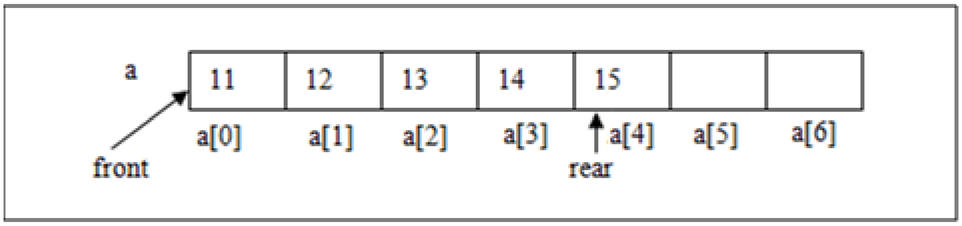
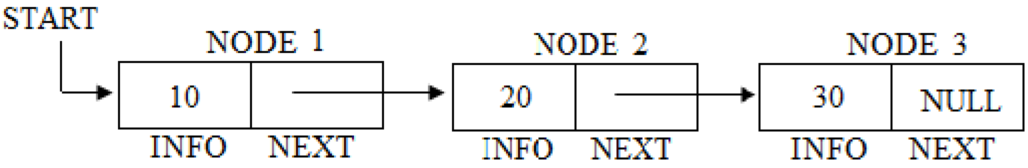
MODEL ANSWER

SUMMER– 17 EXAMINATION

Important Instructions to examiners:

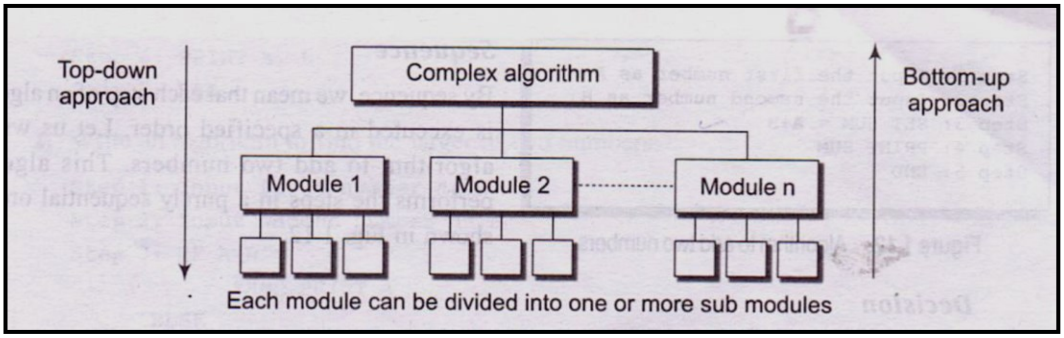
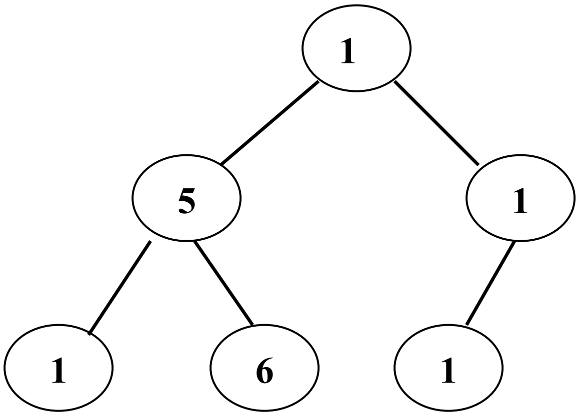
1) The answers should be examined by key words and not as word-to-word as given in themodel answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may tryto assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling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

Q. No	Sub Q. N.	Answer	Marking Scheme
1.	**A)**	**Attempt any six:**	**12Marks**
	a)	**Define complexity and classify it.**	**2M**
	Ans:	**Complexity:** The complexity of an algorithm is a measure that describes its efficiency in terms of amount of time and space required for an algorithm to process. **Classification**: 1. Time complexity **2.** Space complexity	**(Definition: 1mark, Classification: 1mark)**
	b)	**State limitations of the Big 'O'notation.**	**2M**
	Ans:	1. Many algorithms are too hard to analyze mathematically. 2. There may not be sufficient information to calculate the behavior of the algorithm in the average case. **3.** Big-Oh analysis only tells us how the algorithm grows with the size of the problem, not how efficient it is, as it does not consider programming effort.	**(Any two points: 1 Mark each)**
	c)	**Define searching and enlist its types.**	**2M**
	Ans:	**Searching**: It is the process of finding a data element in the given data structure. **Types:** 1. Linear search 2. Binary search	**(Definition:1 mark, types:1/2 mark each)**

<table>
<tr><td></td><td>d)</td><td>Stack is linear data structure. Yes/No? Justify your answer.</td><td>2M</td></tr>
<tr><td></td><td>Ans:</td><td>Yes. Stack is a Linear data structure.
Justification:
Linear data structures allow organization of data elements in proper sequential manner. Stack is linear data structure because it organizes elements in a particular sequence. Stack allows insertion and deletion of element only from one end, so elements are inserted or deleted in a particular sequence.</td><td>(Yes:1 mark, Justification: 1 mark)</td></tr>
<tr><td></td><td>e)</td><td>Sketch representation of queue as an array.</td><td>2M</td></tr>
<tr><td></td><td>Ans:</td><td> </td><td>(Proper sketch representation: 2 marks)</td></tr>
<tr><td></td><td>f)</td><td>Define linked list with example.</td><td>2M</td></tr>
<tr><td></td><td>Ans:</td><td>Linked list: It is linear collection of data elements. Each element in linked list is called as ‘node’. Each node contains two fields. First is INFO which stores data & second is NEXT which is linked with address of next node in a list.
Example:
 </td><td>(Definition: 1 mark, Example: 1 mark)</td></tr>
<tr><td></td><td>g)</td><td>Differentiate between tree and graph (Min. 2 points).</td><td>2M</td></tr>
<tr><td></td><td>Ans:</td><td><table><tr><th></th><th>Tree</th><th>Graph</th></tr><tr><td>Definition</td><td>Tree is defined as a non-empty finite set T of elements, called nodes where nodes contains a root node and others as child nodes.</td><td>A graph is defined as a set of two tuples such that G=(V,E) where G is a graph, V represents vertices and E represents the set of edges of graph G.</td></tr></table></td><td>(Any Two Points: 1 mark Each)</td></tr>
</table>

Root Node	In tree there is exactly one root node and every child have only one parent.	In graph there is no such concept of root node.
Parent child relationship	In trees, there is parent child relationship so flow can be there with direction top to bottom or vice versa.	In Graph there is no such parent child relationship.
Different Types	Different types of trees are: Binary Tree, Binary Search Tree, AVL tree, Heaps.	There are mainly two types of Graphs: Directed and Undirected graphs.
Applications	Tree applications: sorting and searching like Tree Traversal & Binary Search.	Graph applications : Coloring of maps, in OR (PERT & CPM), algorithms, Graph coloring, job scheduling, etc.
Model	Tree is a hierarchical model.	Graph is a network model.

