} * (4d=6e) ^{5}.**	**4M**
	Ans		Correct Expression tree-4M



4. **Attempt any Three of the following:** **12M**

a **Find the position of element 21 using binary search method in array 'A' given below: A=(11,5,21,3,29,17,2,45}** **4M**

Ans **Given Array**

Each correct step -2M each

11	**5**	**21**	**3**	**29**	**17**	**2**	**45**

Sorted Array for input:

2	**3**	**5**	**11**	**17**	**21**	**29**	**45**

Key element to be searched=21

Step1

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
2	**3**	**5**	**11**	**17**	**21**	**29**	**45**

l=0 and u=n-1 =7

mid=(l+u)/2 = 7/2 = 3

a[mid]=11 not equal to 21
and

21 > 11 l=mid+1 = 4 and u = 7

Step 2:

4	5	6	7
17	21	29	45

l=4 and u =7

mid= 11/2 = 5

a[mid]=21 equal to key element 21

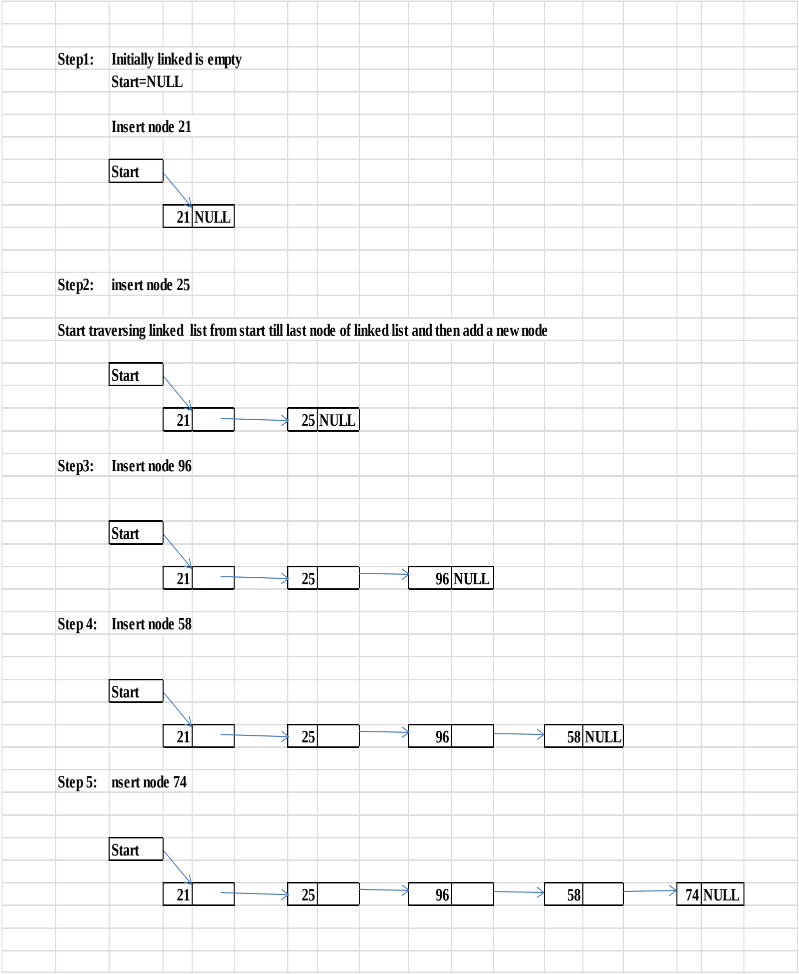
therefore key element 21 is fount un array at position 6

b **Difference between tree and graph(Any 4 points)** **4M**

Ans

Tree	Graph
Tree is special form of graph i.e. minimally connected graph and having only one path between any two vertices.	In graph there can be more than one path i.e. graph can have uni-directional or bi-directional paths (edges) between nodes
Tree is a special case of graph having no loops, no circuits and no self-loops.	Graph can have loops, circuits as well as can have self-loops.
Tree traversal is a kind of special case of traversal of graph. Tree is traversed in Pre-Order, In-Order and Post-Order	Graph is traversed by DFS: Depth First Search and in BFS : Breadth First Search algorithm
Different types of trees are: Binary Tree, Binary Search Tree, AVL tree, Heaps.	There are mainly two types of Graphs: Directed and Undirected graphs.

