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 SIX: **12**

a) Define time complexity and space complexity.
(Definition of Time complexity 1M, Space complexity 1M)

Ans.

Time complexity:-
Time complexity of a program/algorithm is the amount of computer time that it needs to run to completion (execution).

Space complexity:-
Space complexity of a program/algorithm is the amount of memory that it needs to run to completion (execution).

b) Define data structure. Enlist any two types of non-linear data structure.
(Definition data structure 1M, List of non-linear data structure two types -1/2M each)

Ans.

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

Example of non-linear data structure: tree, graph,

c) Define searching. State two advantages of searching.
(***Definition of searching 1M, any two relevant advantages 1/2M each)***

Ans.

It is the process of finding a data element in the given data structure.

Advantages:

1. To check existence of an element
2. To determine position of a search element
3. To determine number of occurrences of a search element.
4. To locate data set in given data structure for further operations (if any).

d) Define push and pop operation of stack.
(***Definition of PUSH and POP 1M each)***

Ans.

PUSH: Push operation is used to insert an element at TOP of stack

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

e) Define the terms of tree-node, leaf, sibling, in-degree.
(*Definition of each term 1/2M)*

Ans.

Node: A node of a tree is an item or information along with the branches to other nodes.

Leaf Node: A leaf node is a terminal node of a tree. It does not have any child.

Siblings: Nodes that have same parent node are called as siblings.

In-degree In degree of a node is number of edges arriving at that node.

f) Define tree traversal. List different types of tree traversal.
(Description of tree traversal 1M, list of types 1M)

Ans.

Tree traversal: It is a process of visiting each node in tree exactly once in systematic manner.

Types of traversal:

- In-order traversal
- Pre-order traversal
- Post-order traversal

g) What is sorting? Enlist two categories of sorting.
(*Definition of sorting 1M, list of two categories 1/2M each)*

Ans.

Sorting: It is a process of arranging collection of items in ascending order or in descending order.

The sorting is classified into two categories:

1. Internal Sorting.
2. External Sorting.

h) Define hashing.
(Definition – 2M)

Ans.

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

Q.1) B) Attempt any TWO: 8

a) Describe any four operations on data structure.
(Description of any four operations: 1M each)

Ans.

Operations on Data Structure:-

- Insertion: It is adding a new data in data structure.
- Deletion: It is removing a data from existing data structure
- Searching: It is finding location or existence of data within given data structure
- Sorting, arranging data in some logical order: It is used to arrange the data items in some order i.e. in ascending or descending order in case of numerical data and in dictionary order in case of alphanumeric data.
- Traversing: It is an operation that access each and every element.
- Merging: It is used to combine the data items of two data structure into single data structure.

b) Distinguish between linear search and binary search. (any 4 points)
(Any four relevant points 1M each)
*****Note: example shall be considered as point of comparison***

Ans.

- Binary search requires the input data to be sorted; linear search doesn't
- Binary search requires an *ordering* comparison; linear search only requires equality comparisons
- Binary search has complexity O(log n); linear search has complexity O(n)
- Binary search requires random access to the data; linear search only requires sequential access

- Linear search is too slow to be used with large lists due to its o(n) average case performance. On the other hand, binary search is considered to be a more efficient method that could be used with large lists.

c) Define the terms FRONT and REAR of queue. Enlist any four operations of queue.
(Definition of FRONT and REAR-1M each, list of four operations 1/2M each)
***** Note: diagram of queue and explanation of operations is optional***

Ans.

1) **FRONT**: A queue is ordered collection of item from which elements deleted at one end called as FRONT of queue.
2) **REAR**: A queue is ordered collection of item from which elements inserted from other end called as REAR of queue.

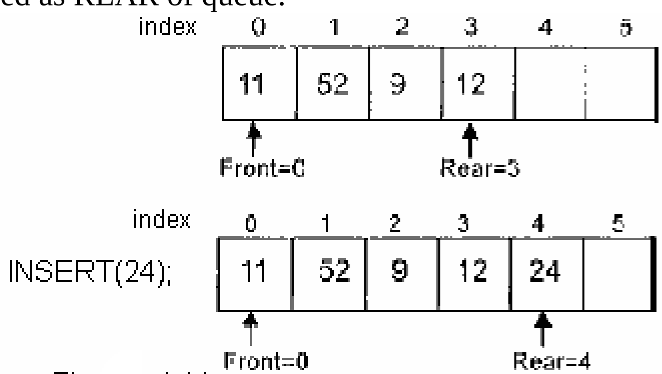


Fig .: Addition of item at the rear end of the queue

Operation:

- Isempty: To check whether queue is empty or not.
- Isfull: To check whether queue is full or not.
- Insertion : Adding new elements to the queue
- Deletion: Removing an element from the queue.
- Retrieve: Retrieve the value of first element of queue, if queue is not empty using/referring FRONT pointer's value.