	h)	**What is indegree and outdegree of a node in graph?**	**2M**
	Ans:	**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. **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.	**(Indegree : 1 mark, Outdegree: 1 mark)**
	B)	**Attempt any two :**	**8 Marks**
	a)	**Describe different approaches to design an algorithm.**	**4M**
	Ans:	**1. Top-down approach:** a. A top-down design approach starts by dividing complex algorithm into one or more modules or subsystems b. Each subsystem is then refined in yet greater detail, sometimes in many additional sub system levels, until the entire specification is reduced to base elements. c. Top-down design method is a form stepwise refinement where we begin with the Top most module and incrementally add modules that it calls. **2. Bottom-up approach:** a. In this approach the individual base elements of the system are first specified in detail. b. These elements are then linked together to form larger subsystems, which then in turn are clubbed in many levels, until a complete top-level system is formed.	**(Explanation of Top-down:2 marks, Bottom-up:2 marks)**

		Example:  	
	b)	**Explain Binary Search Tree (BST) with example.**	**4M**
	Ans:	A binary tree is said to be binary search tree when all nodes less than the root node are placed at the left of root node and all nodes greater than or equal to the root node are placed at the right of root node.The nodes in the binary search tree are ordered. The time needed to search an element from the tree is reduced. Binary search tree also speed up the insertion and deletion operation. **Example:**  In the above example, binary tree is a binary search tree. Its root node is 10. Nodes 5,1 and 6 are less than root node so they are placed at left and nodes 19,17 are greater than root node so they are placed at right of node 10.In each level, root node is compared with its child nodes before placing child nodes in a binary search tree.	**(Binary search Tree Concept: 2 marks, Example: 2 marks)**
	c)	**What are the applications of graph? Explain any two with example.**	**4M**
	Ans:	**Applications of graph:** 1. To represent road map 2. To represent circuit or networks 3. To represent program flow analysis 4. To represent transport network	**(Enlisting application:2 marks, Any two relevant Examples :1 mark each)**

5. To represent social network
6. Neural networks

1. Social Network Graphs: to tweet or not to tweet. Graphs that represent who knows whom, who communicates with whom, who influences whom or other relationships in social structures. An example is the twitter graph of who follows whom. These can be used to determine how information flows, how topics become hot, how communities develop, or even who might be a good match for who, or is that whom.

2. Transportation networks: In road networks vertices are intersections and edges are the road segments between them, and for public transportation networks vertices are stops and edges are the links between them. Such networks are used by many map programs such as Google maps, Bing maps and now Apple IOS 6 maps (well perhaps without the public transport) to find the best routes between locations. They are also used for studying traffic patterns, traffic light timings, and many aspects of transportation.

3. Neural networks: Vertices represent neurons and edges the synapses between them. Neural networks are used to understand how our brain works and how connections change when we learn. The human brain has about 10^{11} neurons and close to 10^{15} synapses.

4. Utility graphs: The power grid, the Internet, and the water network are all examples of graphs where vertices represent connection points, and edges the wires or pipes between them. Analyzing properties of these graphs is very important in understanding the reliability of such utilities under failure or attack, or in minimizing the costs to build infrastructure that matches required demands.

5. Network packet traffic graphs: Vertices are IP (Internet protocol) addresses and edges are the packets that flow between them. Such graphs are used for analyzing network security, studying the spread of worms, and tracking criminal or non-criminal activity.

6. Robot planning: Vertices represent states the robot can be in and the edges the possible transitions between the states. This requires approximating continuous motion as a sequence of discrete steps. Such graph plans are used, for example, in planning paths for autonomous vehicles.

7. Graphs in compilers: Graphs are used extensively in compilers. They can be used for type inference, for so called data flow analysis, register allocation and many other purposes.

2.		**Attempt any four :**	**16 Marks**
	a)	**Write a program for Binary search.**	**4M**
	Ans:		**(Correct logic-2 marks, syntax :2 marks, any relevant logic shall be considered)**

```
#include <stdio.h>
int a[20], n, item, loc, beg, mid, end, i;
void main()
{
clrscr();
printf("\nEnter size of an array: ");
scanf("%d", &n);
printf("\nEnter elements of an array in sorted form:\n");
for(i=0; i<n; i++)
scanf("%d", &a[i]);
printf("\nEnter ITEM to be searched: ");
```

```
scanf("%d", &item);
beg = 0; end = n-1;
mid = (beg+end)/2;
while ((beg<=end) && (a[mid]!=item))
{
if (item < a[mid])
end = mid-1;
else
beg = mid+1;
mid = (beg+end)/2;
}
if (a[mid] == item)
printf("\n\n ITEM found at location %d & item is: %d", mid+1,item);
else
printf("\n\nITEM doesn't exist");
getch();
}
```

b) **Convert following expression into postfix form with illustration of all steps.** **4M**

A + B ↑ C * (D/E) – F%G

Note: ↑ indicates exponent operator.

Ans: **(Correct Answer: 4 marks)**

Sr No	Symbol Scanned	Stack	Expression
1	A		A
2	+	+	A
3	B	+	AB
4	↑	+↑	AB
5	C	+↑	ABC
6	*	+*	ABC↑
7	(	+*(	ABC↑
8	D	+*(	ABC↑ D
9	/	+*(/	ABC↑ D
10	E	+*(/	ABC↑ D E
11	)	+*	ABC↑ D E/
12	-	-	ABC↑ D E/*+

13	F	-	ABC↑ D E/*+F
14	%	-%	ABC↑ D E/*+F
15	G	-%	ABC↑ D E/*+FG
16			ABC↑DE/*+FG%-

Postfix Expression: ABC↑DE/*+FG%-

c) Differentiate between stack and Queue.(Min. 4 points). **4M**

Ans:

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

(Any Four Points: 1 mark each)

d) Draw representation of linear linked list, circular linked list and doubly linked list. **4M**

Ans:

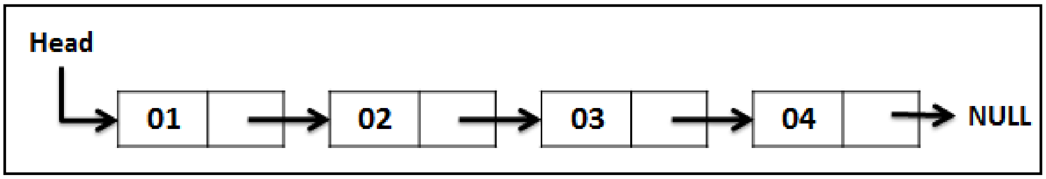


Fig: Linear Linked List

(Representation of linear linked list:1 mark, circular Linked list:1 ½ marks,doubly linked list: :1 ½ marks)

