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 judgment 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.

Marks

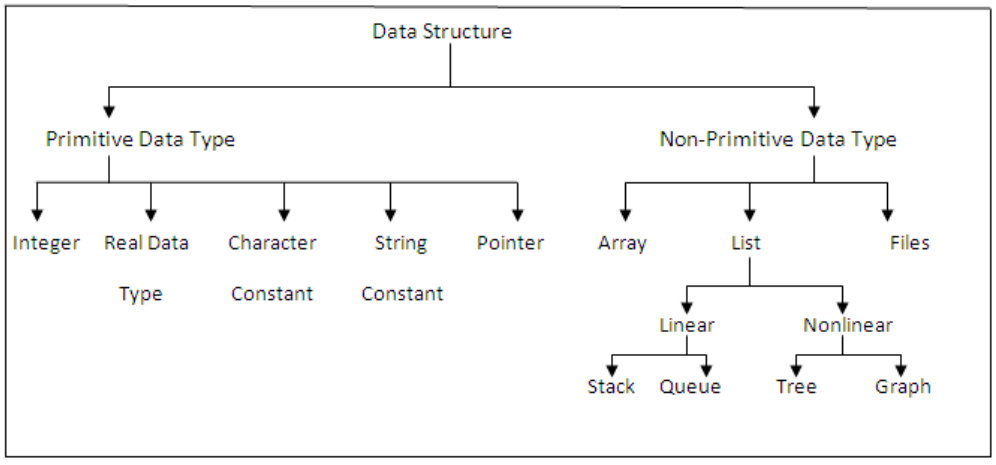
1. a) Attempt any SIX of the following: **12**

(i) Define data structure and give it's classification.

(Definition - 1 Mark, Classification -1 Mark)

Ans: A data structure is a specialized format for organizing and storing data.

Classification of Data Structure



(ii) Enlist various types of operation on data structure.

(Any 4 operations - 2 Marks)

Ans: Operations on Data Structure:-

- Insertion
- Deletion
- Searching
- Sorting
- Traversing
- Merging

(iii) Define sorting. Enlist different sorting technique.

(Definition - 1 Mark, Technique - 1 Mark (any 4 techniques are expected))

Ans: Sorting: Process by which a collection of items are placed into ascending order or in descending order.

Sorting Techniques:

- Bubble Sort
- Insertion Sort
- Selection Sort
- Quick Sort
- Merge Sort
- Radix Sort
- Shell Sort

(iv) State applications of stack.

(Any 2 Applications - 1 Mark each)

Ans: Following are the applications of stack

- Recursion
- Reversing String
- Interrupt handling
- Expression conversion & evaluation: Polish Notation

- Stack machine
- Matching parenthesis in an expression

(v) Explain following tree terminology with the help of diagram.

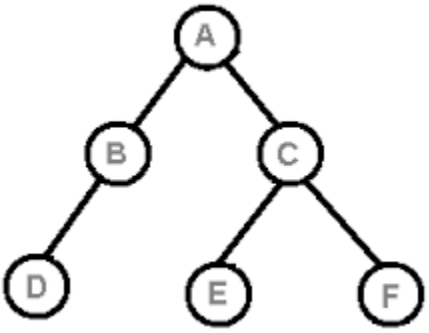
1) Siblings

2) Height of tree

(Siblings - 1 Mark, Height of Tree -1 Mark)

Ans:

1) **Siblings:**Any nodes that have the same parent.
2) **Height of Tree**: The height of a tree is the maximum level of any node in the tree.



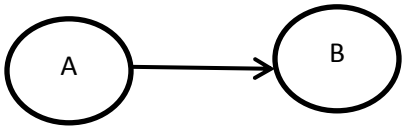
- B and C are siblings; E and F are siblings,
- Height of above tree is 3

(vi) Define directed edge of a tree.

(Definition - 2 Marks)

Ans: Directed edge of a tree is one which has specific direction of flow through which the data can be transferred. Usually directed edges are unidirectional.

Example:



(vii) Differentiate between the Radix sort and Shell sort methods.

(Any 2 points of Comparison - 1 Mark each)

Ans:

Radix Sort	Shell Sort
Sorting is based with integer keys by grouping keys by the individual digits which share the same significant position and value.	Shell sort is an in-place comparison sort. It can be seen as either a generalization of sorting by exchange or sorting by insertion.
Position wise sorting is done	Floor wise sorting is done.
This algorithm is stable in nature	This algorithm is not stable as its iterations are depended on floor value
Its complexity is O(n)	Its complexity is $O(n\log^2 n)$
This is Non-comparison sort as values gets placed in specific position in each iteration	This is comparison sort algorithm as value get compared with all elements within group

(viii) Define hashing.

(Definition - 2 Marks)

Ans: Hashing is a technique used to compute memory address for performing insertion, deletion and searching of an element using hash function.

b) Attempt any <u>TWO</u> of the following: 08

(i) Describe the different approach to design an algorithm.

(Each approach - 2 Marks)

Ans:

1. Top-Down Approach: A top-down approach starts with identifying major components of system or program decomposing them into their lower level components & iterating until desired level of module complexity is achieved . In this we start with topmost module & incrementally add modules that is calls.
2. Bottom-Up Approach: A bottom-up approach starts with designing most basic or primitive component & proceeds to higher level components. Starting from very bottom , operations that provide layer of abstraction are implemented.

(ii) Write a program for sorting the array of 10 elements using the Bubble sort method.

((Any Correct logic shall be considered) for correct logic - 4 Marks)

Ans:

```
#include<stdio.h>
#include<conio.h>
voidbubble_sort(int[]);
void main()
{
intarr[30], num, i;
printf("\nEnter 10 array elements :");
for (i = 0; i < 10; i++)
scanf("%d", &arr[i]);
bubble_sort(arr);
getch();
}
voidbubble_sort(intiarr[])
{
int i, j, k, temp;
printf("\nUnsorted Data:");
for (k = 0; k < 10; k++)
  {
printf("%d", iarr[k]);
   }
for (i = 1; i < 10; i++) {
for (j = 0; j < 10 - 1; j++) {
if (iarr[j] >iarr[j + 1]) {
temp = iarr[j];
iarr[j] = iarr[j + 1];
iarr[j + 1] = temp;
        }
      }
printf("\nAfter pass %d : ", i);
```

```
for (k = 0; k < 10; k++) {
printf("%5d", iarr[k]);
    }
  }
}
```

(iii) Describe the queue as abstract data type.

(Elements - 2 Marks, Operations – 2 Marks)

Ans: Queue is a data structure which allows a programmer to insert an element at one end known as "Rear" and to delete an element at other end known as "Front".

