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

SUMMER-15 EXAMINATION
Model Answer

Subject Code: 17330 **Subject Name: Data Structure Using 'C'**

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 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.

1. A) Attempt any six: **MARKS 12**

a) Define primitive and non-primitive data structure.
(Each type definition carries- 1Mark)

Ans: PRIMITIVE DATASTRUCTURE

the primitive data structures are the basic data types that are available in most of the programming languages. The primitive data structures are used to represent single values.

Example: Integer, character, string, Boolean

NON-PRIMITIVE DATASTRUCTURE

The data structure that are derived from primary data structure is known as non-Primitive data structure. These data types are used to store group of values.

Example: Arrays, Structure, Union, linked list, Stacks, Queue etc.

b) Enlist the operations on data structure.
(2 operation carries- 1 Mark)

Ans:

1. Traversing
2. Searching
3. Sorting
4. Insertion
5. Deletion
6. Merging.

c) Define sorting and give its type.
(Sorting definition -1Mark, Types-1Mark)

Ans: Sorting is the operation of arranging data or elements in some given order, numerically or alphabetically.

The sorting is classified into two categories.

1. **Internal Sorting:** In this sorting technique, all the data is retained in main memory only and the data can be accessed randomly.
2. **External Sorting:** In this sorting, the data to be sorted is moved from secondary storage to main memory. Because the large data may be in secondary storage. It is more time consuming to move records into different location.

d) Give the principal of bubble sort.
(Principal-2 Marks)

Ans: The algorithm in bubble sort involves two steps, executed over & over until data is sorted.

1. Compares two adjacent items
2. If necessary , swap (exchange) them

Sorting begins with 0^{th} elements & it compares with 1^{st} element in list. If 1^{st} element is less than 0^{th} element then interchange takes place. Like this all elements are compared with next element & interchanged if required. At end of 1^{st} pass largest element is placed at last position. In 2^{nd} pass again comparisons starts with 0^{th} element & 2^{nd} largest element is placed at second last position. This process continues till list is in sorted order.

e) Define stack.
(Definition-2Marks)

Ans: A stack is linear data structure or orderly collection of data in which data may be inserted or deleted from one end called 'top of stack' for this reason stack is referred as LIFO(Last in First Out).

f) List operations on trees.
(Each 2 operation-1Mark)

Ans:

1) creation
2) traversal
3) deletion
4) insertion
5) compare
6) merge

g) What is Binary tree?
(Definition-2 Marks)

Ans: A binary tree is a tree data structure in which each node has at most two children, which are referred to as the *left* child and the *right* child

h) Explain term weighted graph.
(Definition-2Marks)

Ans: A graph is a weighted graph if a number (weight) is assigned to each edge. Such weights might represent costs, lengths or capacities etc.

B) Attempt any two: **MARKS 8**

a) Explain different approaches to design an algorithm.
(Each approach-2Marks)

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

b) Explain the linear search algorithm. Also give its limitations.
(Algorithm-1 Mark, Explanation-2 Marks, Limitation-1Mark; any related algorithm can be considered)

Ans: Linear search is the easiest search technique in which each element is scanned in a sequential manner to locate the desire element.

In this method, search begins with the first available record and proceed to the next available record until we find the target key or conclude that it is not found. Linear search is also called as ‘sequential search’.
Example: Consider an array with 5 elements

3	21	11	91	19
0	1	2	3	4

- We are searching for element 91. To search 91 the item 91 is compared with element at A[0] then A[1] and so on. Until we find the target value or reach to the end of array.
- When item is found it displays the location of an item else displays item not found.
- **Algorithm:**

1. Start
2. Accept n values from user i.e. array elements.
3. Accept element to be searched from user i.e. target.
4. Set i=0,flag=0
5. Compare A[i] with target
   If(A[i] is a target)

Set flag=1 go to step 7
Else
Move to next data element
i= i+1;
6. If (i<=n) go to step 5
7. If(flag=1) then
Return i as position of target located
Else
Report as target not found.
8. Stop.