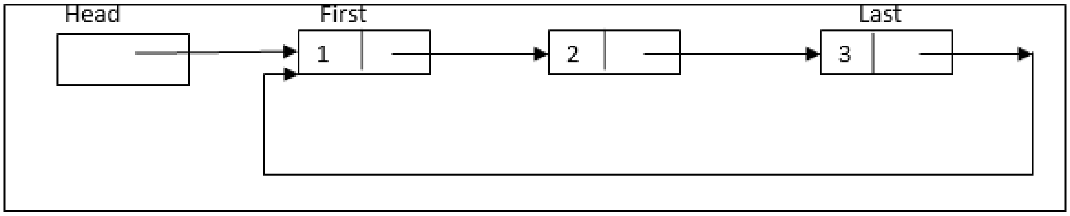


Fig: Circular Linked List

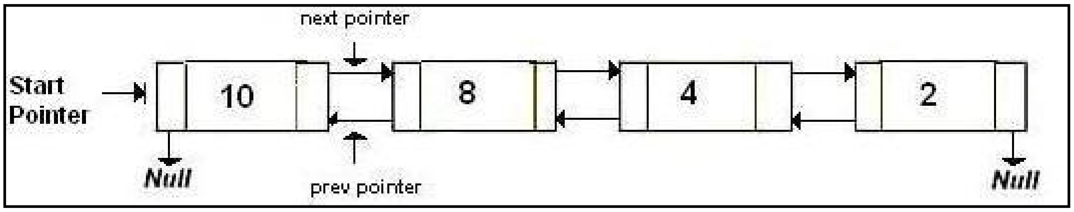
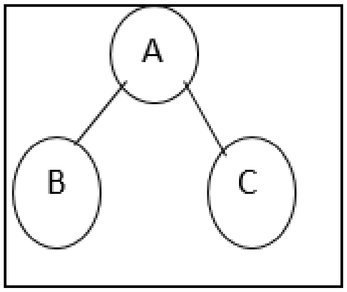
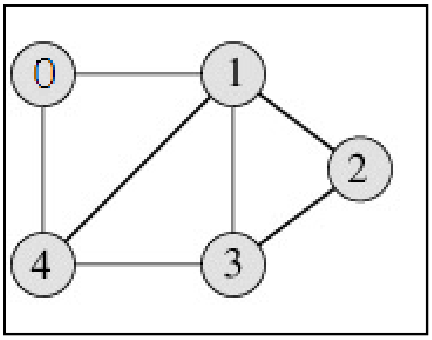


Fig: Doubly Linked List

	e)	**Write algorithm for preorder traversal of binary tree.**	**4M**
	Ans:	**Algorithm for Preorder Traversal:** Step 1: Visit root node. Step 2: Visit left subtree in preorder. Step 3: Visit right subtree in preorder.  **Example:** **Preorder traversal is**: A, B, C. In this traversal method 1st process root element then left subtree & then right subtree.	**(Correct Algorithm: 4 marks)**

f) **Explain representation of graph in detail.** **4M**

Ans: Graph is a data structure that consists of following two components:



(Array representation: 2 marks, linked representation: 2 marks)

Following two are the most commonly used representations of graph:

1.AdjacencyMatrix

2.AdjacencyList

There are other representations also like, Incidence Matrix and Incidence List. The choice of the graph representation is situation specific. It totally depends on the type of operations to be performed and ease of use.

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

The adjacency matrix for the above example graph is:

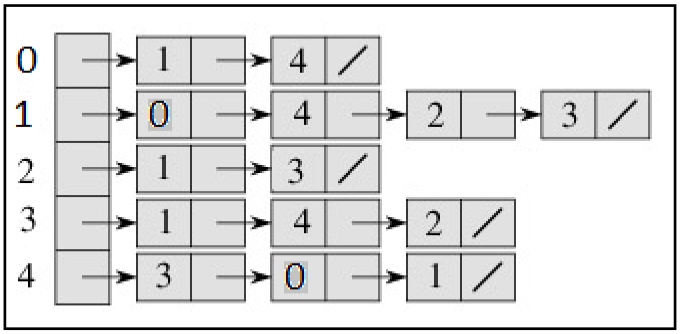
	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

2. Adjacency List: An array of linked lists is used. Size of the array is equal to number of vertices. Let the array be array[]. An entry array[i] represents the linked list of vertices adjacent to the ***i***th vertex. This representation can also be used to represent a weighted graph. The weights of edges can be stored in nodes of linked lists.

Adjacency list contains two columns as vertex name and adjacent vertices. It show who all are adjacent vertices of each vertex in a graph.

Vertex	Adjacent vertices
0	1,4
1	0,2,3,4
2	1,3
3	1,2,4
4	0,1,3

Following is adjacency list representation of the above graph.



Adjacency List Representation of the above Graph

3. **Attempt any four:** **16 Marks**

a) **Describe time and space trade off and time and space complexity with example of each.** **4M**

Ans: **Time and Space trade off:** If we increase the space require to store data then time require for processing will be less and if we decrease the space require to store data then time require for processing will be more. This phenomena is known as time – space trade off.

Time Complexity: Time complexity of program / algorithm is the amount of computer time that is required to run to completion.

Example:

```
Algorithm :  for a=1 to n
             a=a+1
             loop
```

Time complexity of an algorithm with above statement requires n seconds O(n) as the key statement executes n times.

Space Complexity: Space complexity of a program / algorithm is the amount of computer memory that is required to run to completion.

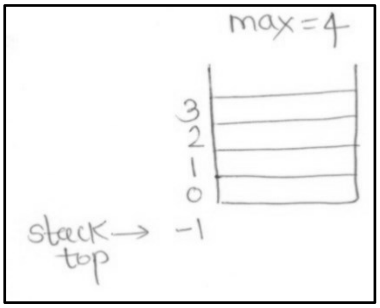
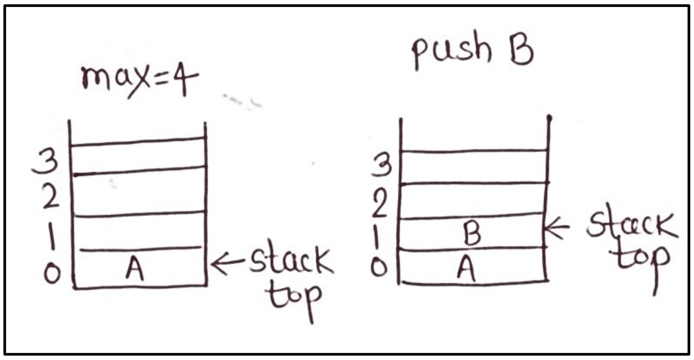
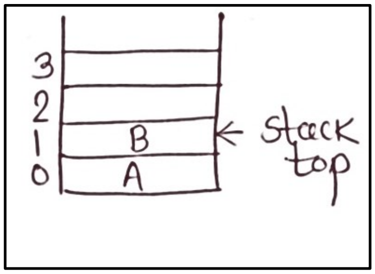
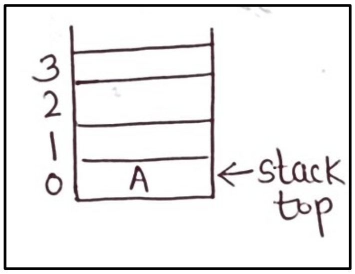
Example: space complexity includes computer space required for storing program instructions, constants, variable, etc.

