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.1.

A. Attempt any three.

1) Explain time complexity and space complexity. (2M for time complexity and 2M for space complexity)

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 n^2 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.

2) WAP to implement linear search for 10 elements in an array.
(Correct program 4 M) (Any other relevant logic can be considered.)

```
# include <stdio.n>
# include <conio.n>
void main ()
{
int a [10] = {10, 20, 30, 40, 50,60,70,80,90,100};
int num, i;
printf ("Enter element to be searched");
scanf ("%d",&num);
for (i = 0; i<10; i ++)
{
if (a[i] = = num)
{
Break;
}
}
if (i = = 5)
printf ("Number not found");
else
printf ("Number found at position %d", i);
}
```

3) Explain stack as an abstract data type. (4 Marks) (Any representation of an ADT containing elements and operation shall be considered)

Stack as an abstract data type contains **stack elements and its operations**.
Stack elements: a **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**.

4) Explain the concept of information, Next, Null Pointer and empty list with respect to link list.(each term 1M)

a. Info Field: It is used to store data inside the node.
b. NEXT: It is used to store reference of next element in the list.
c. Null Pointer: It is used to specify end of the list. The last element of list contains NULL pointer to specify end of list.
d. Empty List: A linked list is said to be empty if head (start) node contains NULL pointer.

5) Write an algorithm for quick sort. (4 marks) (any other relevant steps shall be considered)

Step 1: Use two index variables i& j with initial values of 1st index position & n-1 index position respectively.

Step 2: Compare ithindex with pivot element till it finds greater number than pivot element. Increment ith by 1 if ith element is less than pivot element

Step 3: jth element is compared with pivot element till it finds a number less than pivot element Decrement jth by 1 if jth element is greater than pivot element.

Step 4: After finding elements greater than & less than pivot elements interchange both the elements.

Step 5: After interchange again increment & decrement current position of i& j, & compare each element with pivot element.

Step 6:once all elements are compared with pivot element fix the final position of pivot element in the list.

Step 7: Divide the total array into 2 parts without including fixed position element.

Step 8: Each part then should be sorted with the same procedure as mentioned above till you get a sorted array.

B. Attempt any Two:

1) Write a program to arrange 10 elements in an ascending order. (4 marks)
(Any sorting logic can be considered to arrange 10 elements in ascending order.)

Example bubble sort logic:

```
{
inti,j,temp;
printf("\n\n BUBBLE SORT");
for(i=0;i<10;i++)
 {
for(j=0;j<10-i;j++)
 {
if (a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
}
```

2) Describe priority queue with an example?(Explanation 2M, example 2M)

A priority queue is a queue in which the intrinsic ordering among the elements decides the result of its basic operations i.e. the ordering among the elements decides the manner in which Add and Delete operations will be performed.

In a priority queue,

1) Each element is assigning a priority.

2) The elements are processed according to, higher priority element is processed before lower priority element and two elements of same priority are processed according to the order of insertion.

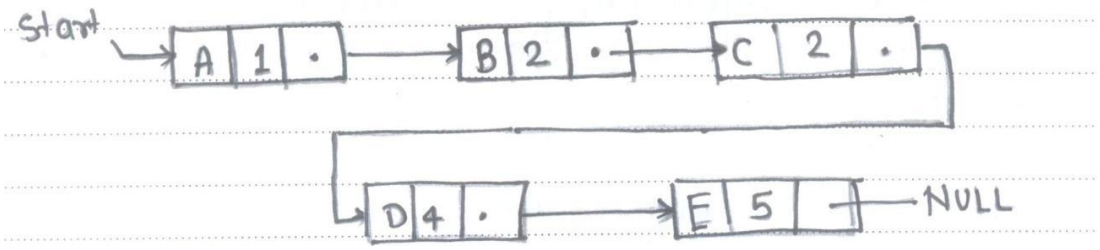
(Represent either with array or linked list)

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:



Above figure shows priority. Queue with 5 elements where B & C have same priority number.
Each node in above priority queue contains three items.
i. Information field INFO
ii. A priority number PR No
iii. Link Next

An example where priority queue are used is in operating systems. The operating system has to handle a large number of jobs. These jobs have to be properly scheduled. The operating system assigns priorities to each type of job. The jobs are placed in a queue and the job with the highest priority will be executed first.

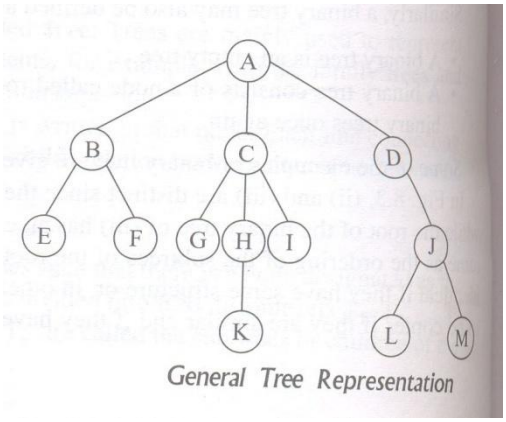
3) List types of trees and explain anyone.(list type 1M,any one type explanation 2M,diagram 1M)

Types of tree:-

- General tree
- Binary tree
- Binary Search tree

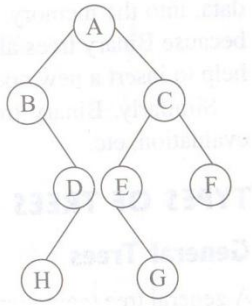
General tree: It is defined as a non-empty finite set T of elements, called nodes, such that:

- The tree contains the root element.
- The remaining elements of the tree form an ordered collection of zero or more disjoint trees T1,T2,…,Tm.T1,T2,…,Tm are sub trees of root and roots of T1,T2,…,Tm are called as successors of root node.