Limitation: Highly inefficient for large data

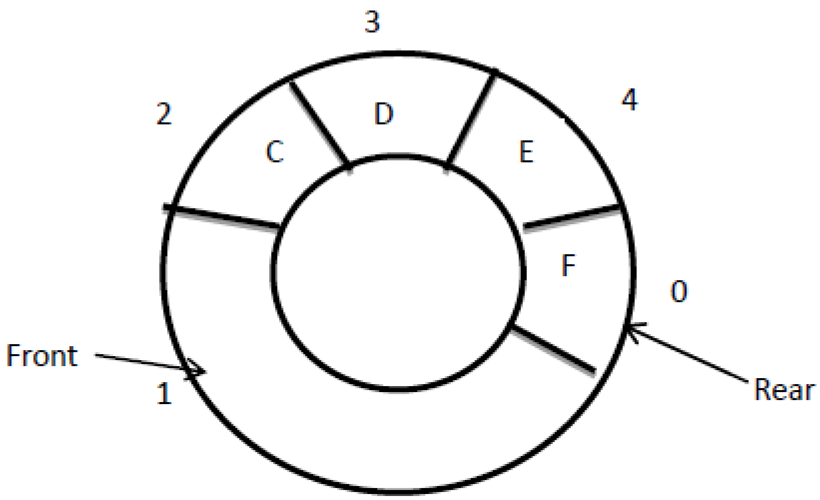
c) Draw and explain circular queue.
(Definition-1Mark, Explanation-3Marks)

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

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 0th position, if it is empty, and continue incrementing rear in same manner as earlier. Thus rear will have to be incremented circularly. For deletion, front will also have to be incremented circularly. Rear can be incremented circularly by the following code.

```
If ((rear == MAX-1) and (front !=0)
        Rear =0;
Else
        Rear= rear +1;
```

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 since initially 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.

2. Attempt any four: **MARKS 16**

a) Sort the given no. in ascending order using radix sort.
Numbers: 348, 14, 614, 5381, 47
(Each pass- 1 Mark)

Ans: 348, 14,614,5381,47

Pass 1: Numbers are sorted on least significant digit. Numbers with the same least significant digit are sorted in the same bucket

Input	Bucket									
	0	1	2	3	4	5	6	7	8	9
348									348	
14					14					
614					614					
5381	5381									
47								47		

Output of Pass 1:- 5381, 14, 614, 47, 348

Pass 2: Numbers are sorted on the second least significant digit

Input	Bucket									
	0	1	2	3	4	5	6	7	8	9
5381									5381	
14		14								
614		614								
47					47					
348					348					

Output of Pass 2:- 14, 614, 47, 348, 5381

Pass3: Numbers are sorted on the third least significant digit

Input	Bucket									
	0	1	2	3	4	5	6	7	8	9
14	14									
614							614			
47	47									
348				348						
5381				5381						

Output of Pass 3:- 14, 47, 348, 5381,614

Pass4: Numbers are sorted on the fourth least significant digit

Input	Bucket									
	0	1	2	3	4	5	6	7	8	9
14	14									
47	47									
348	348									
5381						5381				
614	614									

Output of Pass 4:- 14, 47, 348, 614, 5381

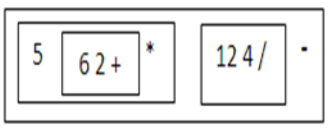
The sorted elements are14, 47, 348, 614, 5341

b) Convert the following arithmetic expression P written in postfix notation into infix.
P: 5, 6, 2, +, *, 12, 4, /, -
also evaluate P for final value.
(Each 2 step carries -1 Mark)

Ans: Postfix to infix form:

Postfix expression (P): 5, 6, 2, +,*, 12, 4, /,-

Step 1: Grouping of tokens as per the sequence of evaluation.