Any correct 4 points- 4M

<table>
<tr><td></td><td></td><td><table>
<tr><td>Tree applications: sorting and searching like Tree Traversal & Binary Search.</td><td>Graph applications : Coloring of maps, in OR (PERT & CPM), algorithms, Graph coloring, job scheduling, etc.</td></tr>
<tr><td>Tree always has n-1 edges.</td><td>In Graph, no. of edges depends on the graph.</td></tr>
<tr><td>Tree is a hierarchical model.</td><td>Graph is a network model.</td></tr>
</table></td><td></td></tr>
<tr><td></td><td>C</td><td><b>Construct a singly linked list using data fields 21 25 96 58 74 and show procedure step-by-step with the help of diagram start to end.</b></td><td><b>4M</b></td></tr>
<tr><td></td><td><b>Ans</b></td><td>
Step1: Initially linked is empty
Start=NULL
Insert node 21
Step2: insert node 25
Start traversing linked list from start till last node of linked list and then add a new node
Step3: Insert node 96
Step 4: Insert node 58
Step 5: nsert node 74
</td><td>correct construction - 3M and explaination- 1M</td></tr>
<tr><td></td><td>d</td><td><b>Show the effect of PUSH and POP operation on the stack of size 10.
PUSH(10)
PUSH(20)</b></td><td><b>4M</b></td></tr>
</table>

POP
PUSH(30)

Ans **Initial Stack empty**

Each correct step-1M

	stack[9]	
	stack[8]	
	stack[7]	
	stack[6]	
	stack[5]	
	stack[4]	
	stack[3]	
	stack[2]	
	stack[1]	
	stack[0]	top= -1

Step 1:

PUSH(0)

top=top+1 stack[0]=10

	stack[9]	
	stack[8]	
	stack[7]	
	stack[6]	
	stack[5]	
	stack[4]	
	stack[3]	
	stack[2]	
	stack[1]	
10	stack[0]	top=0

Step 2:

PUSH(0)

top=top+1 stack[1]=20

	stack[9]	
	stack[8]	
	stack[7]	
	stack[6]	
	stack[5]	
	stack[4]	
	stack[3]	
	stack[2]	
20	stack[1]	top=1
10	stack[0]	

Step 3:

POP

top=top-1 20 is deleted

	stack[9]	
	stack[8]	
	stack[7]	
	stack[6]	
	stack[5]	
	stack[4]	
	stack[3]	
	stack[2]	
	stack[1]	
10	stack[0]	top=0

Step 4:

PUSH(0)

top=top+1 stack[1]=30

	stack[9]	
	stack[8]	
	stack[7]	
	stack[6]	
	stack[5]	
	stack[4]	
	stack[3]	
	stack[2]	
30	stack[1]	top=1
10	stack[0]	

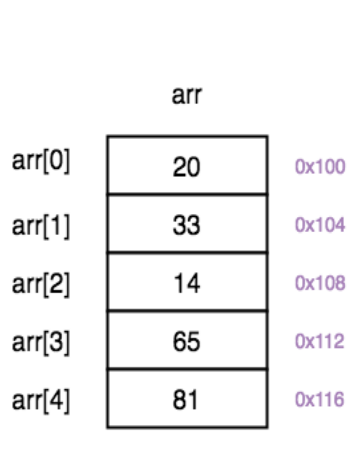
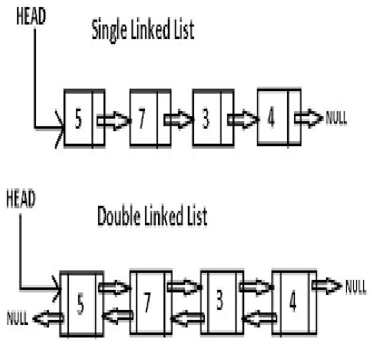
e **Compare Linked List and Array (any 4 points).** **4M**

Ans

Linked List	Array
Array is a collection of elements of similar data type.	Linked List is an ordered collection of elements of same type, which are connected to each other using pointers.
Array supports Random Access, which means elements can be accessed directly using their index, like arr[0] for 1st element, arr[6] for 7th element etc.	Linked List supports Sequential Access, which means to access any element/node in a linked list; we have to sequentially traverse the complete linked list, up to that element.