Q.2. Attempt any FOUR: **16**

a) Describe quick sort. State its advantages and disadvantages.
(Description 2M, advantages 1M, disadvantages 1M)

Ans.

In quick sort, in each pass only one element is fixed at its final position in a list. For sorting, fix the first element as a pivot element. Use two index variables (i,j) with initial values of first index position and n-1 index position respectively. Compare a pivot element with i^{th} index element till you find a number greater than pivot element. If the i^{th} element is less than pivot element then increment the index by one. If i^{th} element is greater than pivot element than mark the element and then compare pivot element with j^{th} element till you find a number less than pivot element. If the j^{th} element is greater than the pivot element then decrement index by one. If i^{th} element is less than pivot element mark that element. After finding elements greater than and less than pivot element, interchange both the elements. Again compare pivot element with i^{th} and j^{th} element till i and j are equal to each other or cross to each other. Then fix the pivot element at its position such a way that all elements preceding it should be less than it and all the elements following it should be greater than it. Then divide the list into two parts without including fix element. Each part of the list should be sorted with the same procedure till all elements are placed at their final position in sorted list.

Advantages:

- Efficient average case as compared to any other sorting algorithm.
- It is faster than other algorithm such as bubbles sort, selection sort and insertion sort.

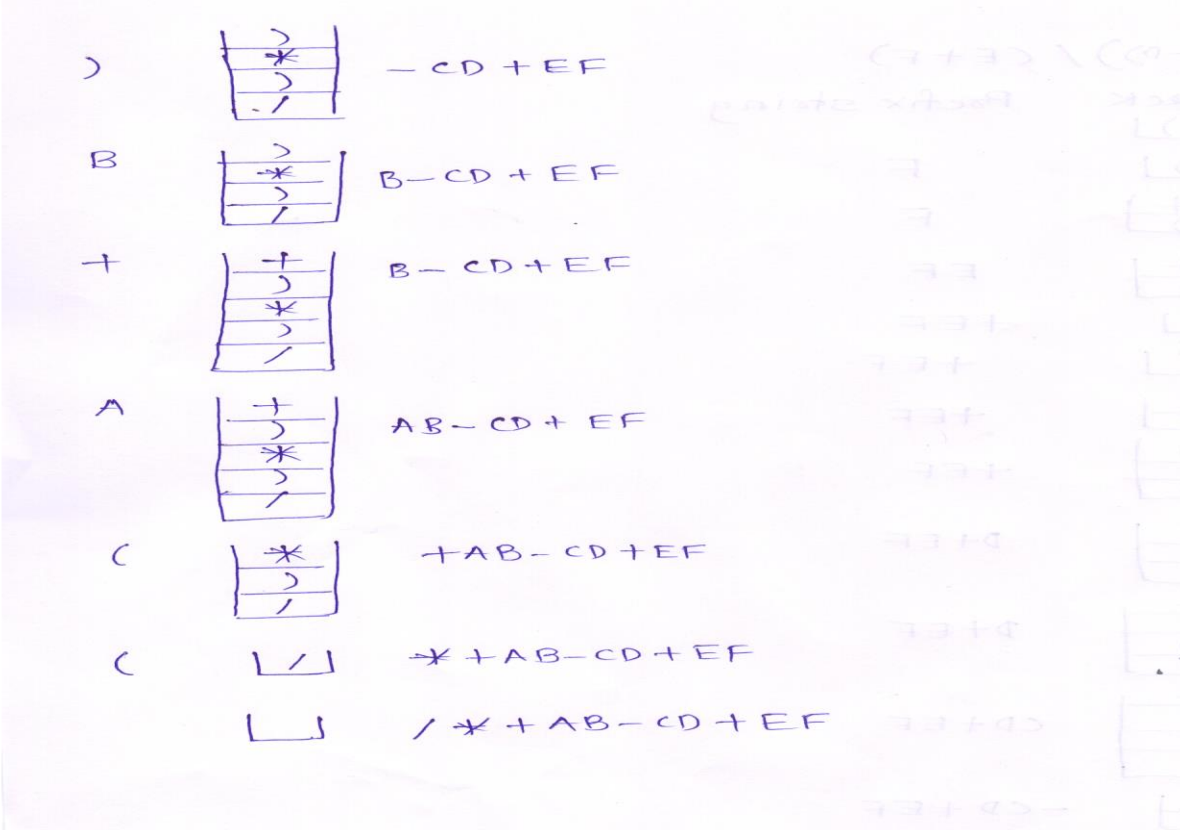
Disadvantages:

- It is complex and massively recursive.
- The worst-case complexity of quick sort is O(n^2), which is worse than the O (n log n) worst-case complexity of algorithms like merge sort, heapsort, binary tree sort, etc.

b) Convert infix string ((A + B) * (C – D)) /(E + F) into prefix string with stack. Show the content of stack in each step.
(Correct answer with stack content 4M)

Ans.