Step 2: Operators are moved between the two operands of the group

5*(6+2)-12/4

	Input	Stack	Operation
1	5,6,2,+,*,12,4,/,-	Empty	
2	6,2,+,*,12,4,/,-	5	
3	2,+,*,12,4,/,-	5,6	
4	+,*,12,4,/,-	5,6,2	
5	*,12,4,/,-	5,8	6+2=8
6	12,4,/,-	40	5*8=40
7	4,/,-	40,12	
8	/,-	40,12,4	
9	-	40,3	12/4=3
10	End	37	40-3=37

c) Define the term:
i) Node
ii) Address
iii) Null pointer
iv) Next pointer for linked list.
(Each definition -1 Mark)

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

ii) Address: Address indicates location of node

iii) Null pointer: A linked field of the last node contains Null rather than a valid address it indicates end of the list

iv) Next Pointer for Linked list: Which Hold the Address of the next node, which intern links the nodes together.

d) Explain the terms with the help of diagram:
i) Siblings
ii) Leaf Node
(Definition-1Mark, Example-1 Mark)

Ans:

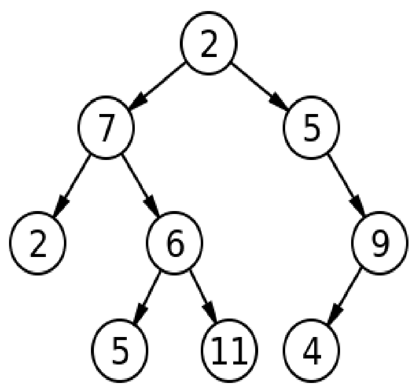
i) Siblings: Any nodes that have the same parent.

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

In below diagram node 7&5 are siblings because they have same parent 2

Other sibling pairs are 2&6(parent 7), 5&11(parent 6)

2, 4,5,11 are leaf nodes because they don't have any children



e) Distinguish between stack and queue with minimum 4 points.
(4 points each -1 Mark)

Ans:

Stack	**Queue**
In Stack insertion and deletion operations are performed at same end	In Queue insertion and deletion operations are performed at different end
In stack the element which is inserted last is first to delete so it is called Last In first Out	In Queue the element which is inserted first is first to delete so it is called first In first Out
In stack only one pointer is used called Top	In Queue two pointer is used called front, rear
In Stack Memory is not wasted	In Queue memory can be wasted/ unusable in case of linear queue.
In computer system stack is for procedure calls	Queue is used for time/resource sharing system;
Plate Counter at marriage reception is an example of stack	students standing in a line at fee counter is an example of queue

f) Write a 'C' program for the selection sort.
(Program-2 Mark, Logic -2Mark, 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++)
  {
            pos=i;
                    for(j=i+1;j<n;j++)
                    {
                             if(array[j]<array[pos])
                            pos=j;
                    }
            temp=array[i];
            array[i]=array[pos];
            array[pos]=temp;

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

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

a) Define recursion. Write a 'C' program for multiplication of natural numbers using recursion.
(Definition -1Mark, Logic of program- 3Marks)

Ans: Definition: - Process of calling function itself is known as **recursion in C** programming.
Program:-

```
#include<stdio.h>
int multiply(int,int);
int main(){
   int a,b,product;
   printf("Enter any two integers: ");
   scanf("%d%d",&a,&b);
   product = multiply(a,b);
   printf("Multiplication of two integers is %d",product);

   return 0;
}
```

```
int multiply(int a,int b){
   static int product=0,i=0;
   if(i < a){
       product = product + b;
       i++;
       multiply(a,b);
   }
   return product;
}
```

b) Describe priority queue and list its advantages.
(Description -2Marks, any two advantages- 2Marks)

[Note: array representation or linked representation shall be considered]

Ans: Description:-

A priority Queue is a collection of elements where each element is assigned a priority and the order in which elements are deleted and processed is determined from the following rules:

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

Advantages:-

- Preferences to the higher priority process are added at the beginning.
- Keep the list sorted in *increasing* order.

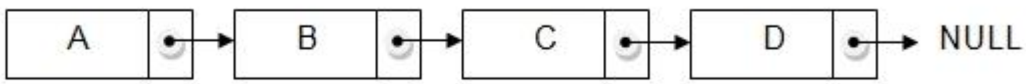
c) With suitable diagram, explain 'searching' of a node in a linked list.
(Explanation -2Marks, example/diagram- 2Marks)

Ans:

Algorithm

Step-1: Initialize the Current pointer with the beginning of the List.
Step-2: Compare the KEY value with the Current node value;
if they match then quit there
else go to step-3.
Step-3: Move the Current pointer to point to the next node in the list and go to step-2, till the list is not over or else quit.