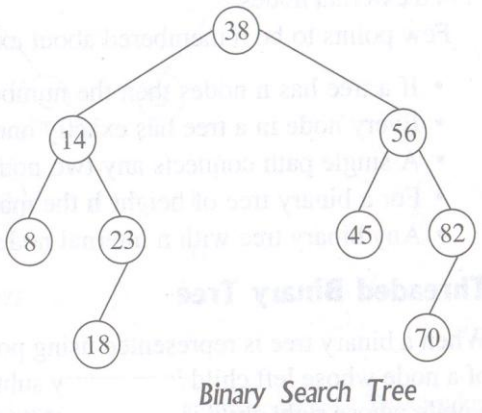
General Tree Representation

Binary Tree: It is a special type of tree in which every node or vertex has nochildren, one child or two children. It is a data structure in which a node can have at most two children. Child of node in a binary tree on the left is called as 'left child' and the node on the right is called as 'right child'.



Binary Tree

Binary Search tree: 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 the root node are placed at the right of root node.



Binary Search Tree

Q.2. Attempt any four:

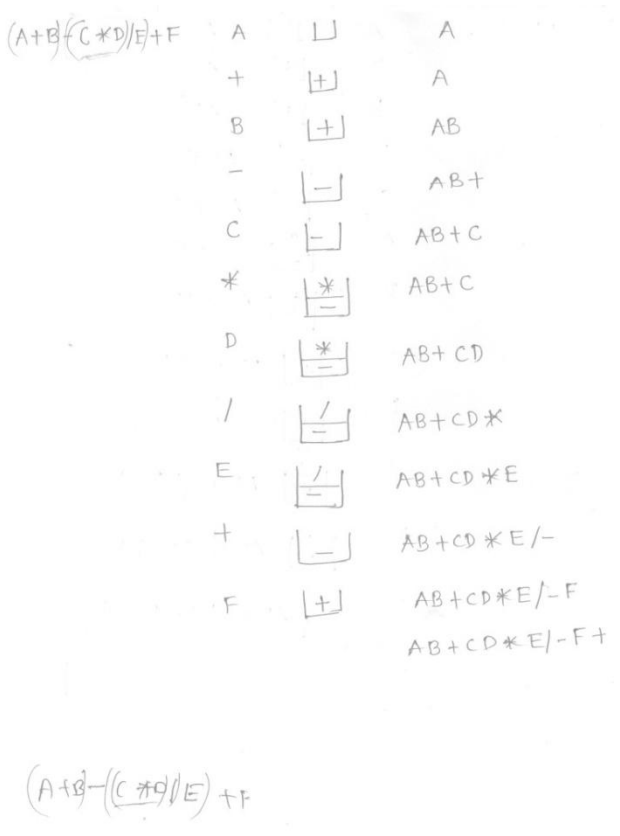
1) WAP to implement bubble sort. (Complete program 4 marks)

(Note: program can be written differently depending on the same logic. It may be without separate function) (Given solution is only code for separate function. Same logic from the code can be used in single program without function.)

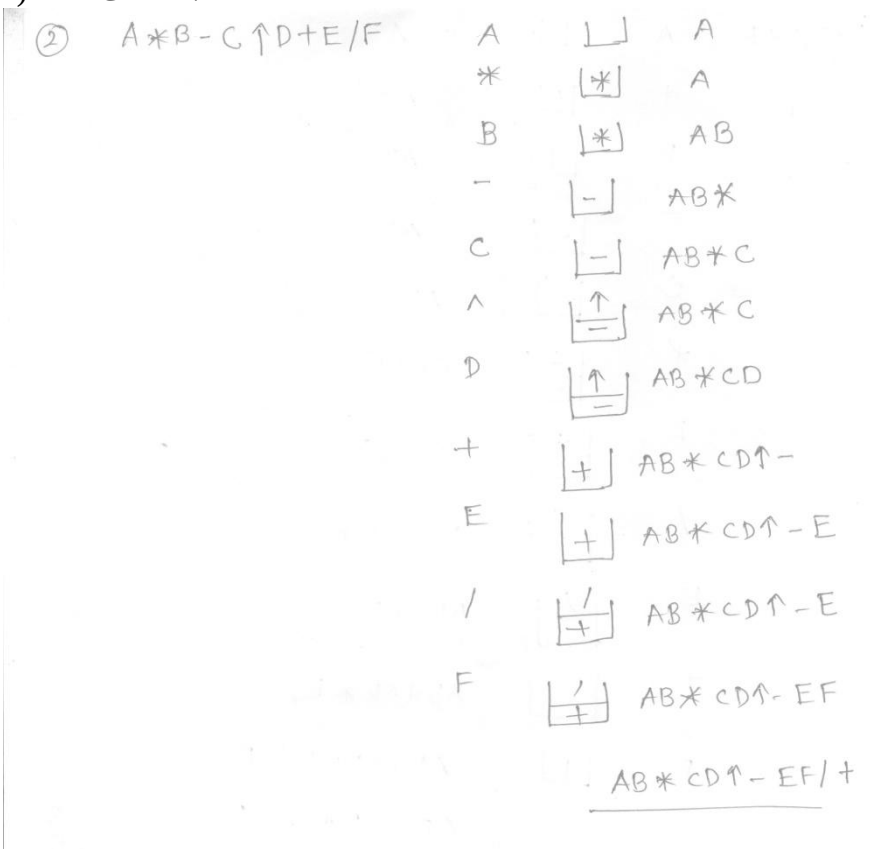
```
bsort(int a[],int n)
{
inti,j,temp;
printf("\n\n BUBBLE SORT");
for(i=0;i<10;i++)
 {
for(j=0;j<10-i;j++)
 {
if (a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
}
```