1M for each valid difference

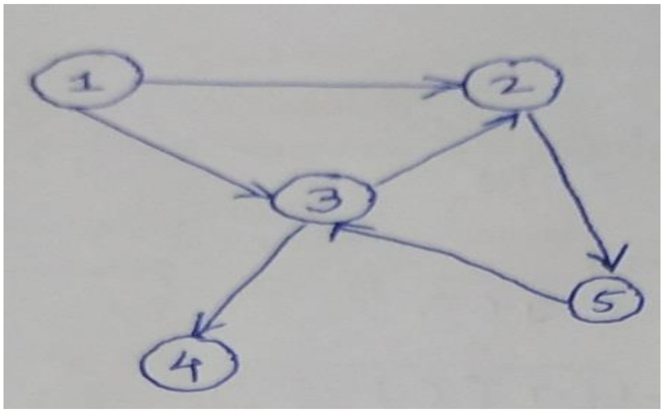
			Hence, accessing elements in an array is fast with a constant time complexity of O (1).	To access nth element of a linked list, time complexity is O (n).		
			In array, Insertion and Deletion operation takes more time, as the memory locations are consecutive and fixed.	In case of linked list, a new element is stored at the first free and available memory location, with only a single overhead step of storing the address of memory location in the previous node of linked list. Insertion and Deletion operations are fast in linked list.		
			Memory is allocated as soon as the array is declared, at compile time. It's also known as Static Memory Allocation.	Memory is allocated at runtime, as and when a new node is added. It's also known as Dynamic Memory Allocation.		
			In array, each element is independent and can be accessed using it's index value	In case of a linked list, each node/element points to the next, previous, or maybe both nodes.		
			Array can single dimensional, two dimensional or multidimensional	Linked list can be Linear (Singly), Doubly or Circular linked list.		
			Size of the array must be specified at time of array declaration.	Size of a Linked list is variable. It grows at runtime, as more nodes are added to it.		
			Array gets memory allocated in the Stack section	Whereas, linked list gets memory allocated in Heap section.		

		 arr arr[0] arr[1] arr[2] arr[3] arr[4] 20 33 14 65 81 0x100 0x104 0x108 0x112 0x116 Array representation  HEAD Single Linked List 5 7 3 4 NULL HEAD Double Linked List NULL 5 7 3 4 NULL **Linked list presentation**	
5.		**Attempt any Three of the following:**	**12- M**
	a	**Implement a 'C' program to insert element into the queue and delete the element from the queue.**	**6M**

	Ans	#include<stdio.h>
		#include<conio.h>
		#define max 5
		void main()
		{
		int a[max],front,rear,no,ch,i;
		clrscr();
		front=rear=-1;
		do
		{
		printf("\n 1.INSERT");
		printf("\t 2.DELETE");
		printf("\t 3.EXIT");
		printf("\n\n ENTER YOUR CHOICE:- ");
		scanf("%d",&ch);
		switch(ch)
		{
		case 1:
		printf("\n ENTER ITEM TO BE INSERTED :- ");
		scanf("%d",&no);
		if(rear==max-1)
		{
		printf ("\n QUEUE IS FULL.");
			Insert logic-3M, delete logic-3M

```
break;
}
rear=rear+1;
a[rear]=no;
if(front==-1)
front=0;
break;
case 2:
if(front==-1)
{
printf ("\n QUEUE IS EMPTY.");
break;
}
no=a[front];
printf("\n DELETED ELEMENT IS:- %d",no);
if(front==rear)
front=rear=-1;
else
front=front+1;
break;
case 3:
exit(0);
}
printf("\n\n DO YOU WANT TO CONTINUE:(1 FOR YES/2 FOR NO):-");
scanf("%d",&ch);
}while(ch==1);
getch();
}
```

b **Consider the graph given in following figure and answer given questions.** **6M**



1)All simple path from 1 to 5

2)In-degree of and out-degree of 4

3) Give Adjacency matrix for the given graph.