Example:-



Single Linked List

d) Explain inorder, preorder and postorder traversal.
(Explanation of each traversal -1Marks, example with all traversal -1Marks)

Ans:

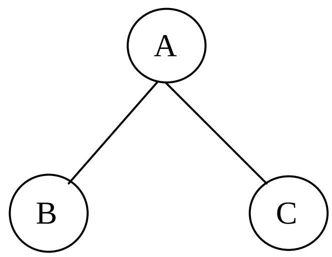
1. Preorder traversal:-In this traversal method first process root element, then left sub tree and then right sub tree.
   Procedure:-
   Step 1: Visit root node
   Step 2: Visit left sub tree in preorder
   Step 3: Visit right sub tree in preorder

2. Inorder traversal:-
   In this traversal method first process left element, then root element and then the right element.
   Procedure:-
   Step 1: Visit left sub tree in inorder
   Step 2: Visit root node
   Step 3: Visit right sub tree in inorder

3. Postorder traversal:-
   In this traversal first visit / process left sub tree, then right sub tree and then the root element.
   Procedure:-
   Step 1: Visit left sub tree in postorder
   Step 2: Visit right sub tree in postorder
   Step 3: Visit root node

Example:-

Preorder: A, B, C
Inorder: B, A, C
Postorder: B, C, A



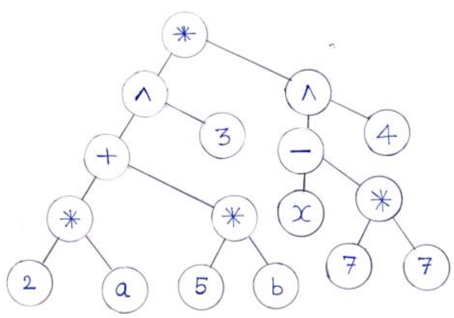
e) Draw the tree structure for the following expressions:

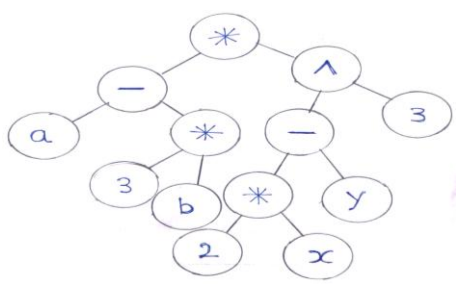
i) **$(2a+5b)^3 (x-7y)^4$**

ii) **$(a-3b)(2x-y)^3$**

(Each equation tree- 2Marks)

Ans:





f) What is Hashing? Give its significance.

(Definition of hashing -2Marks, Significance- 2Marks)

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

Significance of Hashing:-

1. Fast retrieval of information.
2. Element can be placed at appropriate position.
3. Collision can be avoided.

4. Attempt any four: **MARKS 16**

a) Explain time and space complexity with example.
(Explanation with example each complexity - 2Marks)

Ans: Time complexity:-

Time complexity of a program/algorithm is the amount of computer time that it needs to run to completion.

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 space requirements: - It consists of space needed by structured variables whose size depends on particular instance of variables.

Example: - additional space required when function uses recursion.

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

Ans: **Stack As An Abstract Data Type:**

- Representation of stack as an abstract data type is very straightforward.
- We use eltype to denote type of stack element & parameterize stack type with eltype abstract typedef <<eltype>> stack (eltype).
- C implementation of stack: A stack , s of integer data type can be defined as:
```
int s[10];
int top =-1;
```
  Where s is an array of integer type which can hold integer values & top is variable used to hold top position of list.
- Formally stack may be defined as follows:
```
typedef struct stack
{
        int data [size];
        int top;
        }stack;
```
  size is constant, maximum number of elements that can be stored

c) Write a procedure to insert an element into a queue and to delete an element from a queue.
(Insert procedure- 2Marks, Delete procedure- 2Marks)

Ans: Procedure for Insert:-

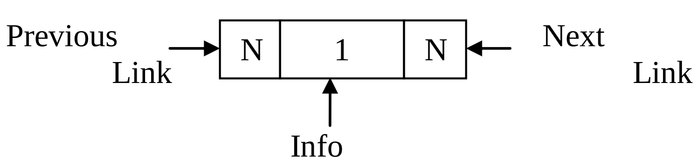
Step 1: [check overflow]
 if rear = max -1 then write
 "queue is full"
 return
 otherwise go to step 2
Step 2: [increment rear point]
 rear = rear + 1
Step 3: [insert element]
 a [rear] = item
Step 4: [check front pointer]
 if front = -1 then assign 0 value to front
Step 5: End / return to calling function