(Time and space trade off 2 marks; Time complexity 1 mark; Space Complexity 1 mark)

	b)	**Write a program for selection sort.**	**4M**
	Ans:		**(Correct Logic :2 marks,syntax :2 marks)**

```
#include <stdio.h>
void main ()
  {
int array[100],n,i,j,temp,pos;
printf("Enter the number of elements to be sorted: ");
scanf("%d",&n);
printf("enter the elements\n");
   for(i=0;i<n;i++)
     {
scanf("%d",&array[i]);
     }
   for(i=0;i<n;i++)
     {
pos=i;
       for(j=i+1;j<n;j++)
         {
          if(array[j]<array[pos])
pos=j;
         }
       temp=array[i];
       array[i]=array[pos];
       array[pos]=temp;
     }
printf("The Sorted List Is ");
   for(i=0;i<n;i++)
     {
printf("%d ",array[i]);
     }
getch();
  }
```

	c)	**Explain operations on stack using array.**	**4M**
	Ans:	Basic operations of stack are: **1)** Create Stack (): To initialize stack as an empty stack. To show stack is empty, initially stack top is initialize to -1.	**(Any 2 Operations:2 mark each)**

		 **2)** PUSH: The process of adding new element to the top of the stack is called PUSH operation. Push operation increments stack top by one and then adds new elements into the array location indicated by stack top.  **3)** POP: The process of deleting an element from top of the stack is called POP operation. Pop operation decrements stack top by one to delete an element from stack.   **4)** Retrieval & Display: Reading elements from stack and display them. The elements from stack are displayed from 0th location of array upto the stack top position.	
	d)	**Describe priority queue with its advantages.**	**4M**
	Ans:	A priority Queue is a collection of elements where each element is assigned a priority and the order in which elements are added into the queue. The rules for processing the elements of priority queue are: 1) An element with higher priority is processed before any element of lower priority. 2) Two elements with the same priority are processed according to the order in which they are added to the queue (FCFS).	**(Description 2 marks; Any 2 Advantages: 2 marks)**

Advantages:

1) Preferences to the higher priority process are added at the beginning. High priority process executes first.
2) Keep the list sorted in increasing order.

e) **Write an algorithm for searching a node in linked list.** **4M**

Ans: Algorithm SEARCH (INFO, LINK, START, ITEM, LOC)
LIST is a linked list in memory. This algorithm finds the location LOC of the node where ITEM firstappears in LIST or sets LOC =NULL.

```
1) Set PTR := START
2) Repeat Step 3 while PTR !=NULL
3) If ITEM = INFO[PTR], then
   Set LOC: = PTR, and exit;
   Else
           Set PTR: = LINK [PTR]. (PTR now points to the next node)
   [End of if loop]
4) [Search is unsuccessful.] set LOC:= NULL.
5) Exit.
```

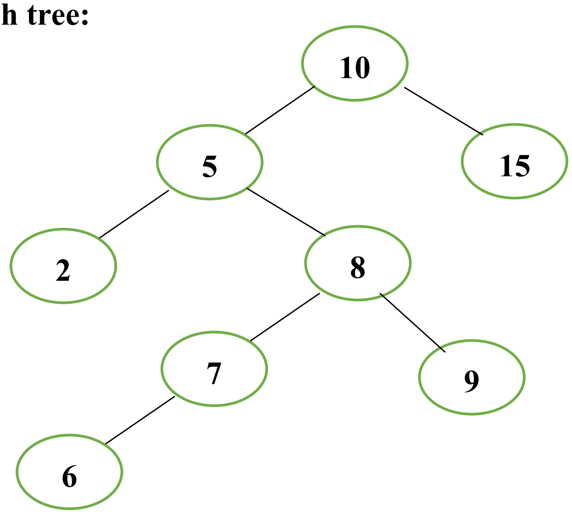
(Correct Algorithm: 4 marks)

(Note: Any set of correct steps shall be considered**)**

f) **Draw a binary search tree for given sequence and write postorder traversal of tree.** **4M**

10 5 8 9 7 6 2 15.

Binary search tree:



Postorder traversal:-

2 6 7 9 8 5 15 10

(Correct binary tree:3 marks, post order traversal:1 mark)

4.		**Attempt any four :**	**16 Marks**
	a)	**Elaborate the steps for performing insertion sort for given elements of array.** **30 10 40 50 20 45**	**4M**
	Ans:	We take given array:	**(Correct Iteration: 4 marks)**

30	10	40	50	20	45

Iteration 1:

Insertion sort compares the first two elements:

It finds that 30 and 10 are not in sorted order; so it will swap 30 with 10; and the resultant sorted sub list will be

10	**30**	40	50	20	45

Iteration 2:

Now it will check for third element; It will first checks 40 with first element that is 10 which is less than 40; then it will check 40 with 30 and concludes that 30 is lesser than 40 so eventually it finds that 40 is already in its desirable location so 40 will be added to sub list.

10	**30**	**40**	50	20	45

Iteration 3:

In Iteration 3 it will check for fourth element which is 50 which will be compare with 10; it finds that 10 is smaller than 50, then it will check next element is sub list, which is 30, this new element is again smaller than 50, so it will consider next element for comparison which is 40. This element is also smaller so it will keep 50 to its place.

10	**30**	**40**	**50**	**20**	45

Iteration 4:

Next it will check for fifth element which is 20 with first element which is 10 so it will jump to next location. At location two it will check with the element 30 with 20 which is greater than 20. So it will shift all elements 30 onwards by one position and inserts 20 at desired location.

10	**30**	**40**	**50**	20	45
	→	→	→	→	
10	**20**	**30**	**40**	**50**	45

Iteration 5:
Now it will check for sixth element which is 45 it is observed as greater than 10 so it will consider next element for comparison which is 20. After comparison with 20 it will compute that 20 is smaller than 45 and hence at its appropriate place. Further it will select next location's element for comparison which is 30, this element is also smaller than 45 hence next element will be selected. After comparing next element which is 40 and smaller than 45 last element will be consider. The last element which is 50 and also greater than 45 will be shifted by one place and will give list in sorted order.