4) Give Adjacency list representation of the given graph.

Ans

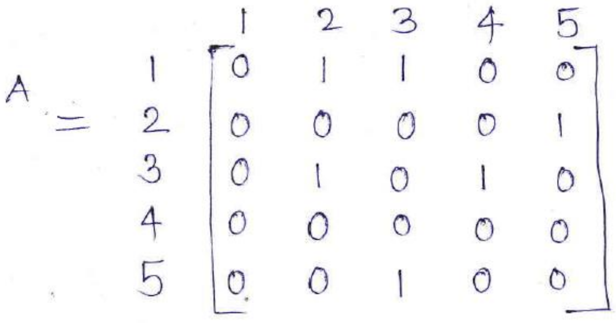
i) Nodes: **1-2-5**

ii) Nodes: **1-3-2-5**

2)

In degree of node 4- **1**, **Out degree** of node 4 - **0**

3)**Correct adjacency matrix:**



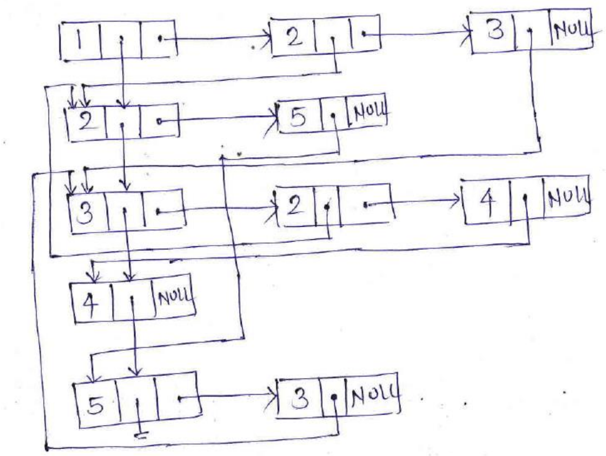
$$A = \begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 \\ \hline \end{array}$$

4) Adjacency list representation

Node	Adjacent nodes
1	2,3
2	5
3	2,4
4	NIL
5	3

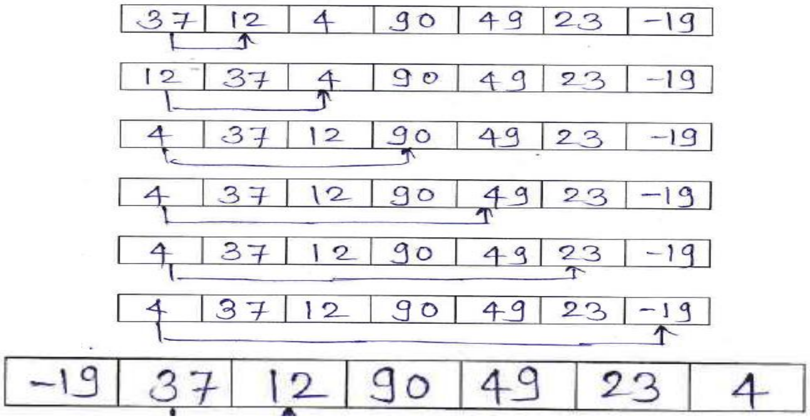
Simple path: - Each path ½ M
Each degree ½ M

Correct adjacency matrix: 2M
Adjacency list representation -2M

		Representation:  1 2 3 NULL 2 5 NULL 3 2 4 NULL 4 NULL 5 3 NULL	
	c	**Write an algorithm to search a particular node in the give linked list.**	**6M**
	Ans	**Assumption:** Node contains two fields: info and next pointer start pointer : Header node that stores address of first node step 1: start step 2: Declare variable no, flag and pointer temp step 3: Input search element step 4: Initialize pointer temp with the address from start pointer.(temp=start), flag with 0 step 5: Repeat step 6 till temp != NULL step 6: compare: temp->info = no then set flag=1 and go to step 7 otherwise increment pointer temp and go to step5 step 7: compare: flag=1 then display "Node found" otherwise display "node not found" step 8: stop	Correct steps of algorithm-6M
6.		**Attempt any Three of the following:**	**12M**
	a	**Elaborate the steps for performing selection sort for given elements of array. A={37,12,4,90,49,23,-19}**	**6M**