2) Convert the following infix expression to its postfix form (2 marks each)

i) A+B-C*D/E+F



ii)A*B-C^D+E/F



3) Explain the concept of circular queue.(Explanation 3 marks and example 1 mark)

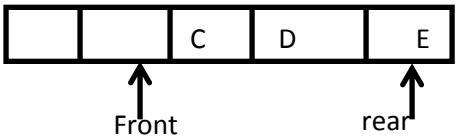
Circular queue are the queues implemented in circular form rather than in a straight line. Circular queues overcome the problem of unutilized space in linear queue implemented as an array. The main disadvantage of linear queue using array is that when elements are deleted from the queue, new elements 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 0^{th} 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.

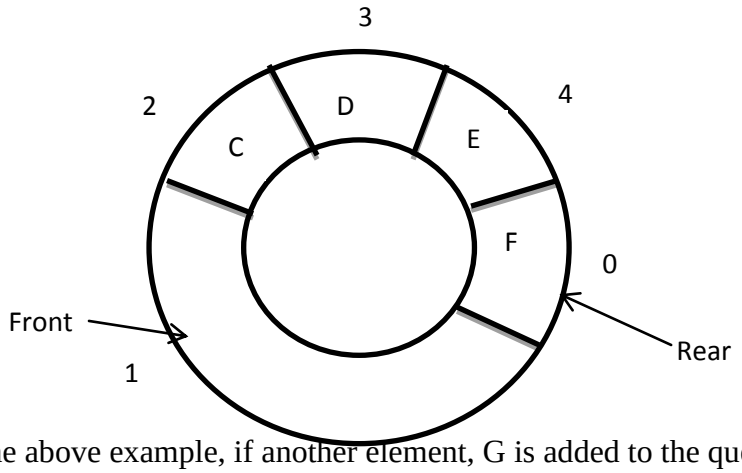
Rear can be incremented circularly by the following code.

```
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.



In the above example, if another element, G is added to the queue, i.e. rear and front coincide. But rear and front coincide even when the queue is full is empty. Thus rear and front cannot be used for both i.e. to check for empty queue as well as the condition for a full queue.
The rear ==front condition is used to check for the empty queue sinceinitially both are initialized to the same value. Thus to check for queue full condition there are three methods.

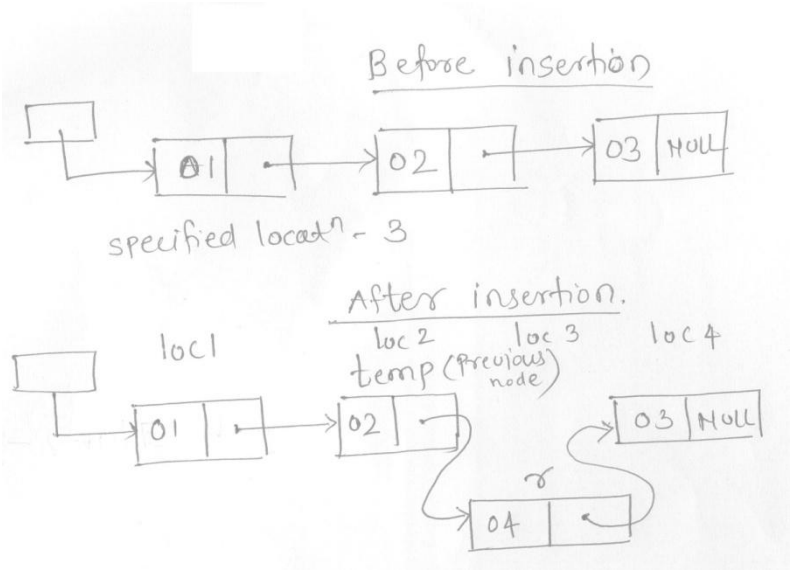
1. Use a counter to keep track of the number of elements in the queue. if this counter reaches to MAX the queue is full.
2. After remove operation rear= front, then set to -1. After add operation if rear = front then will say that queue is full.
3. By checking (rear + 1) % MAX== front.

4) Write an algorithm to insert a node in between in a link list. (4 marks)

Algorithm to insert an element insert a node in between in a link list.