Read char	Stack	Prefix string
)	)	
F	)	F
+	+)	F
E	+)	EF
(		+EF
/	/	+EF
)	) /	+EF
)	)) /	+EF
D	)) /	D+EF
–	–)) /	D+EF
C	–)) /	CD+EF
(	) /	–CD+EF
*	*) /	–CD+EF



Prefix string: /*+AB-CD+EF

c) List and define operations of linked list.
(Any four operations: 1M each)

Ans.

The basic operations that can be performed on a list are:

Creation: This operation is used to create a node in the linked list.

Insertion: This operation is used to insert a new node in the linked list.

- At the beginning of the list,
- At a certain position and
- At the end.

Deletion: This operation is used to delete node from the list.

- At the beginning of the list: To delete first element
- At a certain position:-
- At the end.

Traversing: It is a process of going through all the nodes of a linked list from one end to the other end.

Display: This operation is used to print all node's information field.

Searching: To search a specific element in given linked list.

Count:-To count number of nodes present in list.

Eraseall: To free/delete all elements from the list.

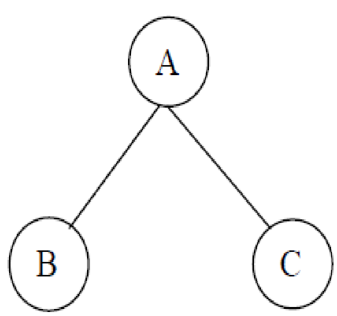
d) Describe the process of pre-order traversal of binary tree. Give one example.
(Description or algorithm 2M, any relevant Example 2M)

Ans.

Preorder traversal:-In this traversal method first process root node, then left sub tree and then right sub tree.

Procedure:-
Step 1: Visit root node
Step 2: Visit left sub tree in preorder
Step 3: Visit right sub tree in preorder

Example:



Preorder: A, B, C

e) Write a 'C' program to implement a queue with insert and delete operation.
(Logic for insert-2M, delete-2M)
Note: - Any logic shall be considered

Ans.

Implementation insertion and deletion on Queue Using array:

```
#include<stdio.h>
#include<conio.h>
#define max 3
int rear=-1;
int front=-1;
int queue_arr[max];
void insert();
void del();
void display();
void insert()
{
int insert_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()
{
int ch;
clrscr();
while(1)
{
printf("\n1.insert()\n2.delete()");
printf("\n enter values");
scanf("\n%d",&ch);
switch(ch)
{
case 1:
insert();
break;
case 2:
del();
break;
default:
printf("wrong choice");
}
}
}
```

f) Write a program in 'C' language for selection sort. Write complexity of selection sort.
(Correct logic-3M, Complexity-1M)
Note: - any relevant logic shall be considered)

Ans.

```
#include <stdio.h>
int 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++)
```

```
{
for(j=i+1;j<n;j++)
{
if(array[j]<array[i])
{
temp=array[i];
array[i]=array[j];
array[j]=temp;
}
printf("The Sorted List Is ");
for(i=0;i<n;i++)
printf("%d ",array[i]);

getch();
}
```

Complexity of selection sort O (n^2)

Q.3. Attempt any FOUR: 16

a) Write an algorithm to convert infix string into postfix (Fully parenthesized)
(Any correct algorithm indicating steps for conversion 4M)

Ans:

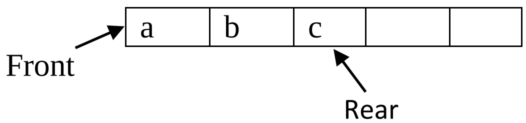
1. Accept infix string as an input array and add the symbol ‘#’ at the end of the array
2. Read the string symbols one – by – one from left to right
3. If the current symbol is an operand, add it into postfix array
4. If the current symbol is left parenthesis ‘(‘, push it into operator stack
5. If the current symbol is an operator POP all the operators having precedence greater than or equal to the current operator and add it to postfix array else PUSH the current operator into operator stack.
6. If the current symbol is right parenthesis ‘)’, POP all the operators from operator stack and Add it to the Postfix array till the respective left parenthesis is encountered
7. Repeat Steps 2 till 5, until the end of input array indicated by ‘#’ symbol.
8. Display Postfix expression for the given infix string.

b) Describe how queue works as an abstract data type.
(Definition-1M, Representaton-1M, operation-2M and conditions are to be considered as an optional)

Ans:

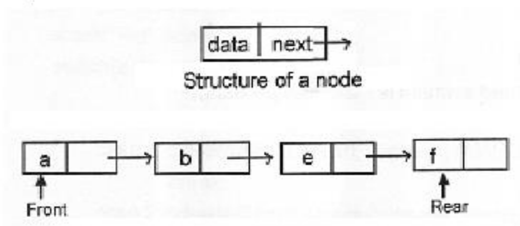
1. **Definition:** Queue is a linear data structure which follows First-In First- Out (FIFO) principle where, elements are added at rear end and deleted from the front end.