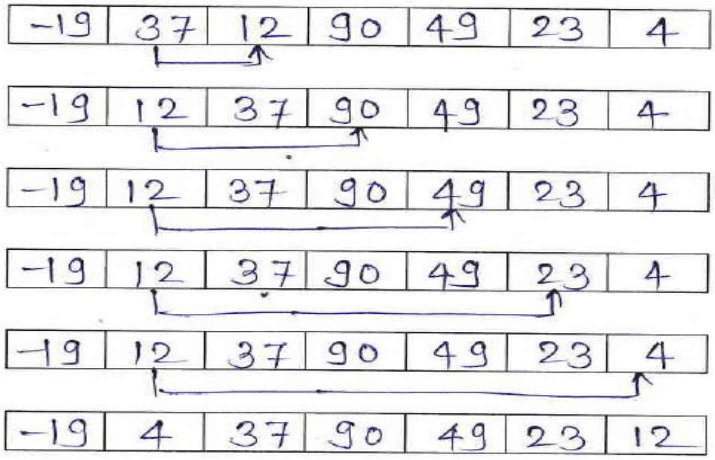
Ans

Pass 1



37	12	4	90	49	23	-19
12	37	4	90	49	23	-19
4	37	12	90	49	23	-19
4	37	12	90	49	23	-19
4	37	12	90	49	23	-19
4	37	12	90	49	23	-19
-19	37	12	90	49	23	4

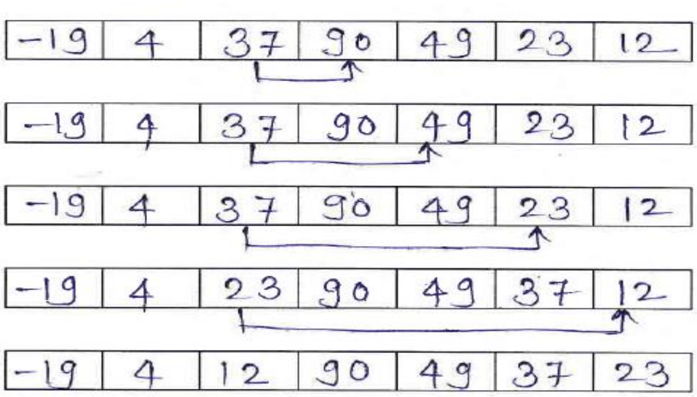
Pass 2



-19	37	12	90	49	23	4
-19	12	37	90	49	23	4
-19	12	37	90	49	23	4
-19	12	37	90	49	23	4
-19	12	37	90	49	23	4
-19	4	37	90	49	23	12

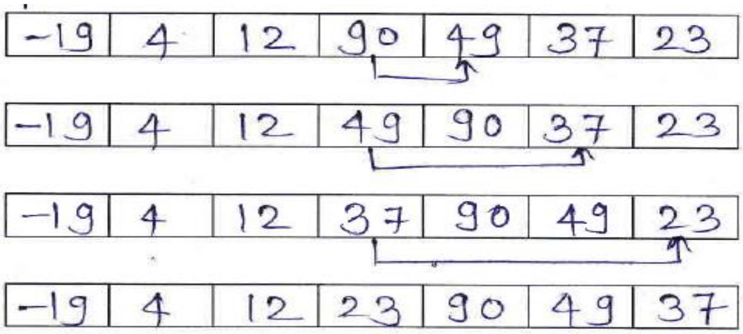
Correct steps: each pass-1M

Pass 3



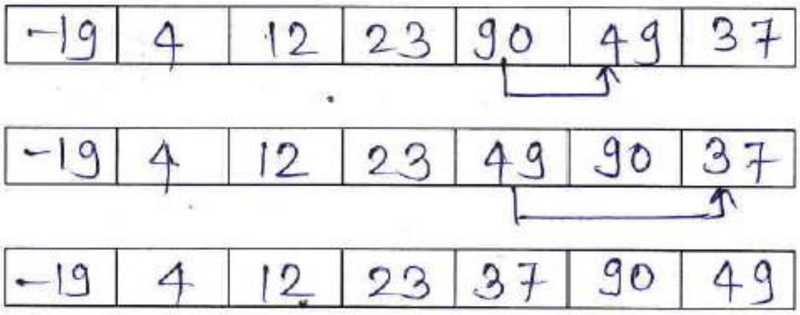
-19	4	37	90	49	23	12
-19	4	37	90	49	23	12
-19	4	37	90	49	23	12
-19	4	23	90	49	37	12
-19	4	12	90	49	37	23