```
void add_at_specified(struct node *q, intloc, int no)
 {
 Step 1: - Begin
 Step 2: -  Allocate memory for temp node and new node (r); set r->info = no

 Step 3:-  Initialize temp to the beginning of the linked list.
 Step 4:-  Traverse temp till the previous node of the specified location given by loc in the linked list (Previous node = temp node)
 Step 5: -  Set r->next = temp-next;
           Set temp->next = r;
Step 6 : - Stop
}
```

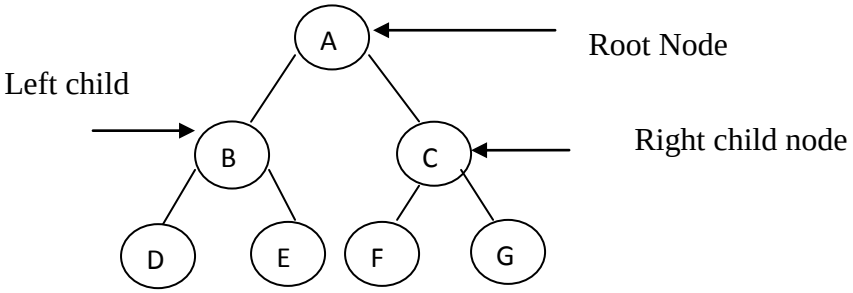


*(***NOTE: Examiner should consider the logic behind the algorithm. It may vary according to books, knowledge.)**

5) Describe the concept of binary tree and its applications. (Binary tree explanation-2M,application 2M)

Binary Tree:-

A binary tree is a special type of tree in which every node has no child, one child or two children. A child node at the left side of root node is called as 'left child' and a child node at the right side of root node is called as 'right child'.

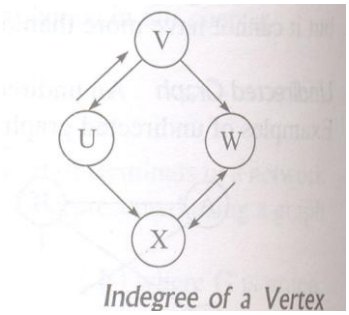


Application:- binary tress are used to represent a non-linear data structure.

- Binary tree is used in searching algorithms. Binary search uses binary search tree. A binary search tree always has two children and the left child node is always less than the root node, right child node is always greater than the root node. Binary search tree reduces the complexity of searching algorithm.
- Binary tree is used in populating a voluminous, relational and hierarchical data into the memory. As binary tree allow the algorithms to access a particular node at low cost and also help to insert new node easily, it increases the efficiency of the algorithm which manages the data.

6) Explain indegree and outdegree of a graph with example.

indegree 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.
Outdegree 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.



Indegree of a Vertex

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

Q.3. Attempt any Four:

1. Explain the two approaches to designing an algorithm.(2 Marks each for the approaches)

The two approaches to design an algorithm are

a) Top down approach: A **top-down** approach is also known as stepwise design, is essentially the breaking down of a system to gain insight into its compositional sub-systems. Example is c programming
b) Bottom up approach: A **bottom-up** approach is the piecing together of systems to give rise to more complex systems, thus making the original systems sub-systems of the emergent system. Example is C++ programming

2. Write an algorithm to implement binary search.(Algorithm Four marks)

1. Binary search algorithm locates the position of an element in a sorted array.
2. Binary search works by comparing an input value to the middle element of the array.
3. The comparison determines whether the element equals the input, less than the input or greater.
4. When the element being compared to equals the input the search stops and typically returns the position of the element.
5. If the element is not equal to the input then a comparison is made to determine whether the input is less than or greater than the element.
6. Depending on which it is the algorithm then starts over but only searching the top or bottom of the array's elements.

3. **WAP using recursion to print the factorial of a number**)
(Program – 3 marks, Output – 1 Mark)

```
#include<stdio.h>
#include<conio.h>
int fact(int n);
void main()
{
   int n;
   clrscr();
   printf("/nEnter an integer:");
   scanf("%d",&n);
   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));
}
```

Output : Enter an integer : 6

The factorial of 6 is = 720

4. **Explain queue as an abstract data type, also give applications of queue.**
(ADT – 3 Marks, Application – 1 mark)

Definition:

1. Queue is the most common abstract data type.
2. It follows the first in first out (FIFO) structure.
3. The elements are added in the end known as the ‘rear’ of the queue.
4. The elements are removed front the first, known as the ‘front’ of the queue.

Operations: Insert, delete an element.

Application areas of queue:

1. Banking
2. Network

5. **WAP to search an element in a link list.**
(Program – 4 Marks, this program or any relevant program)

Code for searching an element in a list