Queue is an abstract data type because it not only allows storing a elements but also allows to perform certain operation on these elements. These operations are as follows.

- Insert
- Delete
- isempty
- isfull

Elements of queue:-

Following are elements of queue

- Front
- Rear
- array

2. Attempt any FOUR of the following: 16

a) Describe Binary search with an example.

(Explanation - 2 Marks, Example - 2 Marks)

Ans: Binary search

Binary search can be performed only on sorted array. In this method first calculate mid position in an array and compare the mid position element with the search element. If a match is found the complete the search otherwise divide the input list into 2 parts. First part contains all the numbers less than mid elements and 2^{nd} part contains all the numbers greater than mid element.

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

The binary search performs the comparison till the element is found or division of list gives one element.

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

Key :15

→ Iteration 1

Lower = 0 Upper = 9

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

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

a[Mid] ! = 15

a[Mid]<KEY; Lower = mid + 1

→ Iteration 2

Lower = 5 Upper = 9

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

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

a[Mid] ! = 25

a[Mid]>KEY; Upper = mid -1

→ Iteration 3

Lower = 5 upper = 6

mid = (Lower + Upper) / 2= (5 + 6) / 2= 5.5

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

a[Mid]! = 15

a[Mid]<KEY; Lower = mid + 1

→ Iteration 4

Lower = 6 upper = 6

mid = (Lower + Upper) / 2= (6 + 6) / 2= 6

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

a[mid] = 15
Number is found

b) Find out infix equivalent of the following postfix expression.

(i) AB + C * D -

(ii) ABC * + D -

(For each correct infix expression - 2 Marks)

Ans:

(i) ((A+B)*C) – D

(ii) A+(B*C)–D

c) Write an algorithm to insert a new node at the beginning of a singly linked list. Give example.

(Algorithm – 2 Marks, Example – 2 Marks)

[Note: Appropriate steps describing procedure for inserting shall be considered.]

Ans: In this operation, new node can be created and added at the beginning of a list. New node points to existing first node and start points to newly added node.

Algorithm:-

Step 1:- Create a new node as temp.

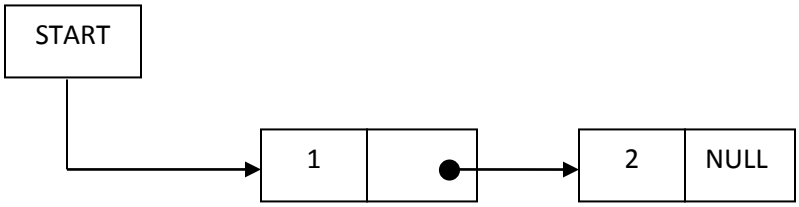
Step 2:- Set temp node's info field.

Step 3:- set temp node's link field with address stored in start node.

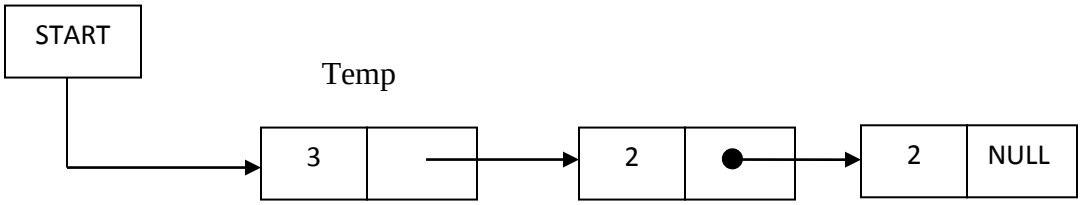
Step 4: set temp node as start node.

Example:-

Before Insertion:-



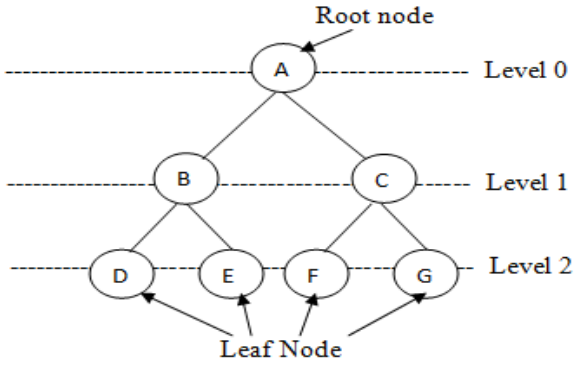
After Insertion:-



d) Define the term related to Binary Tree: Root node, Leaf node, Level and Depth.

(Each -1 Mark)

Ans: Root node



Root Node: A node that has no parent node is known as root node of a tree. In the above example node A is the root node of the tree.

Leaf Node: A leaf node is a terminal node of a tree. It does not have any child. Nodes D, E, F, G are leaf nodes.

Level: A level indicates position of a node in hierarchy of tree structure. Every node has a level number. Root node is placed at level zero. Node D is placed at level 2 & its parent node B is at level 1.

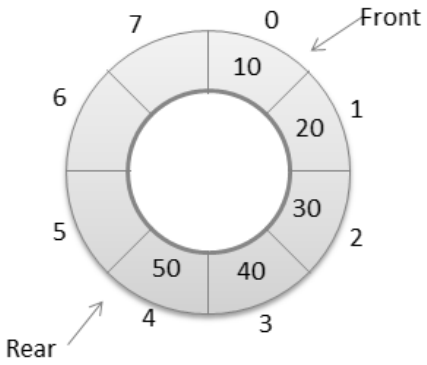
Depth/height: Maximum number of levels in a tree is known as depth of a tree. Depth of the tree in the example above is 3.

e) Define circular queue. Explain insertion and deletion operation on circular queue.

(Circular queue definition – 1 Mark, Diagram - 1 Mark, Explanation of insert operation -1 Mark, delete operation – 1 Mark)

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

Diagram:-



Insert operation: ***(Algorithm or explanation of an algorithm shall be considered)***

Step 1: Check for queue full

If rear=max–1 and front=0 or if front=rear+1 then circular queue is full and insertion operation is not possible.

otherwise go to step 2

Step 2: Check position of rear pointer

If rear=max–1 then set rear=0 otherwise increment rear by 1.

Step 3: Insert element at the position pointer by rear pointer.

Step 4: Check the position of front pointer

If front=–1 then set front as 0.

Delete operation:-***(Algorithm or explanation of an algorithm shall be considered)***

Step 1: Check for queue empty

if front = -1

then circular queue is empty and deletion operation is not possible.

otherwise go to step 2

Step 2: check for position of front and rear pointers.

if front = rear

then

set front=rear=-1

Step 3: check position of front

if front = Max-1

then set front=0;

otherwise

front = front+1

Step 4: Stop

f) Describe merge sort algorithm with an example and state it's time complexity.