10	**20**	**30**	**40**	50	45
				→	→
10	**20**	**30**	**40**	**45**	50

Sorted List

10	**20**	**30**	**40**	**45**	**50**

b) Recursion is one of the application of stack – YES/NO? Explain it for calculating the factorial of a number 5 using recursion. **4M**

Ans:

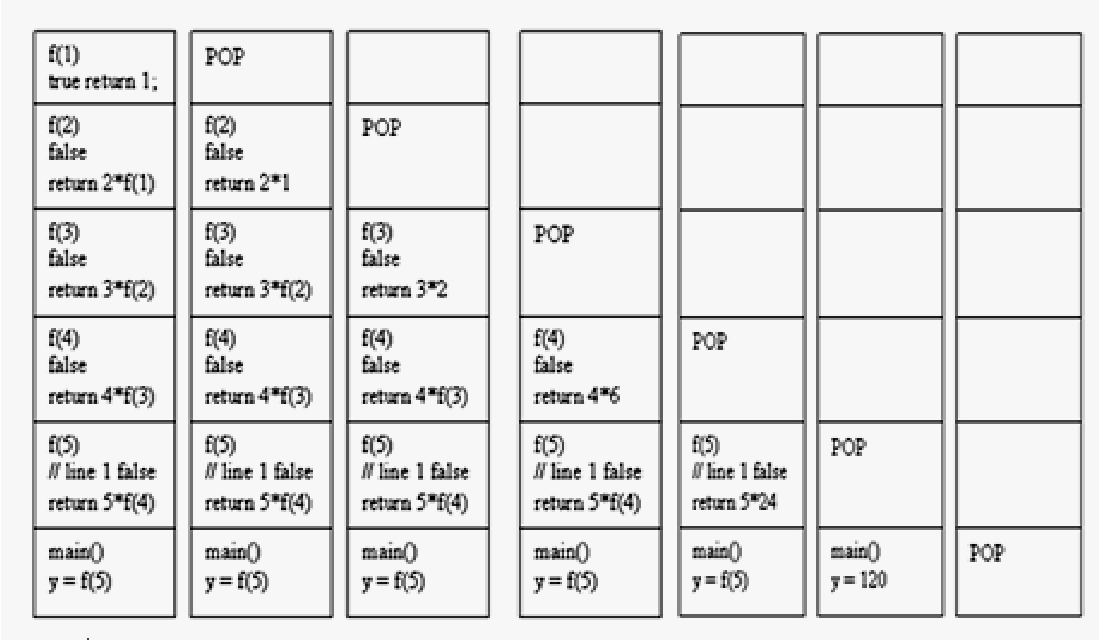
Yes, Recursion is one of the applications of stack.

```
#include<stdio.h>
#include<conio.h>
int fact(int n);
void main()
{
int n=5;
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));
}
```

As shown in above example every time when function fact makes recursive call to itself it has to use stack to save its current status on stack and next function call will be made.

(Recursion application of stack: 1 mark, Description: 3 marks, Program is optional)

In each call value of variable n is stored in stack. Each call execution adds one value in stack. At the end of all recursive calls, all values from stack are retrieved one by one to perform multiplication to calculate. Hence recursion is an application of Stack.



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. Remaining columns shows result of pop operation for calculating factorial. After the last recursive call one by one values are removed from stack till stack becomes empty.

c) **Describe the stack as an abstract datatype.** **4M**

Ans: **(**Note:Any representation of an ADT containing elements and operation shall be considered**)** **(Description: 4 marks)**

An Abstract Data type is one where we can store elements and also perform certain operation on those elements. Stack provides such facilities. Stack as an abstract data type contains stack elements such as array(list), stack top and its operations such as initialize,isempty,isfull,push,pop,display.

Stack elements: array(list), stack top

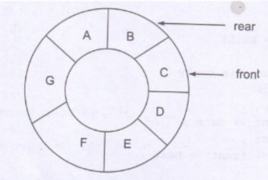
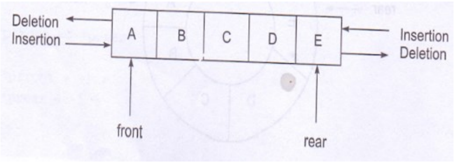
Stack operations:

- Initialize stack to be empty
- Checking whether stack is empty or not
- Checking if stack is full or not
- If stack is not full, then insert a new element. This operation is called as push.
- If stack in not empty, then retrieve element from stack top.
- If stack in not empty, then remove an element from stack top. This operation is called as pop.

d) **Compare circular queue and double-ended queue. (Min. 4 points)** **4M**

Ans:

(Any four points of comparison: 1 mark each)

Circular Queue	Double Ended Queue
Insertion and Removal operation takes place at only one place. i.e. rear and front.	Insertion and removal can take place at both end
It is static in nature.	It can grow dynamically
It works in traditional FIFO manner	It does not require to be in LIFO & FIFO manner; removal can take place at any end.
Insertion and removal is slow as compare to Double Ended Queue	Relative fast insertion and removal operation as compare to Circular Queue.
	

e) **Define :** **4M**

i)Sibling
ii)Depth of tree
iii)Complete binary tree
iv)Degree of tree.

Ans:

(Each Definition:1 mark)

1. **Sibling:** Sibling is defined as any nodes that have the same parent node.
2. **Depth of tree:** Maximum number of levels in a tree is known as depth of a tree.
3. **Complete binary tree:** A binary tree in which every non leaf node has exactly two children and all leaf nodes are at same level is called complete binary tree.
4. **Degree of tree:** Degree of tree is defined as maximum number of child nodes of any node. Or degree of node is the number of nodes connected to a particular node.

f) **Explain Hashing with its significance.** **4M**

Ans:

(Description: 2 marks; Significance: 2 marks)

Hashing is the process of mapping large amount of data item to a smaller table with the help of hashing function. The essence of hashing is to facilitate the next level of searching. Hashing is a process where we use a hashing function that converts range of key values into a range of indexes of an array. The value which is generated by hash function isuse to store a value in hash table.

Significance of Hashing:

- It is efficient to handle vast amount of data items in a given collection.
- No extra space is required to store the index as in the case of other data structures.
- A hash table provides fast data access and rapid updates.
- Due to hashing process, the result is a hash data structure that can store or retrieve data item in an average time disregard to the collection size.