2. **Representation of a Queue**:
   a. Array Representation:



   b. Linked List Representation:

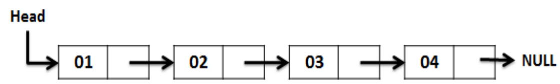
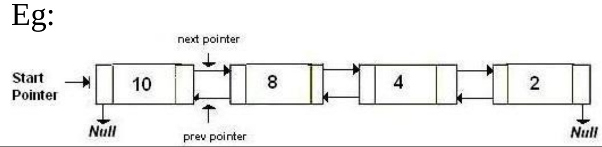
```
struct node
{
    int data;
    struct node * next;
};
```



3. **Operations performed on Queue:**
   a. isEmpty( )- Check whether the queue is empty
   b. isFull( )- Check whether the queue is full
   c. Insert( )- Insert an element in the queue at rear end
   d. Delete( )- Delete an element of a queue from front end
4. **Queue Overflow:** Inserting an element in a queue which is already full is known as Queue Overflow condition (Rear = Max-1)
5. **Queue Underflow:** Deleting an element from queue which is already empty is known as Queue Underflow condition (Front == Rear=-1)

c) Distinguish between singly linked list and doubly lined list. (4 points)
(For each point - 1M)
Ans:

Singly Linked List	Doubly Linked List
1. It is also called One way header list	1. It is also called Two way header list
2. Can be traversed in only one direction	2. Can be traversed in both directions – forward as well as backward
3. It has only 1 pointer	3. It has only 2 pointers

4. Node in a singly linked list has two fields – Information and next pointer	4. Node in a doubly linked list has three fields – information, next pointer and previous pointer
Eg: 	Eg: 

d) Define Tree. Create a binary tree for following data:
12, 25, 14, 8, 3, 5.
(For Definition - 2M & Tree Construction – 2M)

Ans:

Definition:

A tree is a finite set of nodes where starting node is called as Root node of the tree and the remaining nodes are represented as two sub-trees (left sub-tree and right sub-tree) of the root node.

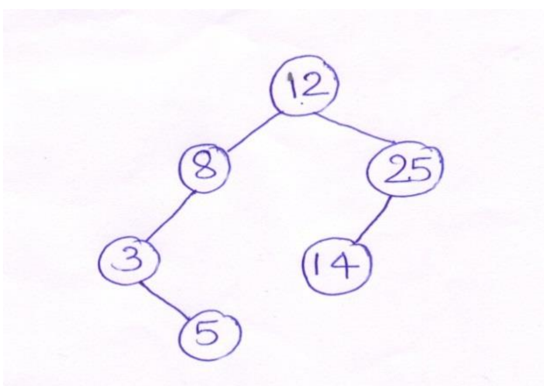


Fig: Construction of binary tree

e) Write a C program to traverse the tree in in-order.
(Assumption: Tree is already created, Node structure-1M, Traversal logic 3M)

Ans:

```
#include<stdio.h>
#include<conio.h>
struct node                                         // Node structure
{
 int info;
 struct node *left,*right;
 }*root;

void inorder(struct node *);

void main()
```

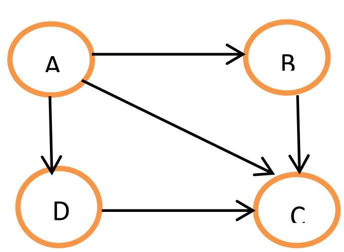
```
{
  clrscr();
  printf("Inorder Traversal\n");
   inorder(root);                              // Function call
   getch();
 }
 void inorder(struct node*q)                   // Function definition / logic
 {
   if(q!=NULL)
   {
      inorder(q->left);
      printf("%d\t",q->info);
      inorder(q->right);
   }
 }
```

f) Define the following terminologies with respect to graph:

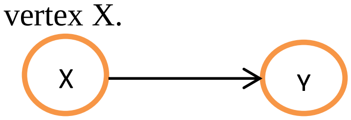
(i) direct graph
(ii) successor
(iii) predecessor
(iv) path
(Definition of each terminology – 1M)

Ans:

(i) Direct Graph:
A directed graph is defined as the set of ordered pair of vertices and edges where each connected edge has **assigned a direction.**

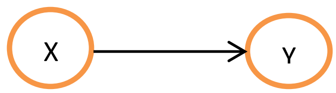


(ii) Successor:
If there is a directed edge from vertex X to vertex Y then vertex Y is said to a successor of vertex X.



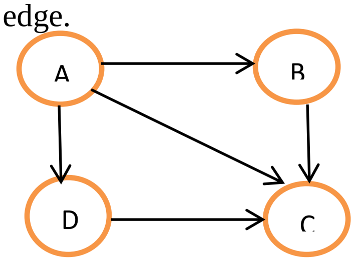
(iii)Predecessor:

If there is a directed edge from vertex X to vertex Y then vertex X is said to a Predecessor of vertex Y.