```
void search(struct node *head, int key)
{
```

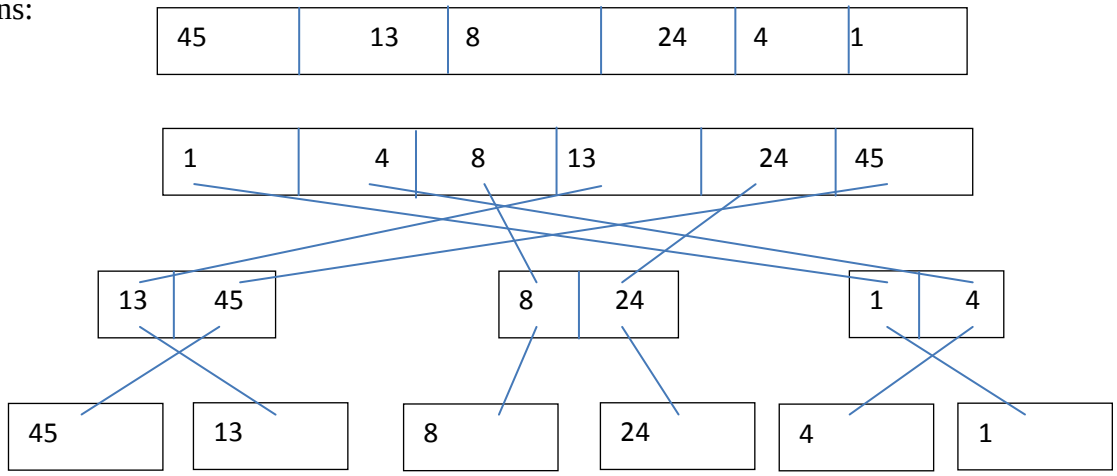
```
while (head != NULL)
   {
if (head->a == key)
      {
printf("key found\n");
return;
      }
head = head->next;
   }
printf("Key not found\n");
}
```

Where *head gives the first element of list;
Key gives the element to be searched in a list.

6. **Write the preorder and post order traversal of a tree.**
   (Pre Order – 2 marks, Post Order – 2 Marks)
   Ans:
   Pre Order – 36, 25, 48,20,32,43,56,10,22,50,60,6,15,58,75
   Post Order – 6,15,10,22,20,32,25,43,50,58,75,60,56,48,36

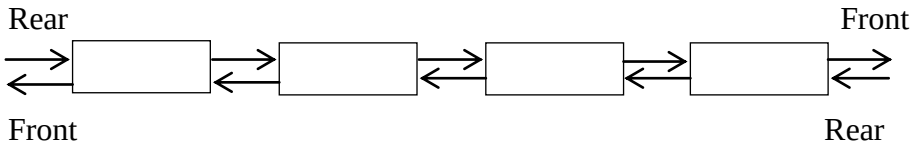
Q4. Attempt any four.

1. **Explain the big 'O' notation. (4 marks)**
   Big O notation characterizes functions according to their growth rates: different functions with the same growth rate may be represented using the same O notation. The letter O is used because the growth rate of a function is also referred to as order of the function. A description of a function in terms of big O notation usually only provides an upper bound on the growth rate of the function. Associated with big O notation are several related notations, using the symbols *o*, Ω, ω, and Θ, to describe other kinds of bounds on asymptotic growth rates.
2. **Explain merge sort for 6 numbers. (4 marks)**
   (For any numbers)
   Ans:

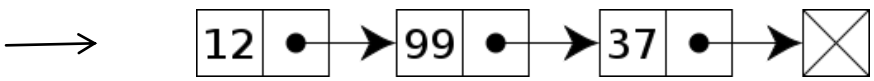


3. **Explain the concept of the double ended queue.**
   (Explaination-3 marks, Diagram – 1 Mark)

1. A double-ended queue or dequeue is an abstract data structure that implements a queue for which elements can only be added to or removed from the front (head) or back (tail).
2. It is also often called a head-tail linked list.
3. Dequeue is a special type of data structure in which insertions and deletions will be done either at the front end or at the rear end of the queue.
4. The operations that can be performed on dequeues are
   a. Insert an item from front end
   b. Insert an item from rear end
   c. Delete an item from front end
   d. Delete an item from rear end
   e. Display the contents of queue



4. **Explain the concept of linear list with example.**
   (Explanation – 3 Marks, Diagram 1 Mark)
   1. Linked list is a data structure consisting of a group of nodes which together represent a sequence.
   2. Each node is composed of a data and a link to the next node in the sequence; more complex variants add additional links.