5. **Attempt any two :** **16 Marks**

a) **Write a program for linear search. Find position of element 30 using linear search algorithm in given sequence.** **8M**

10 5 20 25 8 30 40

Ans;

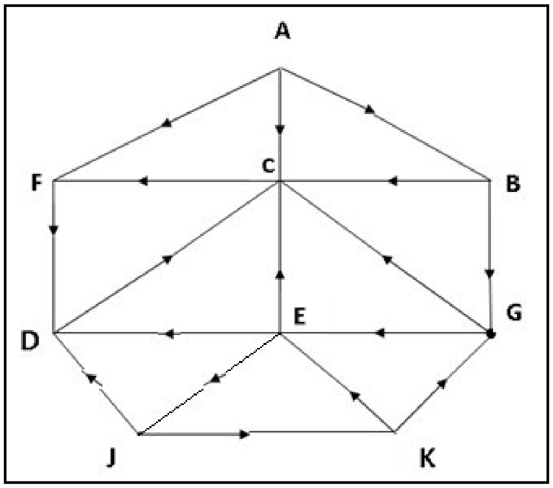
(Correct program:4 marks, (any other logic shall also be considered)

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[20],i,x,n;
clrscr();
printf(“How many elements?”);
scanf(“%d”,&n);
printf(“Enter array elements:n”);
for(i=0;i<n;i++)
scanf(“%d”,&a[i]);
printf(“nEnter element to search:”);
scanf(“%d”,&x);
for(i=0;i<n;i++)
if(a[i]==x)
break;
if(i<n)
printf(“Element found at index %d”,i);
else
printf(“Element not found”);
getch();
}
```

Steps to find position of element 30

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**
array A	10	5	20	25	8	30	40

		Search 30 **step 1:** compare element at position 0 i.e 10 with 30 10==30 NO, increment position by 1 **step 2:** compare Element at position 1 i.e 05 with 30 5==30 NO, increment position by 1 **step 3:** compare Element at position 2 i.e 20 with 30 20==30 NO, increment position by 1 **step 4:** compare Element at position 3 i.e 25 with 30 25==30 NO, increment position by 1 **step 5:**compare Element at position 4 i.e 8 with 30 8==30 NO, increment position by 1 **step 6:** compare Element at position 5 i.e 30 with 30 30==30 YES , SEARCH IS SUCCESSFUL AND ELEMENT IS PRESENT AT POSITION 5	**(correct steps to find Element: 4 marks)**
	b)	**Explain any three application of stack in detail with example.**	**8M**
	Ans:	Following aretheapplications ofstack • Depth first search of graph • Expressionconversion&evaluation • Reversinga list • Recursion • Interrupt handling • Stackmachine • Matchingparenthesis in an expression **1. DepthFirstSearch (DFS)stack is used in searching method to store the values of variables during execution of application.** **Example:-**	**(List of application: 2 marks, any three applications with Example: 2 marks Each)**



a) Initially, push **J** onto
   stackas follows:stack: J
b) Popandprintthetopelement**J**,andthenpushontothestackalltheneighborsof**J**asfollo
   ws:
   PrintJ STACK D,K
c) Pop andprint the top element**K**,and then push onto the stackall the
   unvisitedneighbors of**kPrintKSTACKD, E, G**
d) Pop andprint the topelement G, and then push onto the stackall the
   neighbors of**G.PrintGSTACK D, E, C**

Notethatonly**C**ispushedontothestack,sincetheotherneighbor,**E**isnotpushedbecause**E** has alreadybeen pushedonto the stack).

e) Pop andprint the topelement Cand then push onto the stackall the
   neighbors of **CPrintC STACK D,E, F**
f) Pop andprint the
   top
   element**FPrintFS**
   TACK D, E

Notethatonlyneighbor**D**of**F**isnotpushedontothestack,since**D**hasalreadybeenpushedont
o the stack.

g) Pop andprint the topelement E and push onto the stackall the neighbors of**D**Print E **STACK:D,**

h) Pop and print the top element D, and push onto the stack all the neighbors of**D**Print **D STACK : empty**

The stack is now empty, so the **DFS** of **G** starting at **J** is now complete. Accordingly, the nodes which were printed,

J, K,G, C,F, E, D

These are the nodes reachable from **J.**

2. **Polish notation:- stack is used to process polish notations in the form of infix, prefix and postfix notations. Operands and operators are stored inside the stack while conversion and evaluation of mathematical expressions.**

Example: Conversion of infixexpressionintoapostfixexpression

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

SR.NO	STACK	INPUT	OUTPUT
1	Empty	((A+B)*D)^(E-F)	-
2	(	(A+B)*D)^(E-F)	-
3	((	A+B)*D)^(E-F)	-
4	((	+B)*D)^(E-F)	A
5	((+	B)*D)^(E-F)	A
6	((+	)*D)^(E-F)	AB
7	(	*D)^(E-F)	AB+
8	(*	D)^(E-F)	AB+
9	(*	)^(E-F)	AB+D
10	Empty	^(E-F)	AB+D*
11	^	(E-F)	AB+D*
12	^(	E-F)	AB+D*E
13	^(	-F)	AB+D*E
14	^(-	F)	AB+D*E

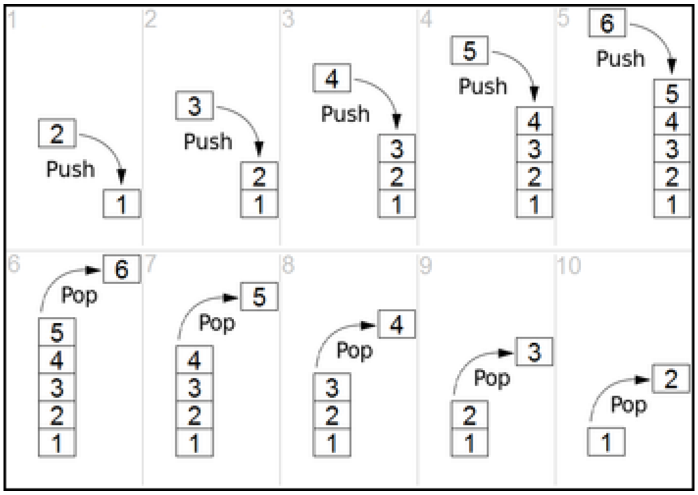
15	^(-	)	AB+D*EF
16	^	End	AB+D*EF-
17	Empty	End	AB+D*EF-^

Postfixexpression=AB+D*EF-^

3. **REVERSAL A LIST:** To reverse a list, the elements of list are pushed onto the stack one by one. Once all elements are pushed on the stack they are popped one by one. Since the element last pushed in comes out first, hence reversal of string occurs.

Example: a list contains elements as {1, 2, 3, 4, 5, 6}. Every push operator will push an element on top ofstack.

Once all elements are pushed one can pop all elements and save it which results in to reversing of list as {6, 5, 4, 3, 2, 1}.



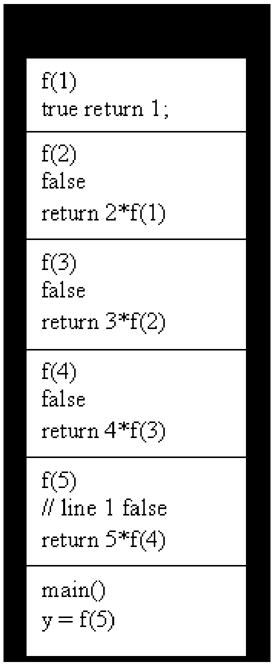
4. **Recursion:A function calls itself repeatedly. In recursion, each recursive call use stack to store the values of the variables during execution of application.**

Example :factorial function using recursion

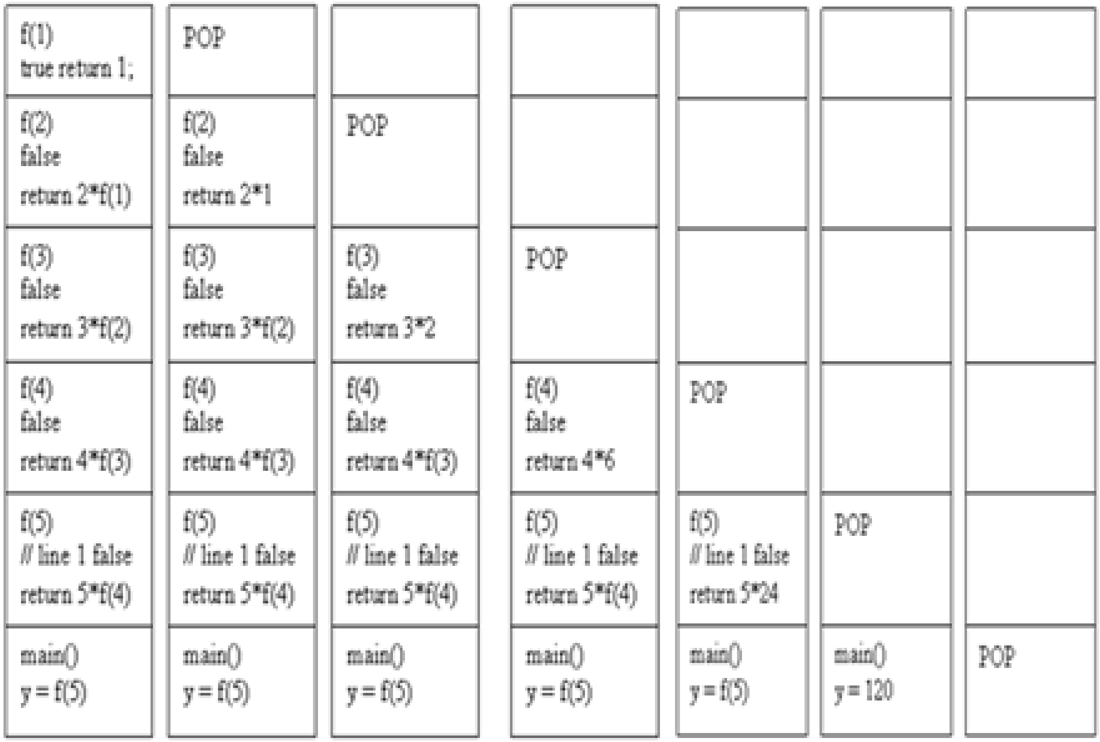
Factorial of 5 is 5x4x3x2x1 = 120 and this can be implemented recursively.