(iv)Path:

Path is a sequence of alternating vertices and edges such that each successive vertex is connected by the edge.



For the above graph paths can be A-B-C, A-B, A-C etc.

Q.4. Attempt any FOUR: 16

a) Describe Top-down and bottom-up approach by giving example.

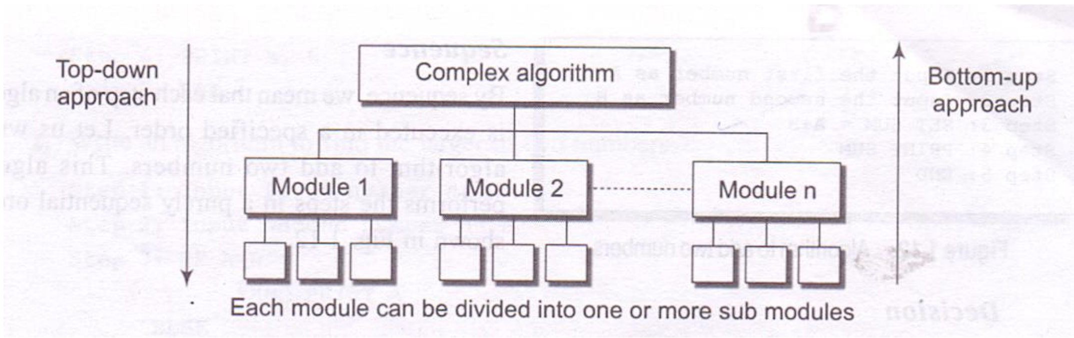
(Explanation of Top-down with example – 2M, Bottom-up with example – 2M)

Note: Any relevant example shall be considered.

Ans:

1. **Top-dpwn 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 subsystem 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 topmost 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 great 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.

3. **Example :**



b) Describe the representation of stack using arrays.
(Relevant description – 4M)

Ans:

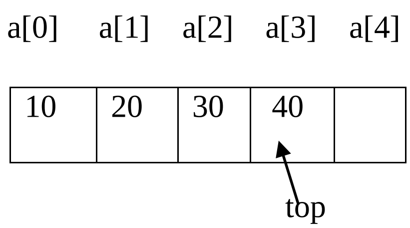
1. **Definition:** Stack is a linear data structure which follows Last-In First- Out (LIFO) principle where, elements are added and deleted from only one end called as stack top
2. **Array representation of a Stack:**

a[0]	a[1]	a[2]	a[3]	a[4]
10	20	30		

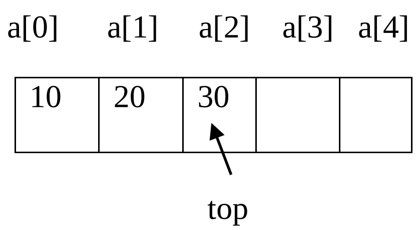
top

3. **PUSH Operation:**
   a. If (top==max-1)
      then display “Stack overflow”
   b. else
      do top=top+1
   c. a[top]=data
4. **POP Operation:**
   a. If (top==- 1)
      then display “Stack underflow”
   b. else
      do Data= a[top]
         Top=top-1

5. **Example:**
   a. PUSH 40



   b. POP



c) Write the procedure for inserting and deleting an element from queue.
(Insertion procedure- 2M, Deletion procedure- 2M)

Ans:

Procedure for Insert:-
Step 1: [check overflow]
if rear = max -1 then write “queue is full” return otherwise go to step 2
Step 2: [increment rear point]
rear = rear + 1
Step 3: [insert element]
a [rear] = item
Step 4: End / return to calling function

Procedure for Delete:-
Step 1: [check Underflow]
if front = -1 then write “queue is empty” return otherwise go to step 2
Step 2: [copy front element of queue]
Data = a [front]
Step 3: [check front and rear pointer]
if front == rear
  then front = rear = -1
else
  front = front + 1
Step 4: Return to calling function

d) Write an algorithm to traverse a singly linked list.
(Correct stepwise algorithm - 4M)
*****NOTE: Description with example can also be considered.***

Ans:

1. if(start==NULL)
   then display "linked list is empty".
2. Otherwise Visit each node of linked list and display its data till end of the list

```
 q=start          // Assign a temporary pointer q to starting node
 while(q!=NULL)
 do
  Display q->data   // display node information
  q=q->link;
```

e) Write a C Program to insert a new node at the beginning into singly linked list.
(Node structure- 1M, insertion logic -3M)