(Merge sort Explanation - 1 Mark, Example - 2 Marks, Complexity - 1 Mark)

Ans: Method 1:

In this method two sorted arrays are merged together in a single array. Both the arrays should be in sorted order. First element from both the array is compared & smaller element is placed inside 3rd array, then 1st element from either of the 2 array is compared with 2nd element of another array again smaller element is placed in 3rd array. This process continues till all elements from both the arrays are placed in 3rd array.

Example: array1 array 2

10 1 911 46 20150 72 2

Both arrays should be in sorted order.

→ **Iteration 1:**

(1) 9 10 11 46 **(0)** 2 15 20 72
array 3 : 0

→ **Iteration 2:**

(1) 9 10 11 46 0**(2)**1520 72
array 3 :0 1

→ **Iteration 3:**

1**(9)**10 11 46 0**(2)** 15 20 72

array 3: 01 2

→ **Iteration 4:**

1 **(9)** 10 11 46 0 2 (15) 20 72

array 3: 0 1 2 9

→ **Iteration 5:**

1 9 **(10)** 11 46 0 2 (15) 20 72

array 3: 0 1 2 9 10

→ **Iteration 6:**

1 9 10 **(11)** 46 0 2 (15) 20 72

array 3: 0 1 2 9 10 11

→ **Iteration 7:**

1 9 10 11 (46) 0 2 **(15)** 20 72

array 3: 0 1 2 9 10 11 15

→ **Iteration 8:**

1 9 10 11 (46) 0 2 15 **(20)** 72

array3: 0 1 2 9 10 11 15 20

→ Iteration 9:

1 9 10 11 **(46)** 0 2 15 20 (72)

array 3: 0 1 2 9 10 11 15 20 46

→ Iteration 10:

1 9 10 11 46 0 2 15 20 **(72)**

Sorted Array

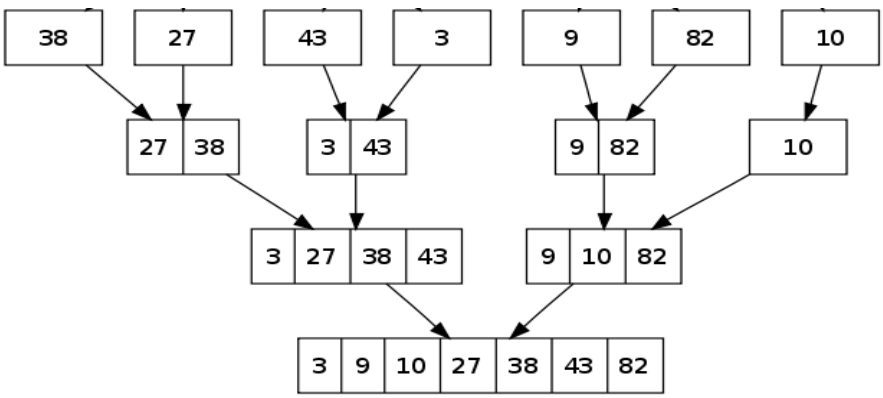
array3: 0 1 2 9 10 11 15 20 46 72

OR

Method 2: Two way merge sort.

All the elements from the inputarray are grouped together sorted in order & merged together. Again groups merged, sort & again same procedure is followed till all elements are placed in one single group.

Example:



Time complexity:-

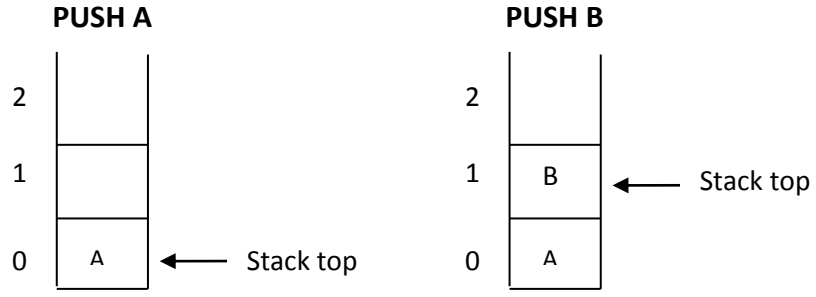
Merge sort	Best-0(n log n)	Average-0(n log n)	Worst-0(n log n)

3. Attempt any FOUR of the following: 16

a) Explain PUSH and POP operation on stack using Array representation.

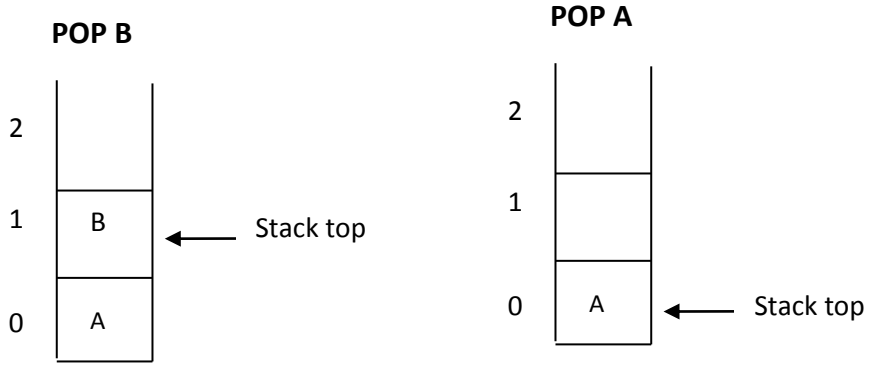
(Push operation with array representation - 2 Marks, pop operation with array representation - 2 Marks)

Ans: PUSH: Push operation is used to insert an element in stack.



Initially stack top is set to -1.i.e stack is empty .When an element is to be inserted first increment the stack top by 1 and then insert new element into it.

POP: Pop operation is used to remove an element from stack.



When pop operation is called stack top is decremented by 1.

b) Describe priority queue with suitable example.

(Description - 2 Marks, Example - 2 Marks)

Ans: A priority queue is a collection of elements where each element is assigned a priority.

The order in which elements are deleted and processed is determined from following rules:

- An element of higher priority is processed before any element of lower priority.
- Two elements with same priority are processed according to the order in which they are added to the queue.

This is a queue in which we are able to insert items and remove items from any position based on priority.