   3. This structure allows for efficient insertion or removal of elements from any position in the sequence.



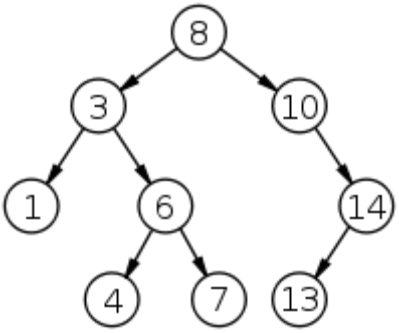
5. **What is meant by binary search tree with example?**
   (Explanation 3 marks, Example 1 mark)
   1. A binary search tree (BST), also called an ordered or sorted binary tree, is a node-based binary tree data structure where each node has a comparable key and an associated value
   2. The restriction that the key in any node is larger than the keys in all nodes in that node's left subtree and smaller than the keys in all nodes in that node's right sub-tree.
   3. Each node has no more than two child nodes.

The common properties of binary search trees are as follows:

a) The left subtree of a node contains only nodes with keys less than the node's key.
b) The right subtree of a node contains only nodes with keys greater than the node's key.
c) The left and right subtree each must also be a binary search tree.
d) Each node can have up to two successor nodes.
e) There must be no duplicate nodes.

f) A unique path exists from the root to every other node.

Example



6. Explain the concept of hashing and hash function.
(Hashing – 2 Marks, hash Function)

Ans: The hashing is defined as

1. Key-value pairs are stored in a fixed size table called a hash table.
2. A hash table is partitioned into many buckets.
3. Each bucket has many slots.
4. Each slot holds one record.

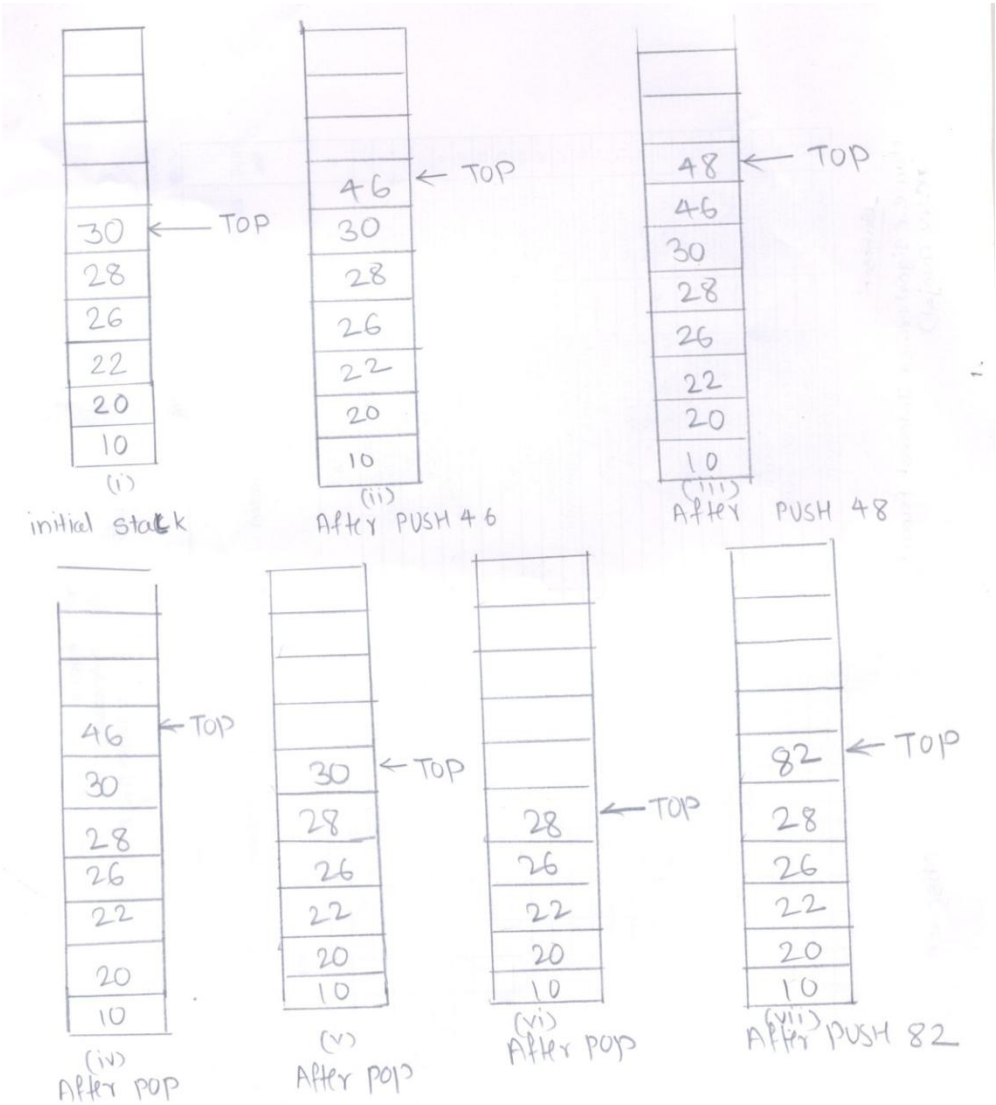
A hash function f(x) transforms the identifier (key) into an address in the hash table.

Q.4. Attempt any Two.

1. What is the Affect of PUSH and POP operation on to the stack? The stack contain 10,20, 22, 26, 28, 30 with 30 being at top of the stack. Show diagrammatically the effect of
   PUSH 46
   PUSH 48
   POP
   POP
   POP
   PUSH 82

   (1mark each explanation of push and pop. 6 marks for diagram of affect of operation)

   1) PUSH: The process of adding new element to the top of the stack is called PUSH operation. As the new element is added to the top, after every push operation the top is incremented by one.
   2) POP: The process of deleting an element from top of the stack is called POP operation. As deletion takes place from the from top of the stack. After every POP operation the top of the stack is decremented by one.



2. Write a menu driven program to insert, delete an element in a queue and display the queue.

(2 marks each for insert, delete and display function. 2 marks for menu driven main program)