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

d) Describe doubly linked list with its node structure.
(Description- 2Marks, Node structure- 2Marks)

Ans: Description:-
Doubly linked list is also called as two way linked list. The advantage of doubly linked list is that it allows traversal in both the direction throughout the list.
Each node contains 3 fields:-
• Info field is used to store data.
• A Pointer field next which contains pointer to next node.
• A pointer field previous which contains pointer to previous node

Node structure:-

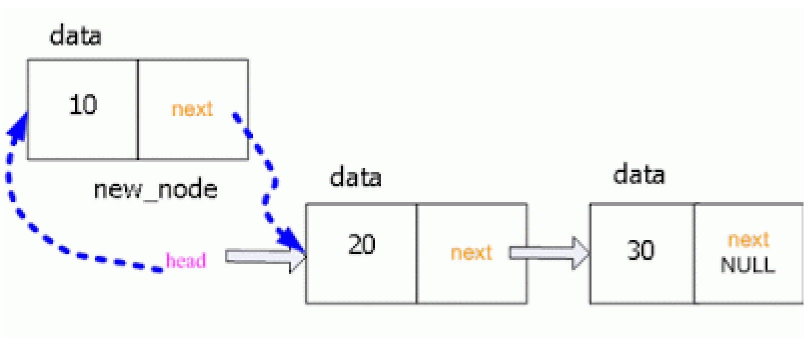


struct node
{
int info;
struct node * node;
struct node * prev;
} *start;

e) Explain insertion of new node at start and at end in singly linked list.
(Explanation of insertion at start- 2Marks, At end- 2Marks)

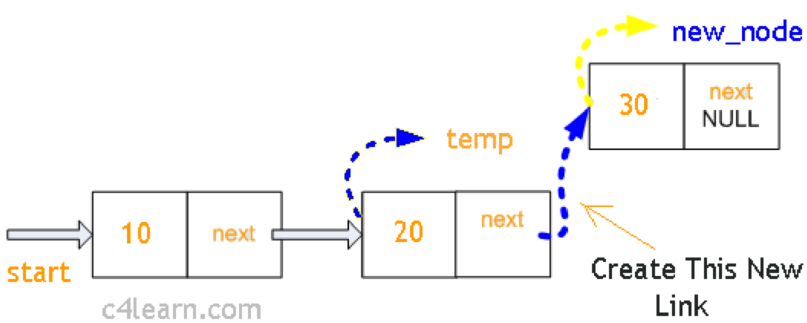
Ans: Inserting node at start in the SLL (Steps):

1. Create New Node
2. Fill Data into "Data Field"
3. Make it's "Pointer" or "Next Field" as NULL
4. Attach This newly Created node to Start
5. Make new node as Starting node



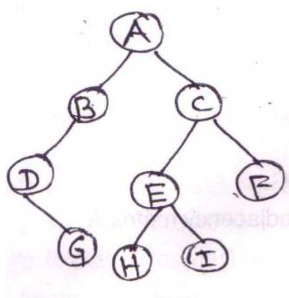
Inserting node at last in the SLL (Steps):

1. Create New Node
2. Fill Data into "Data Field"
3. Make it's "Pointer" or "Next Field" as NULL
4. Node is to be inserted at Last Position so we need to traverse SLL up to Last Node.
5. Make link between last node and new node



f) Consider the following tree. Write its:
- **a) In-order traversal sequence**
- **b) Pre-order traversal sequence**
- **c) Post-order traversal sequence.**

(Inorder -1 Mark; Pre order- 1 Marks; Postorder- 2 Marks)



(fig. Q. 4f)

Ans: Inorder-D G B A H E I C F
Preorder:- A B D G C E H I F
Postorder:- G D B H I E F C A

5. Attempt ant two: **MARKS 16**

a) Find the position of element 29 using binary search method in an array 'A' given below:
A= {11, 5, 21, 3, 29, 17, 2, 43}
Write a 'C' program for binary search.
(Solution-4 Marks Program-4 Marks)

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

Pre-condition for Binary search is array elements must be sorted in ascending order.