Example

Array representation:

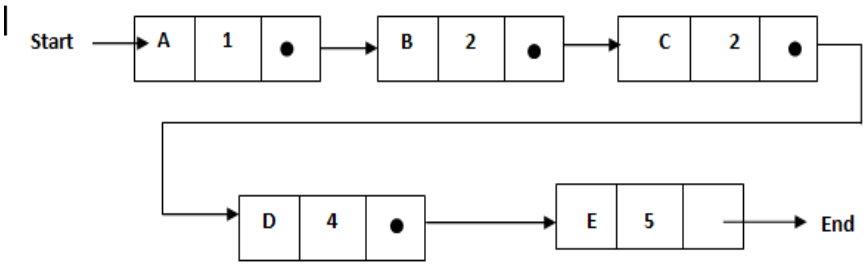
Array element of priority queue has a structure with data, priority and order.

Priority queue with 5 elements:

C,1,4	B,3,2	B,3,5	A,4,1	D,5,3

OR

Linked representation:



c) Write an algorithm for 'search' operation in an unsorted linked list.

(Algorithm - 4 Marks; any other logic shall be considered)

[Note: Appropriate steps describing procedure for searching shall be considered.]

Ans: Algorithm: SEARCH (INFO, NEXT, START, ITEM, LOC)

This algorithm finds ITEM first appears in linked list or sets LOC=NULL

1. Set PTR :=START.
2. Repeat Step 3 while PTR!=NULL
3. If(PTR->INFO == ITEM), then:

PRINT: SEARCH IS SUCCESSFUL and Exit

Else:

Set PTR:=PTR->NEXT[PTR now points to the next node]

[End of If structure]

[End of Step 2 loop]

4. [Search is unsuccessful.] setLOC:=NULL.

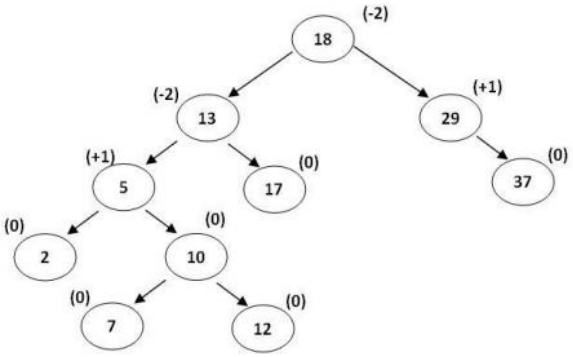
5. Exit.

d) Describe weight balanced tree and height balanced tree along with an example.

(Weight balanced tree with example - 2 Marks, height balanced tree with example - 2 Marks)

Ans: **Weight-balanced tree** (WBT) is a binary search tree, whose balance is based on the sizesof the subtrees in each node.

Example



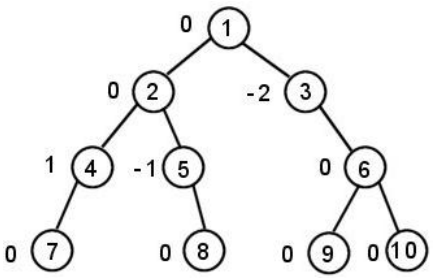
Height balanced trees

AVL trees are binary search trees, which have the balance propriety. The balance property is true forany node and it states: "the height of the left subtree of any node differs from the height of the right subtree by 1".

The binary tree is balanced when all the balancing factors of all the nodes are -1,0,+1.

Formally, we can translate this to this: | hd – hs | ≤ 1, node X being any node in the tree, where hs andhd represent the heights of the left and the right subtrees.

Example



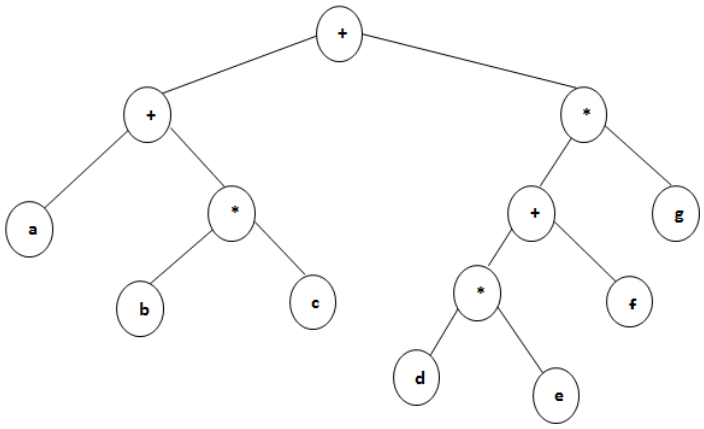
e) Describe expression tree with an example.

(Describe - 2 Marks, Example - 2 Marks)

Ans:

- Expression trees are a special kind of binary tree used to evaluate certain expressions.
- Two common types of expressions that a binary expression tree can represent are algebraic and boolean.
- These trees can represent expressions that contain both unary and binary operators.
- The leaves of a binary expression tree are operands, such as constants or variable names, and the other nodes contain operators.
- Expression tree are used in most compilers.

Example:-



f) Explain the following term with respect to graph using suitable example.

(i) In-degree

(ii) Out-degree

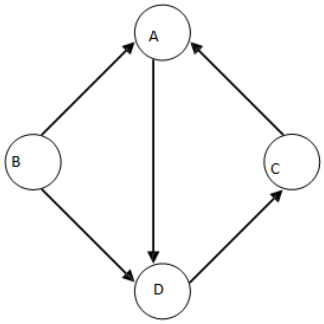
(In-degree with example - 2 Marks, Out-degree with example - 2 Marks)

Ans: The total number of edges linked to the vertex is called as **degree.**

In-degree of a specified vertex in a graph is number of edges coming towards it i.e. number of edges that have that specified vertex as the head.

Out-degree of a specified vertex in a graph is number of edged going out from a specified vertex i.e. number of edges that have that specified vertex as the tail.

EXAMPLE



vertex A has indegree 2 as only two edges are coming towards vertex A. vertex C has out degree 1 as one edge is going out from vertex C

4. Attempt any FOUR of the following: 16

a) Describe algorithm analysis in term of time complexity and space complexity.

(Space complexity - 2 Marks, Time complexity - 2 Marks)

Ans: **Time complexity** of 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 statementswhich are important.

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 2
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 a key statement executes 'n' times as loop runs 'n' times.

Frequency count for algorithm C is n2 as a=a+1 is a key statement executes n2 times as the inner loop runs n times, each time the outer loop runs and the outer loop also runs for n times.

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 followingcomponents.

Fixed space requirements:-Fixed space is not dependent on the characteristics of the input and outputs. Fixed space consists of space for simple variable, fixed size variables, etc.

Variable space requirements:- Variable space includes space needed by variables whose size depends upon the particular problem being solved, referenced variables and the stack space required for recursion on particular instance of variables.

e.g. Additional space required where function uses recursion.