Pass 4



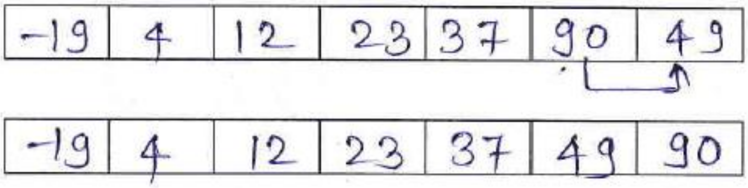
-19	4	12	90	49	37	23
-19	4	12	49	90	37	23
-19	4	12	37	90	49	23
-19	4	12	23	90	49	37

Pass 5



-19	4	12	23	90	49	37
-19	4	12	23	49	90	37
-19	4	12	23	37	90	49

Pass 6



-19	4	12	23	37	90	49
-19	4	12	23	37	49	90

	b	**Explain the concept of recursion using stack.**	**6M**
	Ans	Recursion is a process of calling a function by itself. a recursive function body contains a function call statement that calls itself repetitively. Recursion is an application of stack. When a recursive function calls itself from body, stack is used to store temporary data handled by the function in every iteration.	Explanation-4M & 2M for Example

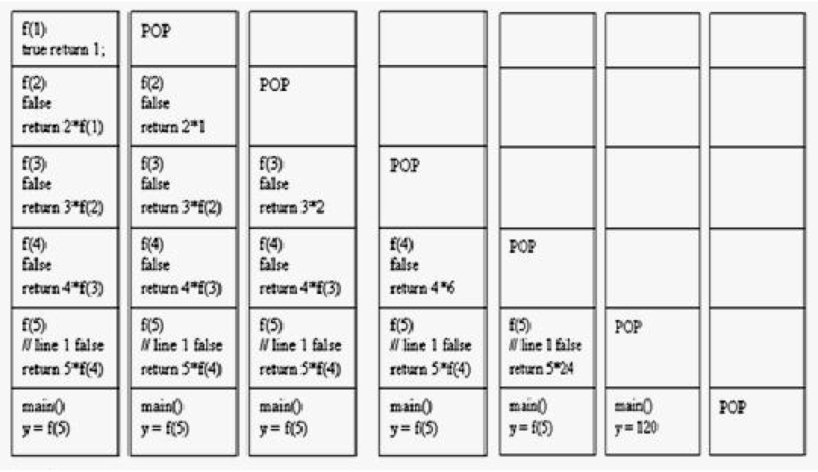
Example:

function call from main() : fact(n); // consider n=5

Function definition:

```
int fact(int n)
{
if(n==1)
return 1;
else
return(n*fact(n-1));
}
```

In the above recursive function a function call fact (n-1) makes a recursive call to fact function. Each time when a function makes a call to itself, it save its current status in stack and then executes next function call. When fact ( ) function is called from main function, it initializes n with 5. Return statement inside function body executes a recursive function call. In this call, first value of n is stored using push ( ) operation in stack (n=5) and a function is called again with value 4(n-1). In each call, value of n is push into the stack and then it is reduce by 1 to send it as argument to recursive call. When a function is called with n=1, recursive process stops. At the end all values from stack are retrieved one by one using pop ( ) operation to perform multiplication to calculate factorial of number.