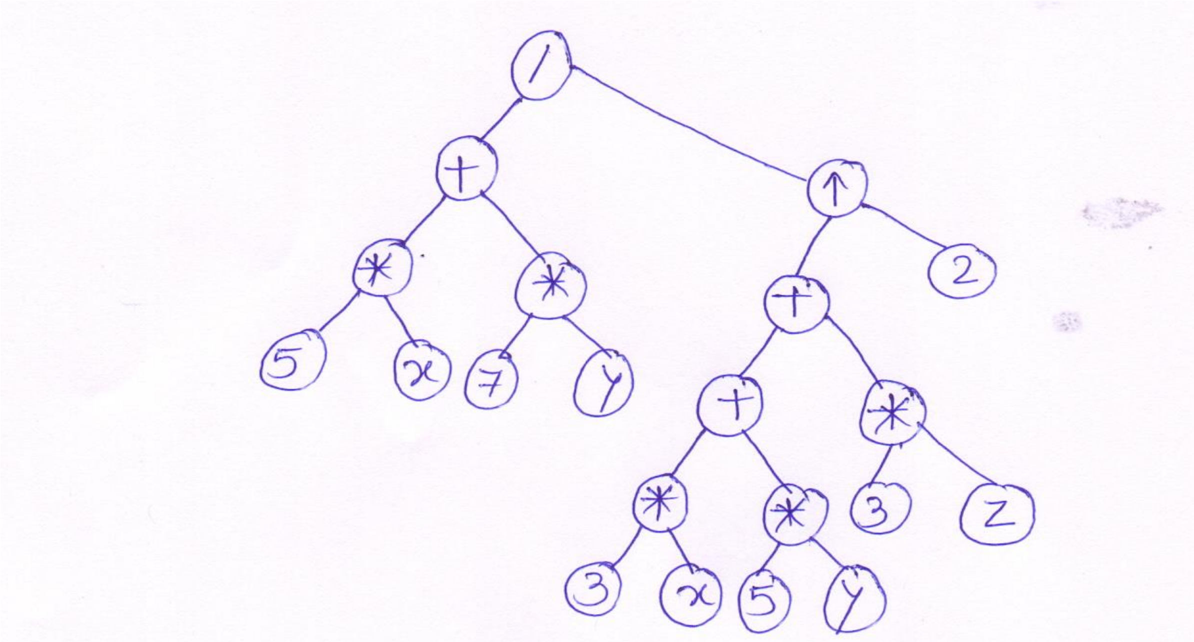
Ans:

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

void create_list(int);
void addatbeg(int);

struct node
{
 int info;
 struct node *link;
}*start;

void main()
{
   int m,n,pos,i;
   clrscr();
   start=NULL;
   printf("how many nodes u want\n");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
     printf("enter data value\n");
     scanf("%d",&m);
     create_list(m);
    }
```

```
    printf("enter data value\n");
    scanf("%d",&m);
    addatbeg(m);
    display(start);
    getch();
 }
void create_list(int data)
{
 struct node *tmp,*q;
 tmp= malloc(sizeof(struct node));
 tmp->info=data;
 tmp->link=NULL;
 if(start==NULL)
  start=tmp;
 else
 {
  q=start;
  while(q->link!=NULL)
    q=q->link;
  q->link=tmp;
 }
}

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

f) Draw the tree structure for the following expression:
$(5x+7y)/(3x+5y+3z)^2$
(Correct tree structure - 4M)

Ans:



Q 5. Attempt any TWO: 16

a) Describe radix sort algorithm. Sort the following numbers in ascending order using radix sort.
12, 8, 25, 4, 66, 2, 98, 225.
(Description of Algorithm 4Marks, radix sorting on given numbers-4Marks)

Ans: Radix sort algorithm:-

Radix Sorting: - In this method, ten buckets (0-9) are used to sort elements of an input list. All the elements are sorted according to their digit position from each element. In pass one each element is placed inside the bucket with respect its unit position digit. After placing all elements inside the buckets, read those from 0^{th} bucket to 9^{th} bucket. In pass 2, elements are placed in buckets with respect to 10^{th} position digit from each element. In each pass one position is considered to arrange all the elements in bucket. At the end of each pass elements are collected from buckets and given as input to the next pass. Total number of passes required for sorting is equal to maximum number of digits present in the largest number from the input list. Last pass gives sorted list after reading all elements from 0^{th} bucket to 9^{th} bucket.