In given example array elements are not sorted. Applying Bubble sort we sort the elements of array.

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

i. Low=0 high=7 k=29

Mid= (0+7)/2 = 3

A[mid] =a [3] =11
29>11
k>a [mid]
ii. Low=mid+1 High=7
Mid= (4+7)/2=5
A[mid] =a [5] =21
29>21
k>a [mid]
iii. Low=mid+1
Mid= (6+7)/2=6
A[mid] =a [6] =29
A[mid] =k
Therefore key element is found at 6th position, no. of comparison required = 3. Search is successful

'C' program for Binary Search.

```
 #include<stdio.h>
 #include<conio.h>
void main()
{ int a[12];
int e,n,mid,i;
int l; int r;
int flag=0;
 int count=0;
char ch='y';
 clrscr();
printf("\n **** BINARY SEARCH METHOD *****");
 printf("\n Enter The Number Of Elements You Want In The Array: 1 To 12 \n ");
scanf("%d",&n);
printf("Enter the elements of the array\n");
for(i=0;i<n;i++)
{ scanf("%d",&a[i]); }
do { printf("\nEnter The Key Element To Be Searched:");
scanf("%d",&e);
l=0;
```

```
r=(n-1);
flag=0;
count=0;
 while(l<=r)
{ mid=(l+r)/2;
count++;
 if(e==a[mid])
{ printf("\nThe element %d is found at: %d",e,mid);
 flag=1; break; }
else { if(e>a[mid]) { l=(mid+1); }
 else { r=(mid-1); } }}
 if(flag!=1)
 { printf("\nThe element is not found!"); }
printf("\nThe no of comparisons:%d",count);
printf("\n Do You Want To Continue : (Yes/No)");
fflush(stdin);
scanf("%c",&ch); }
while(ch == 'y' || ch == 'Y');
getch(); }
```

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

b) Convert the given infix expression to postfix expression using stack and the details of stack at each step of conversion
Expression: a↑ b*c-d+e/f/ (g+h)
Note: ↑ indicates exponent operator.
(Correct Postfix Expression -8 Marks; Stepwise solution shall be consider)

Ans:

Sr.no.	Details of Stack	Input	Output(postfix)
1	Empty	a↑ b* c - d + e / f /(g + h)	-
2	Empty	↑ b* c - d + e / f /(g+ h)	a
3	↑	b* c - d + e / f /(g+ h)	a
4	↑	* c - d + e / f /(g+ h)	ab
5	*	c - d + e / f /(g+ h)	ab↑
6	*	- d + e / f /(g+ h)	ab↑c*
7	-	d + e / f /(g+ h)	ab↑c*
8	-	+ e / f /(g+ h)	ab↑c*d
9	+	e / f /(g+ h)	ab↑c*d-
10	+	/ f /(g+ h)	ab↑c*d-e
11	+,/	f/(g+h)	ab↑c*d-ef
12	/	/(g+h)	ab↑c*d-ef/
13	+,/	(g+h)	ab↑c*d-ef/
14	+,/ ,(	g+h)	ab↑c*d-ef/g
15	+,/ ,(	+h)	ab↑c*d-ef/g
16	+,/,+	h)	ab↑c*d-ef/g
17	+,/	)	ab↑c*d-ef/gh+
18	+,/,)		ab↑c*d-ef/gh+
19	+,/	/	ab↑c*d-ef/gh+/
20	+		ab↑c*d-ef/gh+/
21	Empty		ab↑c*d-ef/gh+/+

Resultant Postfix Expression is- ab↑c*d-ef/gh+/+

c) Explain DFs with suitable example.
(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.