```
// Implementation Of Queue And Operations On Queue
#include<stdio.h>
#include<conio.h>
# define max 6
int q[max];
int f=0,r=-1;
void empty();
void full();
void dis();
void insert();
void del();
void empty()
{
```

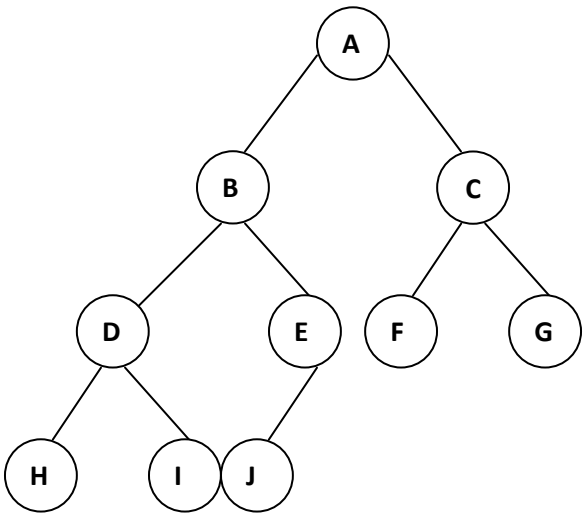
```
if(r<=-1)
{
printf("\nQueue is empty");
}
else
{
printf("\nQueue is full");
}
}
void full()
{
if(r>=(max-1))
{
printf("\nQueue is full");
}
else
{
printf("\nQueue is not completely full");
}
}
void insert()
{
intx,j;
if(r>(max-1))
{
printf("\nQueue is full");
}
else
{
r=r+1;
printf("\nEnter the element:");
scanf("%d",&q[r]);
}
}
void del()
{
int t=0;
if(r<=-1)
{
printf("\nQueue is empty:");
}
else
{
t=q[f];
if(f!=r)
{
f++;
}
else
{
```