```
int f(int x){

  if(x == 1) return 1;// line 1
```

```
  return f(x-1)*x;    // line 2

}

void main(){

  int y = f(5);// main call

  // y should get 120

}
```



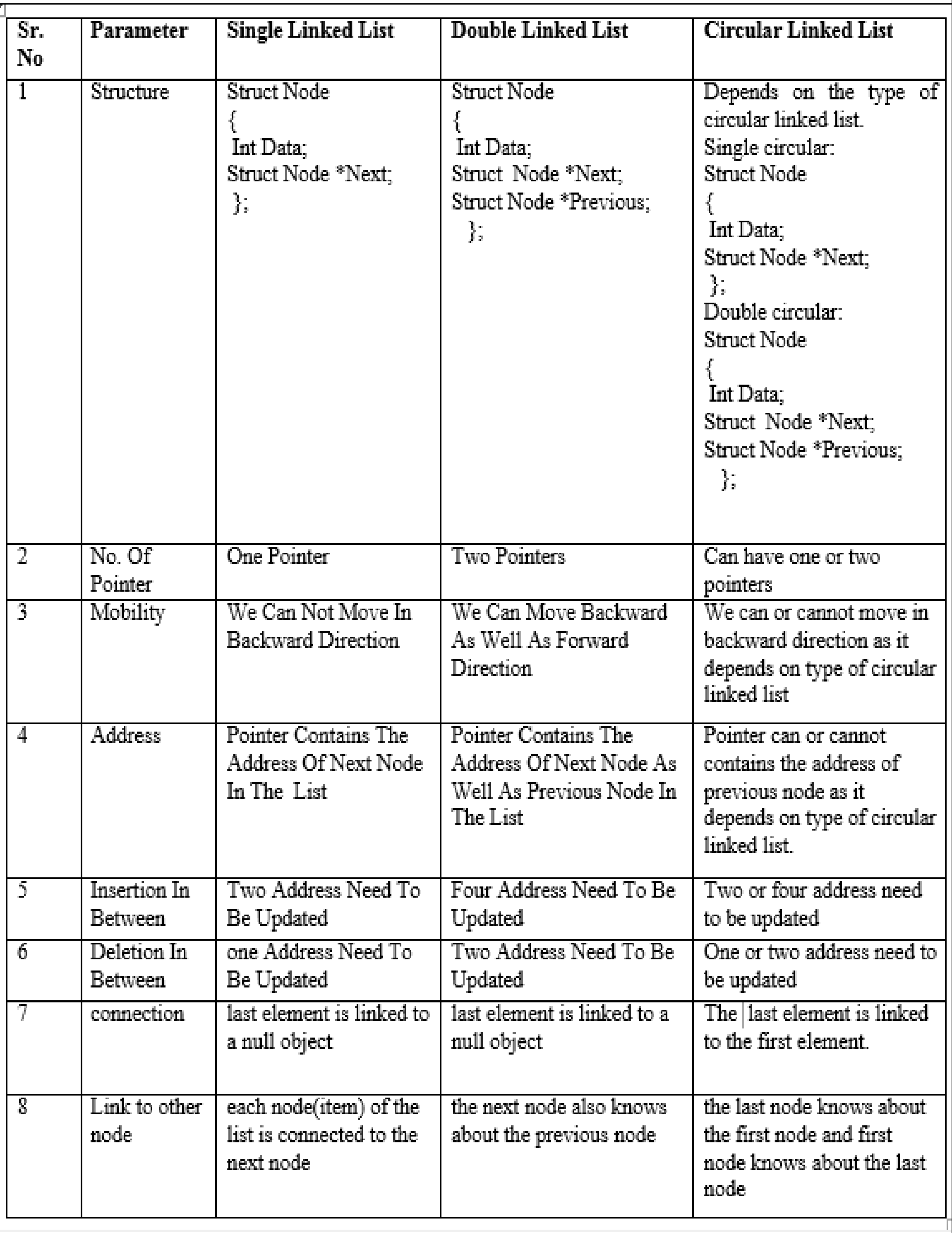
So at this point none of the functions have yet returned! The first to return will be f(1) which will return 1. Then f(2) will return 2. Then f(3) will return 6. As i



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

c) **Differentiate between linear linked list, circular linked list and doubly linked list.(Min. 4 points each).**

Ans:



Sr. No	Parameter	Single Linked List	Double Linked List	Circular Linked List

1	Structure	Struct Node
		{
		Int Data;
		Struct Node *Next;
		};
			Struct Node
			{
			Int Data;
			Struct Node *Next;
			Struct Node *Previous;
			};
				Depends on the type of circular linked list.
				Single circular:
				Struct Node
				{
				Int Data;
				Struct Node *Next;
				};
				Double circular:
				Struct Node
				{
				Int Data;
				Struct Node *Next;
				Struct Node *Previous;
				};

2	No. Of Pointer	One Pointer	Two Pointers	Can have one or two pointers
3	Mobility	We Can Not Move In Backward Direction	We Can Move Backward As Well As Forward Direction	We can or cannot move in backward direction as it depends on type of circular linked list
4	Address	Pointer Contains The Address Of Next Node In The List	Pointer Contains The Address Of Next Node As Well As Previous Node In The List	Pointer can or cannot contains the address of previous node as it depends on type of circular linked list.
5	Insertion In Between	Two Address Need To Be Updated	Four Address Need To Be Updated	Two or four address need to be updated
6	Deletion In Between	one Address Need To Be Updated	Two Address Need To Be Updated	One or two address need to be updated
7	connection	last element is linked to a null object	last element is linked to a null object	The last element is linked to the first element.
8	Link to other node	each node(item) of the list is connected to the next node	the next node also knows about the previous node	the last node knows about the first node and first node knows about the last node

(4 points: 8 marks)

(Note: description of each stating difference shall also be considered**)**

6.		**Attempt any two :**	**16 Marks**
	a)	**Write a program for insert and delete operation perform on queue. State any two application of queue.**	**8M**
	Ans:	**Note: Any other correct logic shall be considered**	**(Program: 6 marks, any two applications : 2 marks)**

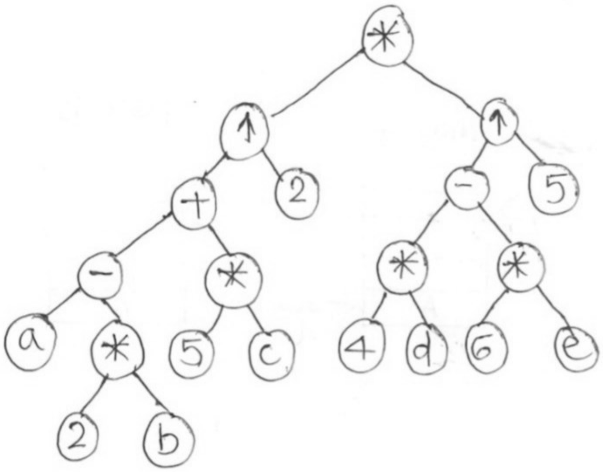
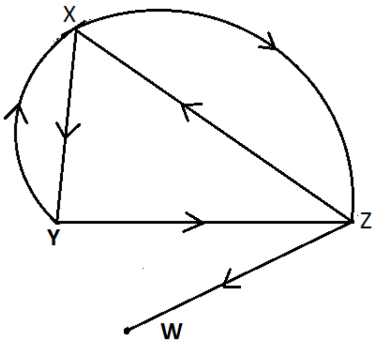
```
#include<stdio.h>