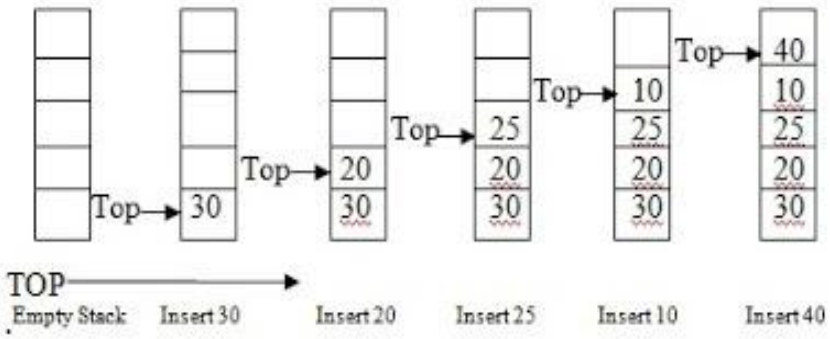
b) Explain the term 'overflow' and 'underflow' with respect to stack use suitable data and diagram.

(Explanation overflow and underflow - 2 Marks, Diagram of each -1 Mark)

Ans: A stack cannot grow indefinitely, because there is always a limit to memory.

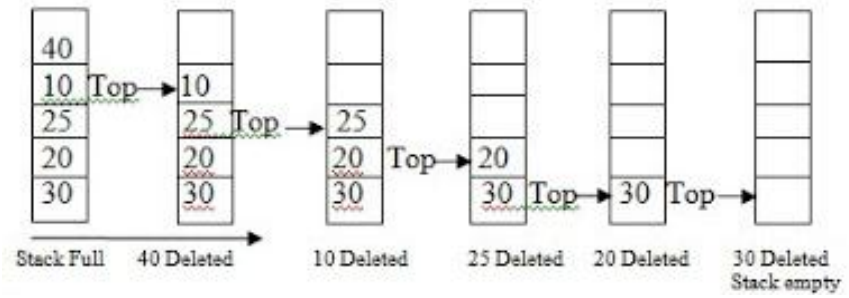
Overflow: Whenthe stack is completely full and then a PUSH operation is performed onto the stack then a '**stack overflow**' occurs.

Example:let us assume that 5 items 30, 20, 25, 10 and 40 are to be placed on the stack. The items can be inserted one by one as shown in following figure.



After inserting 40 the stack is full. In this situation, if push operation is called then stack overflow occurs.Underflow:If the stack is completely empty and a POP operation is called, then '**stack underflow**' occurs.

Example: When an item is to be deleted, it should be deleted from the top as shown in the following figure.



The items deleted in order are 40, 10, 25, 20 and 30. Finally, when all items are deleted then TOP points to bottom of stack. When the stack is empty, if POP operation is called stack underflow occurs.

c) Write a procedure for inserting and element in a queue.

(Procedure- Any other logic shall be considered - 4 Marks)

[Note: Appropriate steps describing procedure for inserting shall be considered.]

Ans: INSERT(QUEUE, F, R, N, item)

[This procedure will insert an element 'item' in QUEUE where 'R' is rear end of queue, 'F' is front end of queue and 'N' is size of array.]

1. [Check for Overflow]

if (R >= N-1)

write queue is full.

return.

2. [Check position of rear pointerin a queue]

If(R==-1)

Set R=0;

F=0;

Else

[Increment R by 1]

R =R + 1

3. [Insert Element]

QUEUE [R]=item.

4. Exit.

d) **Describe doubly linked list with an example.**

(Description - 2 Marks, Example - 2 Marks)

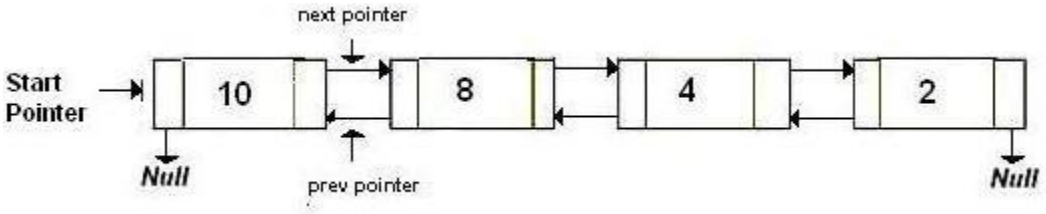
Ans:

- In Doubly-linked list each node contains two links- one to the previous node and other to the next node.
- The previous link of the first node and the next link of the last node points to NULL.
- In Doubly linked list, a node comprises of total three fields
  1) prev, a pointer holding the address of previous node in the list
  2) data, the information part
  3) next, a pointer holding the address of next node in the list

A generic doubly linked list node can be designed as:

```
struct node
{
int data;
struct node* next;
struct node* prev;
};
```

EXAMPLE:



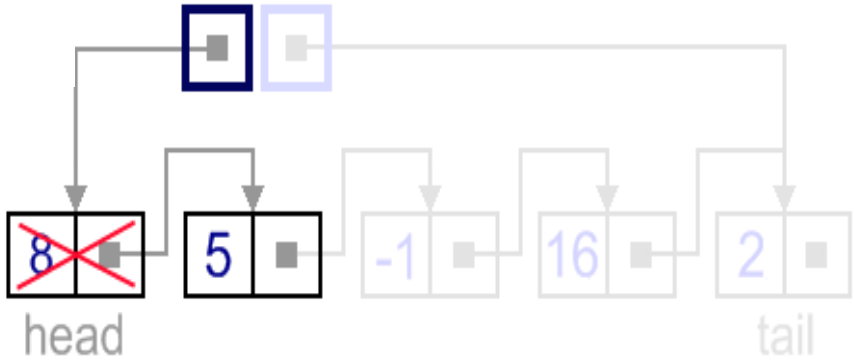
e) Explain how to delete a node in linked list.

(Any one deletion explanation - 4 Marks)

Ans: There are three cases, which can occur while removing the node.