```
f=0;
r=-1;
}
}
printf("\nThe deleted element is:%d",t);
}
void dis()
{
if(r<=-1)
{
printf("\nQueue is empty");
}
else
{
inti;
for(i=f;i<=r;i++)
{
printf("\t%d",q[i]);
}
}
}
void main()
{
intch;
clrscr();
do
{
printf("\n....Queue Operations....");
printf("\n1.Insert \t2.Delete \t3.Queue Full");
printf("\t 4.Queue Empty \t5.Display \t6.Exit");
printf("\nEnter your choice:");
scanf("%d", &ch);
switch(ch)
{
case 1:
insert();
break;
case 2:
del();
break;
case 3:
full();
break;
case 4:
empty();
break;
case 5:
dis();
break;
case 6:
```

```
break;
default:
printf("\n\n!!!!!Invalid Option!!!!!");
}
}while (ch!=6);
getch();
}
```

Note: program can be written differently depending on the same logic

3. Explain the following terms w.r.t tree with diagram (any four)
(Any relevant diagram of tree and definition of any four term, each term 2 marks)

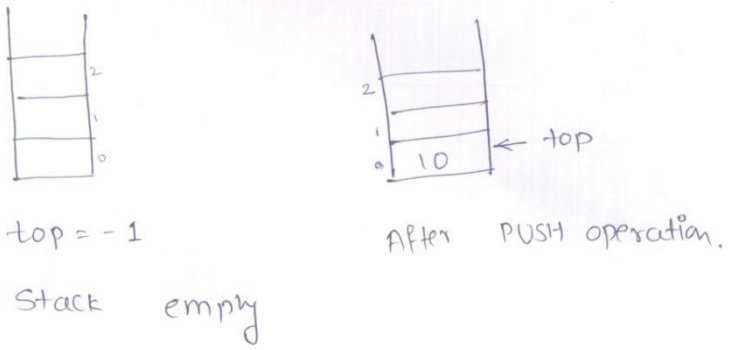


Degree of node is the total no. of child nodes of that node .
Degree of node B is 2
Degree of a tree is maximum number of child nodes of any node. In above example degree of a tree is 2.
Level of the node: Root is having 0 level. Level of any other node in the tree is one more than the level of its father.
Level of node C is 1, level of node G is 2
Leaf node: A node of degree 0 is known as a leaf node. A leaf node is terminal node and has no children. In fig H, I, J, F, G are leaf nodes.
Height of a tree: For a tree with just one node, the root node, the height is defined to be 0, if there are 2 levels of nodes the height is 1 and so on. A null tree (no nodes except the null node) is defined to have a height of –1.
In example height of tree is 4(maximum level +1)
In degree: Number of edges coming towards node is indegree of node.
Indegree of node B is 1
Outdegree: Number of edges going out from node is outdegree of node.
Out degree of node D is 2

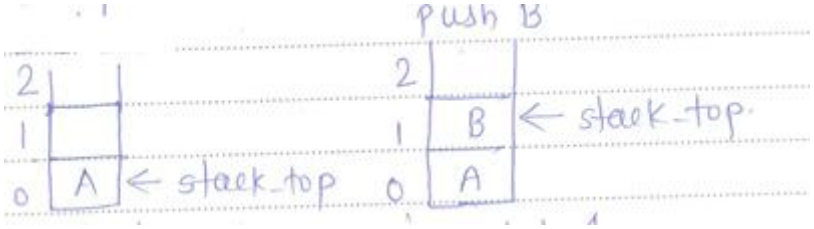
Q.6. Attempt any two.

1. Explain the concept of representing stack through arrays. Explain the concept of PUSH, POP and TOP of stack with example.
(2marks for stack representation, PUSH-2MARKS, POP 2 MARKS, TOP 2 MARKS) (Diagram required))

Stack is ordered collection of items. In it item inserted from one end of stack known as top of stack. In stack item deleted from top of stack.



Elements of stack accessed only through TOP of stack.
Push : Push operation is used to insert an element into the stack.



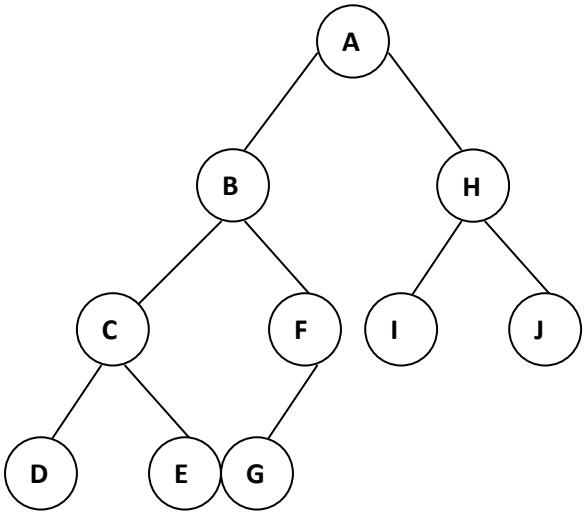
Initially stack top is set to -1 as array started with 0 to indicate empty stack. When element is to be inserted first increment the stack top by 1 and then insert two new elements into it.
Algorithm for push operation (Steps)
Step 1: First check for overflow i.e. if stack top =max-1 (maximum size of an array) then push operation cannot be performed.
Step 2: If there is no overflow then increment the stack top by 1 and insert new element at stack top position.
Pop operation is used to remove an element from stack .an element at the stack top is removed then pop operation is called.

When pop operation is called stack top is decremented by 1.
Algorithm (Procedure) for pop operation
Step 1: First check for underflow i.e. if stacktop is -1 means there is no element in the stack then pop operation cannot be performed.

Step 2: If there is no underflow then decrement the stacktop by 1. It removes an element placed at stacktop of the stack.

2. Write an Algorithm for post order traverse in a tree.
(Algorithm 4 marks, any example showing post order traversal 4 marks)
Post order traversal in a tree
Algorithm:

1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the root.

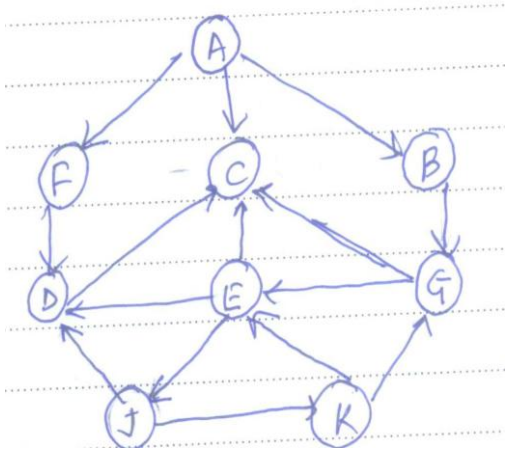


Post order traversal (left-right-root): D E C G F B I J H A

3. Describe breadth first search traversal in a graph with example.
(Algorithm 4 marks, Any relevant example explaining BFS 4 marks)

Algorithm for BFS:

1. Initialize all nodes to ready state.
2. Insert starting node in a queue and change its state to waiting state.
3. Repeat steps 4 to 6 till the queue becomes empty.
4. Remove front node N from queue 2 change its status to visited.
5. Insert all adjacent nodes of N at the rear end of the queue and change their status to ‘waiting state’.
6. From the origin find path from source node to destination node or from the queue element list find all nodes that are reachable.
7. Stop.



Front of the queue is set to '0'
Rear of the queue is set to '0'
Queue is used to indicate elements of the graph which are visited.
Origin is used to keep track of origin of each node. Find all nodes reachable from 'A'

1. Insert A into a queue.

A									

Front=1 Rear=1
Queue=A Origin=0

2. Remove front element A and insert adjacent nodes of a in queue.

	F	C	B						

Queue=A front=1
Origin=O, A, A, A rear=3

3. Remove element F and insert its adjacent nodes in the queue.

		C	B	D					

Queue=A, F, C, B, D front=2
Origin=O, A, A, A, F rear=4

4. Remove front element C and insert its adjacent nodes in the queue(F is already visited)

			B	D					

Queue=A, F, C, B, D front=3
Origin=O, A, A, A, F rear=4

5. Remove front element B and insert adjacent nodes

				D	G				

Queue=A, F, C, B, D, G front=4
Origin=O, A, A, A, F, B rear=5

6. Remove front element D and insert adjacent nodes (C is already visited)

					G				

Queue=A, F, C, B, D, G front=5
Origin=O, A, A, A, F,B,G rear=5

7. Remove front element G and insert adjacent nodes

						E			

Queue=A, F, C, B, D, G, E front=6
Origin=O, A, A, A, F, B, G rear=6

8. Remove front element E and insert its adjacent nodes (D is already a visited node)

							J		

Queue=A, F, C, B, D, G, E, J
Origin=O, A, A, A, F,B, G,E

9. Remove J

Queue = A, F,C,B, D, G, E,J
All nodes readable from A-A, F, C, B, D, G, E, J
Origin= O, A, A, A, F, B, G, E, J