Sorting of Given Numbers: -

Pass 1:

Bucket Elements	0	1	2	3	4	5	6	7	8	9
12			12							
8									8	
25						25				
4					4					
66							66			
2			2							
98									98	
225						225				

Output of Pass 1: 12 02, 04, 25, 225, 66, 8, 98

Pass 2:

Bucket Elements	0	1	2	3	4	5	6	7	8	9
12		12								
2	2									
4	4									
25			25							
225			225							
66							66			
8	8									
98										98

Output of Pass 2: 02, 04, 08, 12, 25, 225, 66, 98

Pass 3:

Bucket Elements	0	1	2	3	4	5	6	7	8	9
2	2									
4	4									
8	8									
12	12									
25	25									
225			225							
66	66									
98	98									

***Sorted list*: 2, 4, 8, 12, 25, 66, 98, 225**

b) Consider the following arithmetic expression written in postfix notation:
10, 2, *, 15, 3, /, +, 12, 3, 2, ↑, +, + . Evaluate this expression to find its value.
(Correct answer with appropriate steps 8 M)

Ans:-

Q. → 10, 2, *, 15, 3, /, +, 12, 3, 2, ↑, +, +

Ans: –

	Stack
Step 1 → Read 10	10
Step 2 → Read 2	2 10
Step 3 → Read * 10 * 2	20
Step 4 → Read 15	15 20
Step 5 → Read 3	3 15 20
Step 6 → Read / 15/3	5 20
Step 7 → Read + 20 + 5	25
Step 8 → Read 12	12 25
Step 9 → Read 3	3 12 25
Step 10 → Read 2	2 3 12 25
Step 11 → Read ↑ 3 ↑ 2	9 12 25
Step 12 → Read + 12 + 9	21 25
Step 13 → Read + 25 + 21	46

Final value of expression is $\boxed{46}$.

c) Describe the breadth first search technique with suitable example.
(Description of breadth first technique 4M, any suitable example 4M)

Ans.

Description of Algorithm:

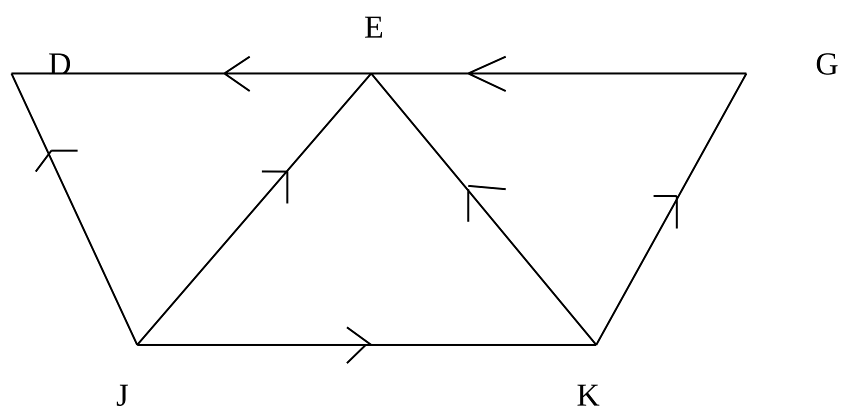
BFS (Breadth first search) is a graph search algorithm that begins that specific node and explores on the neighbouring nodes. It is used to find minimum path P between two particular nodes.

The algorithm works as follows:

Initialize all the nodes to the ready state. Insert starting node in a queue and change its status to waiting state. Then remove front node (N) from queue and change its status to visited state. Find adjacent nodes of removed node (N). Insert them into the queue and change their status to waiting state. It continues the process of insertion and deletion of elements till queue becomes empty. To find path algorithm uses origin to keep track of nodes whose adjacent nodes are inserted into the queue. At the end origin is used to find path from source node to destination node.

Example: - Consider following graph G.

Find all nodes reachable from node K to D.

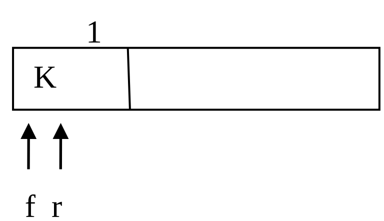


All nodes are in ready state.

Step 1- Initially insert node K in a queue and add NULL to orig

Front = 1 queue = K

Rear = 1 orig = O



Step 2 – Remove front element K and change its states to visited. Find its adjacent nodes and insert them into queue if their states is ready.

Front = 2 queue = K, E, G

Rear = 3 orig = O, K, K

1	2	3
	E	G
	↑ f	↑ r

Step 3 – remove front element E and change its states to visited. Find its adjacent nodes and insert them into queue if their states is ready.

Front = 3 queue = K, E, G, D

Rear = 4 orig = O, K, K, E

1	2	3	4
		G	D
		↑ f	↑ r

Step 4 – Remove front element G and change its states to visited. Find its adjacent nodes and insert them into queue if their state is ready. E is already visited.

Front = 4 queue = K, E, G, D

Rear = 4 orig = O, K, K, E

			D
		↑ f	↑ r

Step 5 - Remove front element D and change its states to visited. Find its adjacent nodes and insert them into queue if their state is ready.