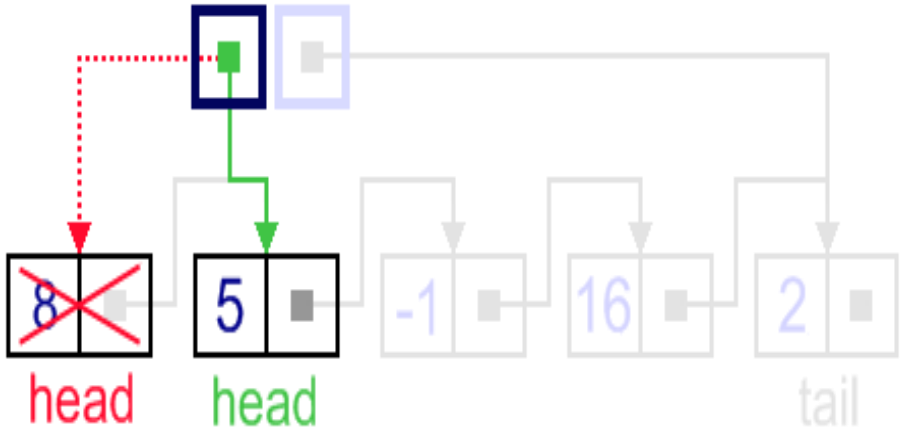
i) Remove first

In this case, first node (current head node) is removed from the list.

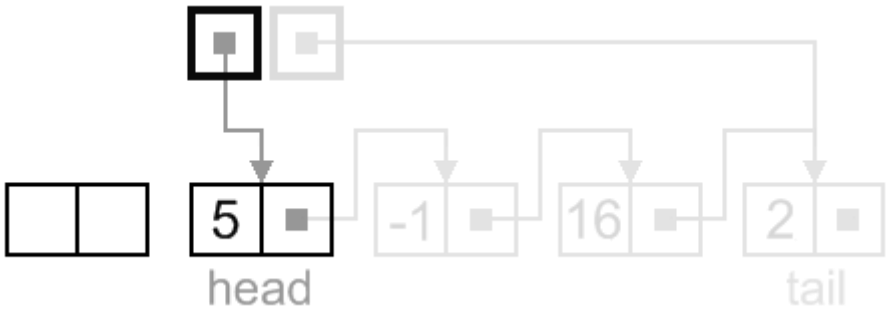


It can be done in two steps:

1. Update head link to point to the node, next to the head.

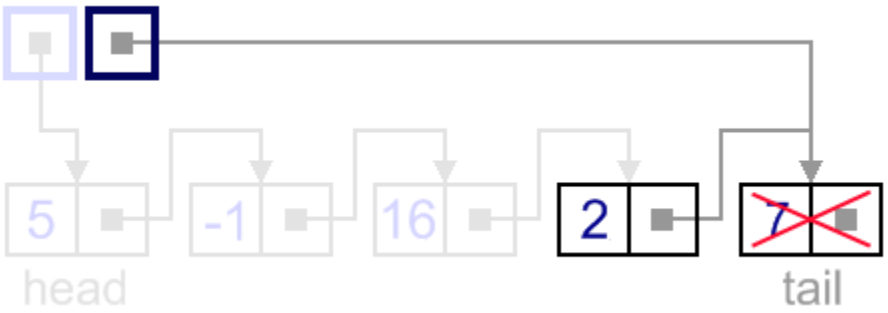


2. Dispose removed node.



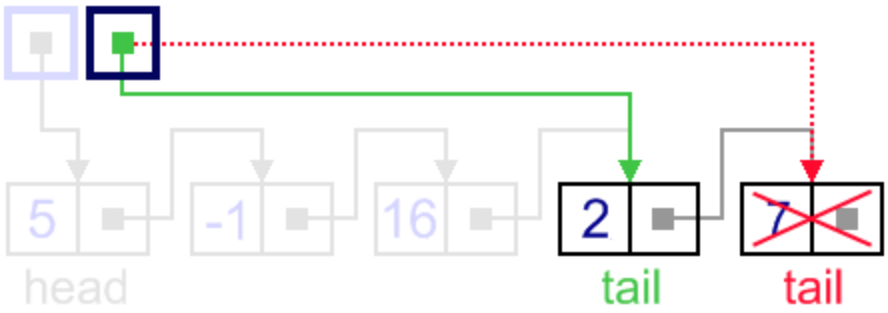
ii) Remove last

In this case, last node (current tail node) is removed from the list. This operation first finds previous node of last node.

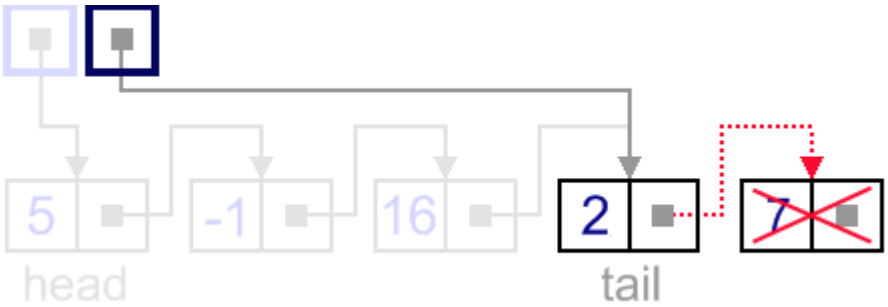


It can be done in three steps:

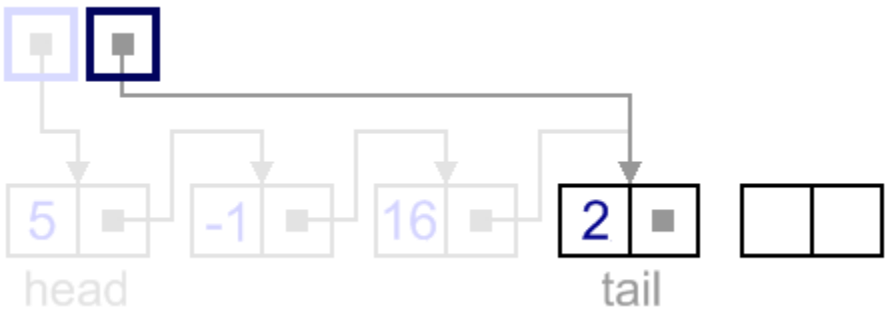
1. Update tail link to point to the node, before the tail. In order to find it, list should be traversed first, beginning from the head.