#include<conio.h>
#define max 3
int rear=-1;
int front=-1;
intqueue_arr[max];
void insert();
void del();
void display();
void insert()
{
intinsert_item;
if(rear==(max-1))
printf("\n queue is full");
else
{
printf("\n enter element to be inserted:");
scanf("%d",&insert_item);
rear=rear+1;
queue_arr[rear]=insert_item;
if(front==-1)
{
front=0;
}
}
}
void del()
{
if(rear==-1)
printf("\n queue is empty");
else
{
printf("\n delete element %d",queue_arr[front]);
queue_arr[front]=0;
if(front==rear)
{
front=-1;
rear=-1;
}
else
```

```
front=front+1;
}
}
void main()
{
intch;
clrscr();
while(1)
{
printf("\n1.insert()\n2.delete()");
printf("\n enter choice");
scanf("\n%d",&ch);
switch(ch)
{
case 1:
insert();
break;
case 2:
del();
break;
default:
exit(0);
}
}
}
```

Application of queue:

1) Simulation
2) CPU scheduling in multiprogramming and time sharing systems.
3) Queue is in round robin scheduling algorithm
4) Serving requests on a single shared resource, like a printer, CPU task scheduling etc.
5) In real life, Call Center phone systems will use Queues, to hold people calling them in an order, until a service representative is free.
6) Handling of interrupts in real-time systems.

	b)	**Draw tree for given expression and find inorder, preorder and postorder traversal. (a – 2b + 5c)^2 (4d – 6e)^5.**	**8M**

	Ans:	 Inorder := a - 2 * b + 5 * c ↑ 2 * 4 * d - 6 * e ↑ 5 Preorder := * ↑ + - a * 2 b * 5 c 2 ↑ - * 4 d * 6 e 5. Postorder := a 2 b * - 5 c * + 2 ↑ 4 d * 6 e * - 5 ↑ *	**(Tree creation:4 marks, inorder:1 mark, preorder:1 mark,postorder:2 marks)**
	c)	**Consider the graph G in Fig. 1**  **Figure – 1** i)Write Adjacency matrix representation. ii) Depth first taversal sequence. iii) Find all simple path from X to W. iv) Find indegree (X) and outdegree (W).	**8M**

Ans: **i)Write Adjacency matrix representation** **(2 marks)**

$$\begin{array}{c|cccc} & W & X & Y & Z \\ \hline W & 0 & 0 & 0 & 0 \\ X & 0 & 0 & 1 & 1 \\ Y & 0 & 1 & 0 & 1 \\ Z & 1 & 1 & 0 & 0 \end{array}$$

ii) Depth first taversal sequence. **(4 marks)**

Note:Any source can be considered

NODE	**ADACENT NODE**
W	**----**
X	**Y,Z**
Y	**X,Z**
Z	**W,X**

Let consider source as X

Step 1: Push X on stack.POP and print X. PUSH the adjacent of X on stack.
Step 2: POP and print Z.PUSH the adjacent of Z on stack.
Step 3: POP and print W.PUSH the adjacent of W on stack
Step 4: POP and print Y.PUSH the adjacent of Y on stack.

Nodes reachable from node X are X,Z,W,Y.

iii) Find all simple path from X to W. **(1 marks)**

a) X-Z-W

b)X-Y-Z-W

iv) Find indegree (X) and outdegree (W). **(1 marks)**

indegree(X)=2

outdegree(W)=0