Path between K to D is K -> E -> D

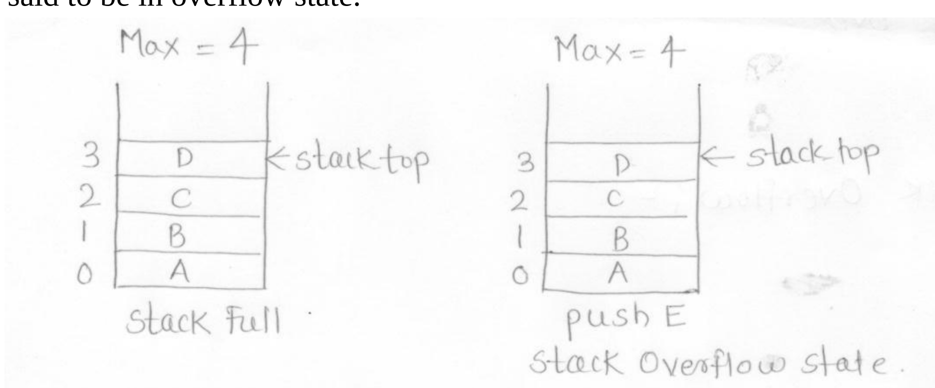
Q.6. Attempt any TWO: 16

a) Describe the term stack overflow and stack underflow with example. State any four applications of stack.

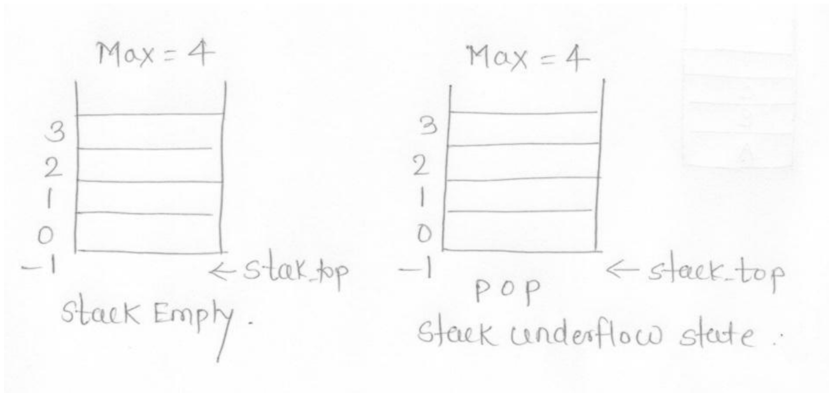
(Description of stack overflow 3M, underflow 3M, any four applications each ½ M)

Ans.

Stack overflow: When a stack is full and push operation is called to insert a new element, stack is said to be in overflow state.



Stack underflow: When there is no element in a stack (stack empty) and pop operation is called then stack is said to be in underflow state.



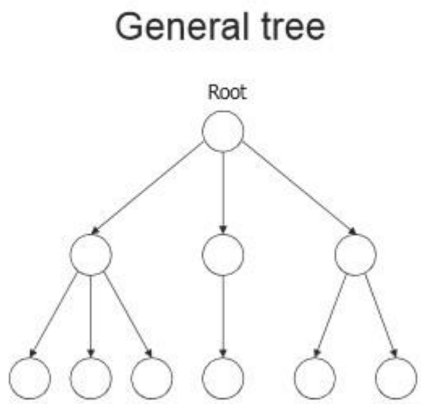
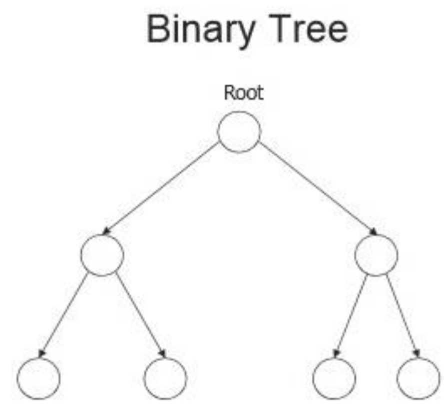
Applications for Stack:-

1. Recursion
2. Polish Notation Conversion
3. Expression Evaluation
4. Reversal of List

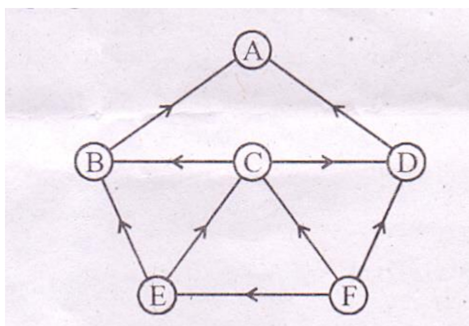
b) Compare between General Tree and Binary Tree (any 4 points)
(Any four relevant points: 2M each)

Ans:-

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

6	Example:- 	Example:- 

c) Consider the graph given below:



(i) Give adjacency matrix representation.
(ii) Give adjacency list representation of graph.
(Correct Adjacency Matrix 4M, adjacency list 4M)

Ans.

i) Adjacency matrix

$$A_{ij} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

ii) Adjacency list representation

Nodes	**Adjacency List**
A	-
B	A
C	B, D
D	A
E	B, C
F	E, C, D