2. Set next link of the new tail to NULL.

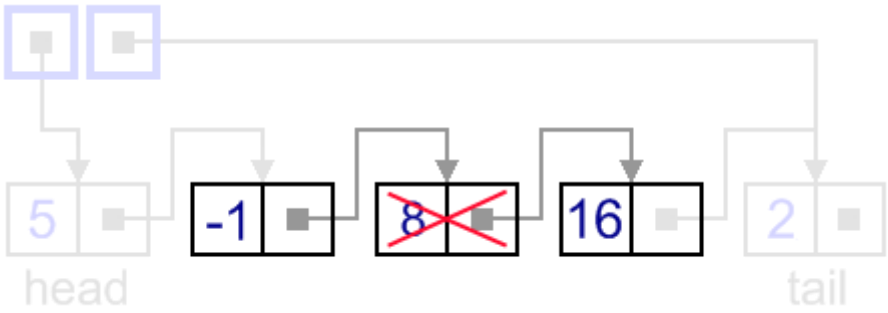


3. Dispose removed node.



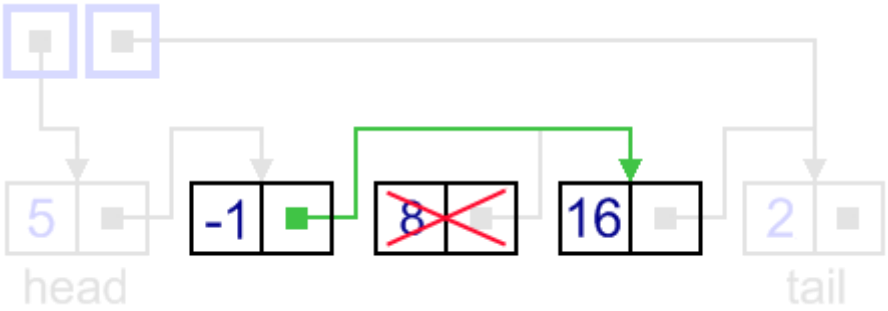
iii) Remove in between

In general case, node to be removed is **always located between** two list nodes. Head and tail links are not updated in this case.

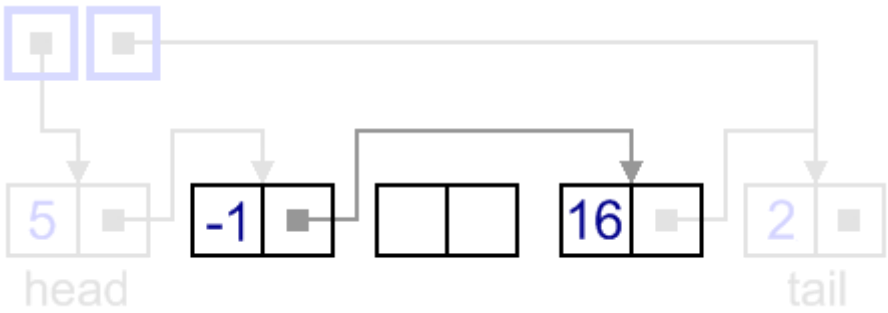


Such a removal can be done in two steps:

1. Update next link of the previous node, to point to the next node, relative to the removed node.



2. Dispose removed node.



f) Perform in-order, post-order and per-order traversal of binary tree, Refer Figure No.1.

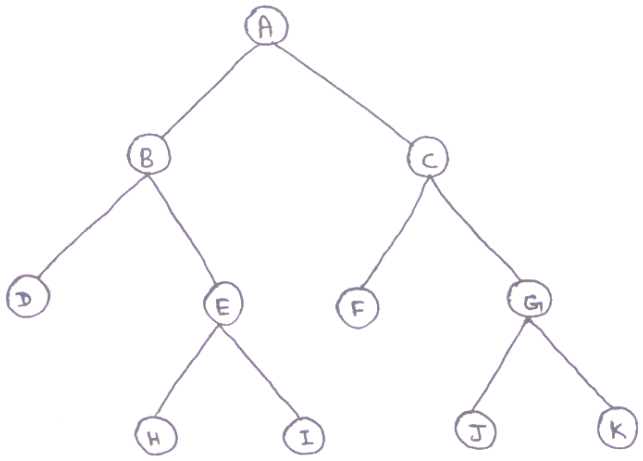


Fig. No. 1

(in-order - 1 Mark, pre-order - 2 Marks, post-order - 1 Mark)

Ans:

in-order:DBHEIAFCJGK

pre-order:ABDEHICFGJK

post-order:DHIEBFJKGCA

5. Attempt any <u>TWO</u> of the following: 16

a) Describe insertion sort algorithm and give steps of insertion sort for sorting the following list in ascending order.

List: 2, 15, 42,26,39,92, 20

also find total number of comparison made.

(Algorithm description – 3Marks, Problem solving – 4 Marks, no of comparison -1 Mark)

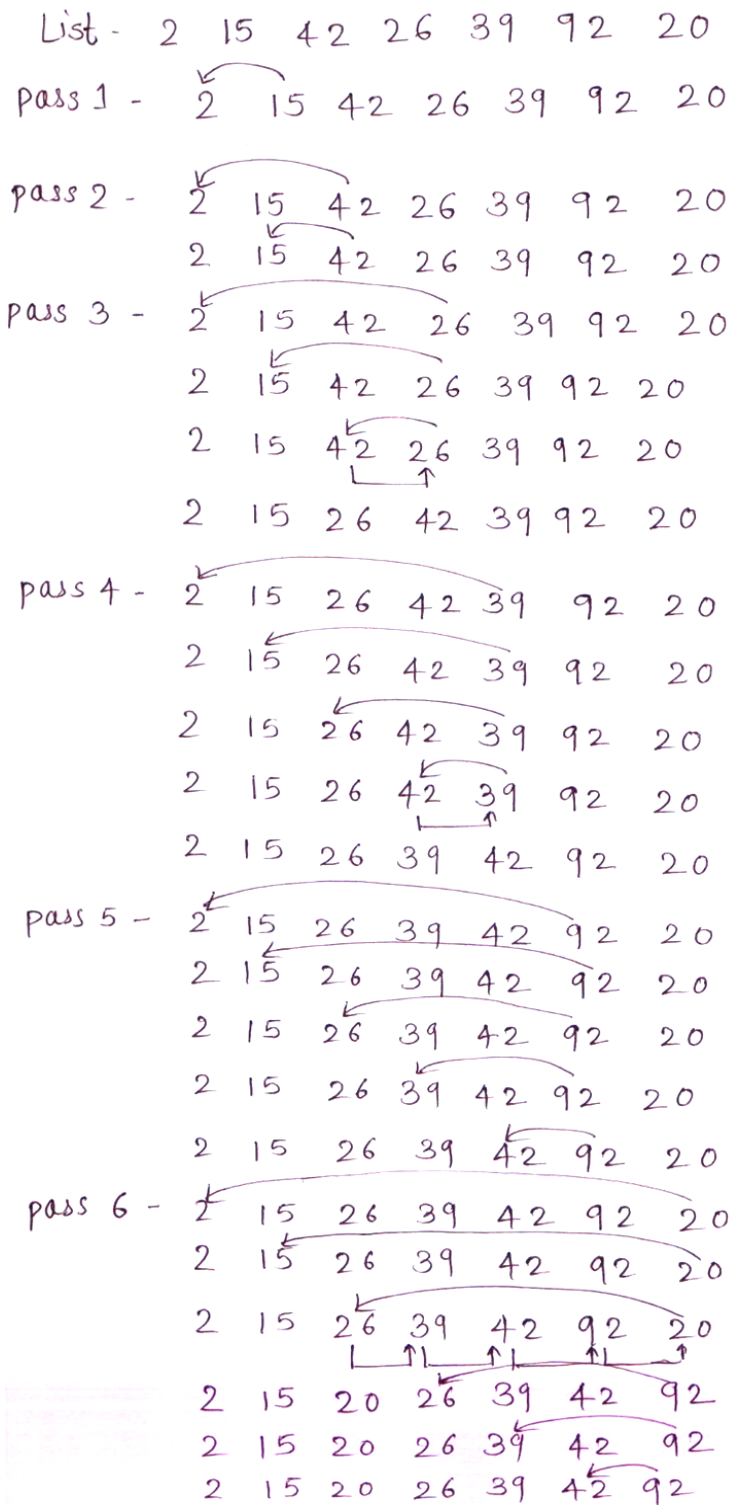
[Note: Appropriate steps describing procedure for sorting shall be considered.]