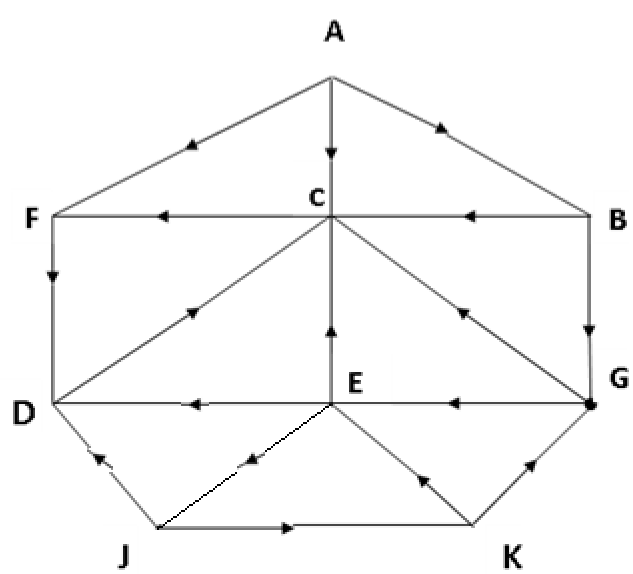
Step4: Pop the top node from stack and mark is as visited. Push all its adjacent nodes into the stack &mark them as waiting.

Step 5 .Repeat step 4 until stack 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

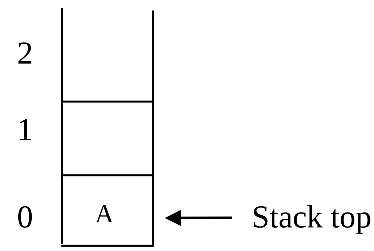
These are the nodes reachable from **J.**

6. Attempt any two: **MARKS 16**

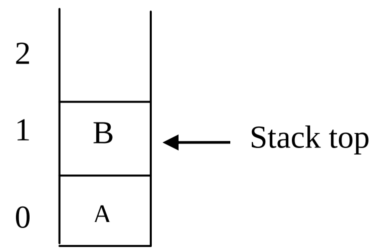
a) Explain push and POP operation on stack with algorithm and example.
(Push Algorithm 2 Marks, Example 2 Marks, POP Algorithm 2 Marks, Example 2 Marks)

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

PUSH



PUSH B



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.

Algorithm

1. If TOP = SIZE – 1, then:

(a) Display "The stack is in overflow condition"

(b) Exit

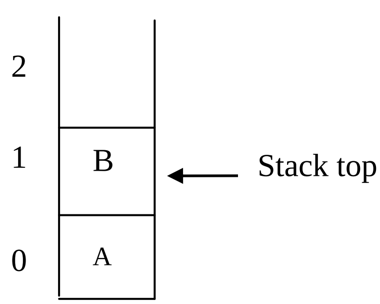
2. TOP = TOP + 1

3. STACK [TOP] = ITEM

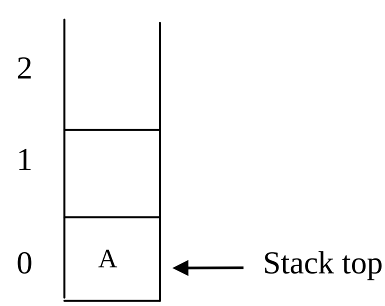
4. Exit

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

POP B



POP A



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

Algorithm:-

1. If TOP < 0, then

(a) Display "The Stack is empty"