f(1)
true return 1;
	POP					

f(2)
false
return 2*f(1)
	f(2)
	false
	return 2*1
		POP				

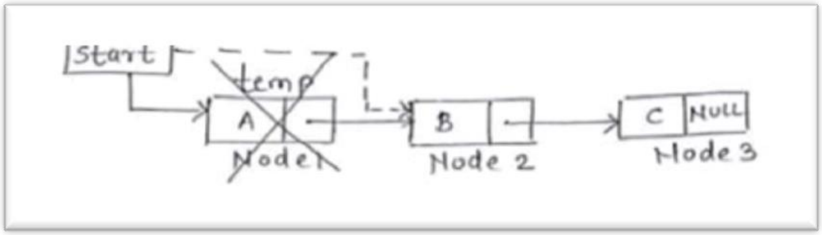
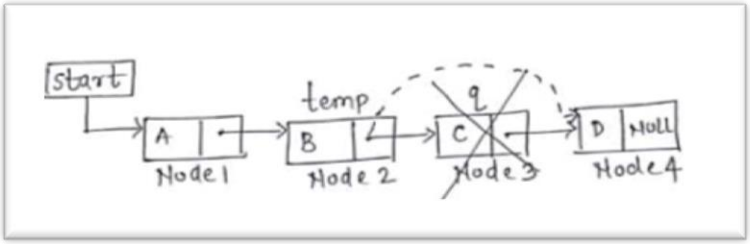
f(3)
false
return 3*f(2)
	f(3)
	false
	return 3*f(2)
		f(3)
		false
		return 3*2
			POP			

f(4)
false
return 4*f(3)
	f(4)
	false
	return 4*f(3)
		f(4)
		false
		return 4*f(3)
			f(4)
			false
			return 4*6
				POP		

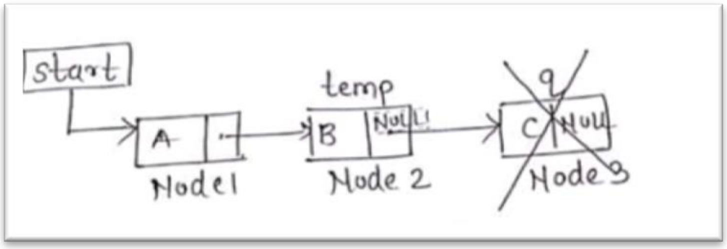
f(5)
// line 1 false
return 5*f(4)
	f(5)
	// line 1 false
	return 5*f(4)
		f(5)
		// line 1 false
		return 5*f(4)
			f(5)
			// line 1 false
			return 5*f(4)
				f(5)
				// line 1 false
				return 5*24
					POP	

main() y = f(5)	main() y = f(5)	main() y = f(5)	main() y = f(5)	main() y = f(5)	main() y = 120	POP

In the above diagram, first column shows result of push operation after each

		recursive call execution. Next columns shows result of pop operation for calculating factorial.	
	c	**Show with suitable diagrams how to delete a node from singly linked list at the beginning, in between and at the end of the list.**	**6M**
	Ans	In a linear linked list, a node can be deleted from the beginning of list, **from in between positions and from end of the list.** **Delete a node from the beginning:-**  Start temp A Node1 B Node 2 C Null Node3 Node to be deleted is node1.Create a temporary node as ‘temp’. Set ‘temp’ node with the address of first node. Store address of node 2 in header pointer ‘start’ and then delete ‘temp’ pointer with free function. Deleting temp pointer deletes the first node from the list. **Delete a node from in between position:-**  Start temp q A Node1 B Node 2 C Node3 D Null Node4 Node to be deleted is node3.Create a temporary node as ‘temp’ and ‘q’. Set ‘temp’ node with the address of first node. Traverse the list up to the previous node of node 3 and mark the next node (node3) as ‘q’. Store address from node ‘q’ into address field of ‘temp’ node. Then delete ‘q’ pointer with free function. Deleting ‘q’ pointer deletes the node 3 from the list.	Diagram for beginning-2M, end-2M, inbetween-2M

Delete a node from the end:-



Node to be deleted is node 3.Create a temporary node as ‘temp’ and ‘q’. Set ‘temp’ node with the address of first node. Traverse the list up to the second last node and mark the last node as ‘q’. Store NULL value in address field of ‘temp’ node and then delete ‘q’ pointer with free function. Deleting q pointer deletes the last node from the list.