Ans: Insertion Sort

In insertion sort sorting takes place by inserting a particular element at the appropriate position

Given List:

2,15,42,26,39,92,20



Algorithm

Step1: start

Step2: First iteration 1st element is compared with 0th element if it is smaller it is inserted in the 0th Place of sorted array.

Step3: Compare the remaining elements one by one and place all elements in proper position using shift insert function

Step4: Stop

Total No of comparison is 21

b) Define recursion. Write a 'C' program to calculate the factorial of number using recursion.

(Definition – 2 Marks, Program – 6 Marks)

Ans: A function is called by itself is called as recursion

```
#include<stdio.h>
int fact(int n)
{
        If(n==0)
        return 1;
        else
        return n*fact(n-1);
}
main()
{
        int n;
        printf("Enter the number \n");
        scanf("%d",&n);
        printf("The factorial of %d=%d\n",n,fact(n));
}
```

c) Describe the depth first search traversal of graph.

(Description – 4 Marks, Example – 4 Marks (any valid example shall be considered))

Ans: Depth First Search (DFS)

The aim of DFS algorithm is to traverse the graph in such a way that it tries to go far from the root node.

Stack is used in the implementation of the depth first search.

Back tracking used in this algorithm

Algorithm

Step1: Start

Step2: Initialize all nodes as unvisited

Step3: Push the starting node onto the stack.Mark it as waiting.

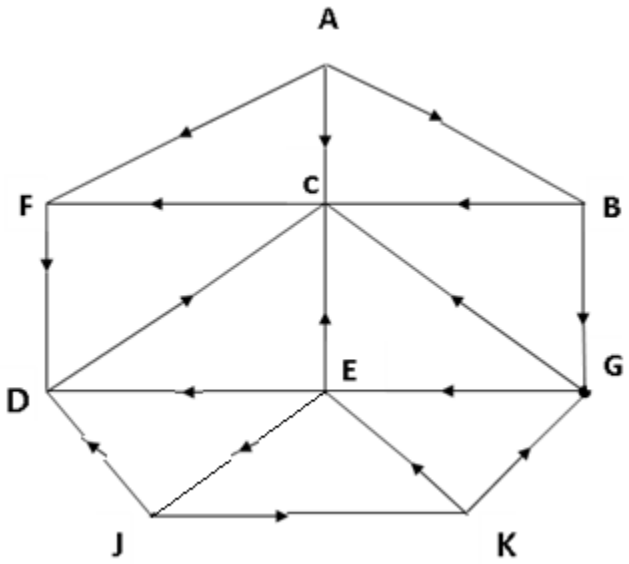
Step4: Pop the top node from stackandmark is as visited. Push all its adjacent nodesinto the stack &mark them as waiting.

Step 5 .Repeat step 4 untilstack is empty.

Step 6: Stop

For example, consider the following graph **G** as follows:

Suppose we want to find and print all the nodes reachable from the node **J** (including **J** itself). The steps for the **DFS** will be as follows:



a) Initially, push **J** onto stack as follows:

stack: J

b) Pop and print the top element **J** , and then push onto the stack all the neighbors of **J** as follows:

Print J STACK D, K

c) Pop and print the top element **K**, and then push onto the stack all the unvisited neighbors of **k**

Print KSTACK D, E, G

d) Pop and print the top element G, and then push onto the stack all the neighbors of **G.**

Print G STACK D, E, C

Note that only **C**is pushed onto the stack, since the other neighbor, **E**is not pushed because **E** has already been pushed onto the stack).

e) Pop and print the top element C and then push onto the stack all the neighbors of **C**

Print C STACK D, E, F

f) Pop and print the top element **F**

Print F STACK D, E

Note that only neighbor **D** of **F** is not pushed onto the stack, since **D** has already been pushed onto the stack.

g) Pop and print the top element E and push onto the stack all 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.**

6. Attempt any TWO of the following: 16

a) Covert the following infix expression into a postfix expression and show details of stack at each step of conversion.

Expression: ((A+B)*D) ∧ (E - F)

(Correct postfix expressions - 2 Marks, For steps - 6 Marks)

Ans:

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-^

Postfix expression =AB+D*EF-^

b) Write a program to count the number of node in a Binary Search Tree.

(Program – 8 Marks, Any other logic can be considered))

Ans:

```
struct node
    {
        int data;
        struct  node *left;
        struct node *right;
     };
voidcountnodes(Node *root)
{
        static  int count=0;
        node *temp=root;
        if(temp!=NULL)
        {
                count++;
                countnodes(temp→left);
                countnodes(temp→right);
        }
        return count;
}
void main()
{
        countnodes(root);
        getch();
}
```

c) Consider the graph shown in Figure No. 2.

(i) Give adjacency matrix representation.

(ii) Give adjacency list representation of the graph.

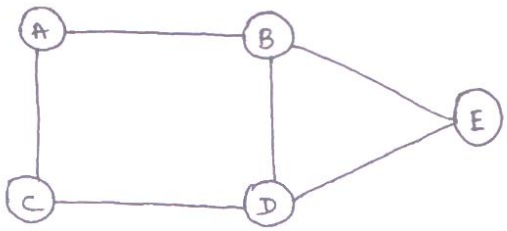


Fig. No. 2

(Adjacency matrix – 4 Marks, Adjacency list – 4 Marks)

Ans:

(i) Adjacency matrix

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

(ii) Adjacency list