(b) Exit

2. Else remove the Top most elements

3. DATA = STACK [TOP]

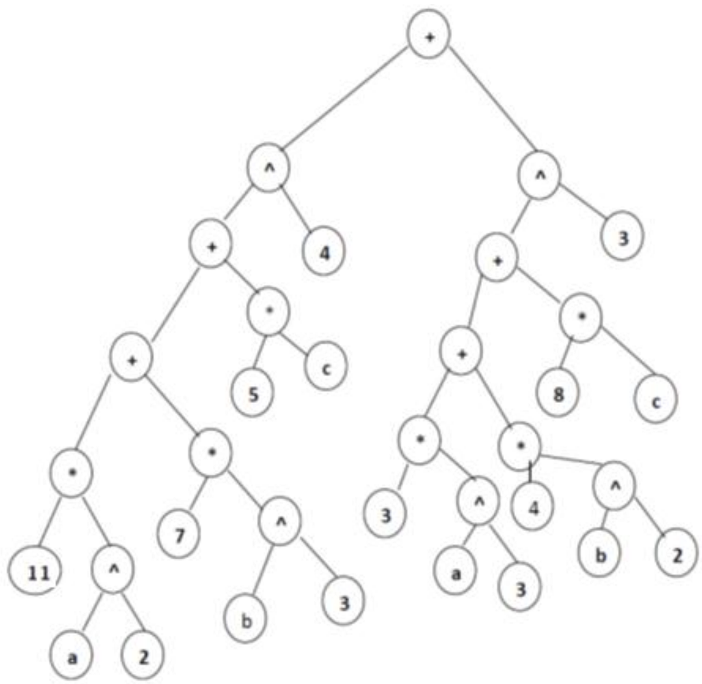
4. TOP = TOP – 1

5. Exit.

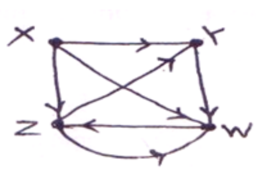
b) What is tree? Define any four terminologies related to tree and draw the tree structure for following expression
$(11a^2 +7b^3+5c)^4 + (3a^3+4b^2+8c)^{3.}$
(Definition- 2 Mark, Four Terminology-2 Marks, Tree structure- 4Marks)

Ans: **Tree** is non-linear data structure with finite number of nodes in which there is one special node called root node and remaining nodes are divided into disjoint set which form the sub trees of the main tree.

(Any four terminology)
Root: It is special node in tree structure & the entire tree is referred through it.
Parent: It is an immediate predecessor of node.
Child: All immediate successors of node are its children.
Siblings: Nodes with the same parent are siblings.
Degree of node: The degree of node is number children of a node.
Leaf node: A node of degree 0 is known as leaf node.
Level: The level of node in binary tree is the root of tree has level 0. The level of any other node in the tree is one more than level of its father.
Depth: The depth of binary tree is the maximum level of any node in the tree expression tree for
$(11a^2 +7b^3+5c)^4 + (3a^3+4b^2+8c)^3$



c) Consider the graph 'G' in fig.



i) Find all the simple paths from X to Z
ii) Find all the simple paths from Y to Z.
iii) Find indeg (Y) and outdeg (Y).
iv) Find the adjacency matrix A of the graph G.
v) Find the path P of G using powers of the adjacency matrix A.

(Path from X to Z-1Mark, Paths from Y to Z-1Mark, In deg- 1 Mark, Out deg-1 Mark, Adjacency matrix- 2 Marks, Path P of G using powers of the Adjacency matrix A-2 Marks)

Ans:

i) Simple path from x to z
   a) $X \rightarrow z$
   b) $x \rightarrow w \rightarrow z$
   c) $x \rightarrow y \rightarrow w \rightarrow z$

ii) Simple path from y to z
   $Y \rightarrow w \rightarrow z$

iii) Indeg (y) =2
   Outdeg(y) =1

iv) Adjacency matrix

A=

	w	x	y	z
w	0	0	0	1
x	1	0	1	1
y	1	0	0	0
z	1	0	1	0

v) path matrix
   $P=A+A^2+A^3+A^4$

$$A=\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

$$A^2=\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$$A^3=A^2*A=\begin{bmatrix}1&0&1&0\\1&0&1&1\\0&0&0&1\\1&0&0&1\end{bmatrix}*\begin{bmatrix}0&0&0&1\\1&0&1&1\\1&0&0&0\\1&0&1&0\end{bmatrix}=\begin{bmatrix}1&0&0&1\\1&0&1&1\\1&0&1&0\\1&0&1&1\end{bmatrix}$$

$$A^4=A^3*A=\begin{bmatrix}1&0&0&1\\1&0&1&1\\1&0&1&0\\1&0&1&1\end{bmatrix}*\begin{bmatrix}0&0&0&1\\1&0&1&1\\1&0&0&0\\1&0&1&0\end{bmatrix}=\begin{bmatrix}1&0&1&1\\1&0&1&1\\1&0&0&1\\1&0&1&1\end{bmatrix}$$

$P=A+A^2+A^3+A^4$

$$P=\begin{bmatrix}1&0&1&1\\1&0&1&1\\1&0&1&1\\1&0&1&1\end{bmatrix}$$