Data Structure and Algorithm (MCA 271)

ESE 3 –

BY

Himanshu Heda (24225013)

SUBMITTED TO

Prof. Vandna Kansal

SCHOOL OF SCIENCES

2024-2025

Program Description:

Code of the program

Output: - Paste the o/p of the program.

F4

Page No.

CHRIST
(DEEMED TO BE UNIVERSITY)
DELHI-NCR, INDIA

Name - Himanshu Heda Set - 2
Roll No. - 2422501 3 ESE-3
Date - 15-01-25

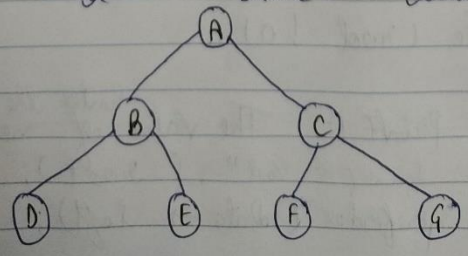
Q.1)

Tree -

→ It is a non-linear DataStructure.
→ It is basically present in the form of a tree like structure which contains root node, head node, leaf node which will define the relationships among the nodes like the parent child relationship.

Tree Traversal :-

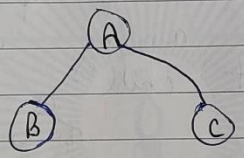
→ It is a traversal which is performed on a tree we will arrange the tree's data in an order with the help of tree traversal.
→ In tree traversal we will traverse the data from the root node, left node or right node to sort any data which is present in a tree like structure.



-> Basically we can perform tree traversing with 3 methods That are :—

1.) Pre order :-

-> It is use to traverse from the left to the head then to ther righ side.



Algo. :-

Step 1 :- write while root =! 0
Step 2 :- Tree -> Data
Step 3 :- Preorder (Data -> left)
Step 4 :- Preorder (Data -> right)
Step 5 :- Preorder (Data -> root)
Step 6 :- close while

Code :-

```
while (root =! 0)
{
    printf (" Enter the value : ");
    scanf (" %d ", &root);
    preorder ( Data -> left);
```

```
Preorder (Data → Right);
preorder (Data → Root);
}
```

2) Post order :-

-> It will traverse from Right to the left and from left to the head node.

-> It will first traverse from Head to the left then to the right.

Algo :-

Step 1 :- While root != 0
Step 2 :- write Tree -> Data
Step 3 :- Postorder (Data -> root)
Step 4 :- Postorder (Data -> left)
Step 5 :- Postorder (Data -> right)
Step 6 :- Close while

Code :-

```
while (root != 0)
{
    printf("Enter the value : ");
    scanf("%d", &root);
    Postorder (Data -> root);
    postorder (Data -> left);
    postorder (Data -> right);
}
```

3) ~~Infix~~ : Inorder :-

→ In this fix first we will go to the left node then we will go to the root node the ~~root~~ we will traverse to the right node.

Algo :-

Step 1 :- while (root =! 0)
Step 2 :- if (~~root~~ Data =0)
~~Step 3 :-~~ {
Step ~~3~~ 4 :- Inorder (Data → head)
Step 4 :- Close if
Step 5 :- else
Step 6 :- Inorder (Data → left)
Step 7 :- Inorder (Data → right)
Step 8 :- Close else
Step 9 :- Close while

Code :-

```
while ( root =! 0){
if (Data =0)
{
    Inorder (Data → head);
}
else
{
    Inorder (Data → left);
    Inorder (Data → right);
}
}
```

Tree Traversal : --

```c
#include <stdio.h>
#include <stdlib.h>

// Define the structure for a binary tree node
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

// Function to create a new node
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

// Function to perform In-order traversal
void inOrderTraversal(struct Node* root) {
    if (root == NULL) return;
    inOrderTraversal(root->left);
    printf("%d ", root->data);
    inOrderTraversal(root->right);
}

// Function to perform Pre-order traversal
void preOrderTraversal(struct Node* root) {
    if (root == NULL) return;
    printf("%d ", root->data);
    preOrderTraversal(root->left);
    preOrderTraversal(root->right);
}

// Function to perform Post-order traversal
void postOrderTraversal(struct Node* root) {
```

```
    if (root == NULL) return;
    postOrderTraversal(root->left);
    postOrderTraversal(root->right);
    printf("%d ", root->data);
}

int main() {
    // Creating a sample binary tree
    struct Node* root = createNode(1);
    root->left = createNode(2);
    root->right = createNode(3);
    root->left->left = createNode(4);
    root->left->right = createNode(5);
    root->right->left = createNode(6);
    root->right->right = createNode(7);

    // Performing In-order traversal
    printf("In-order Traversal: ");
    inOrderTraversal(root);
    printf("\n");

    // Performing Pre-order traversal
    printf("Pre-order Traversal: ");
    preOrderTraversal(root);
    printf("\n");

    // Performing Post-order traversal
    printf("Post-order Traversal: ");
    postOrderTraversal(root);
    printf("\n");

    return 0;
}
```

OUTPUT : --

```
[Running] cd "d:\2MCA\DSA\" && gcc Tree_Traversal.c -o T
In-order Traversal: 4 2 5 1 6 3 7
Pre-order Traversal: 1 2 4 5 3 6 7
Post-order Traversal: 4 5 2 6 7 3 1

[Done] exited with code=0 in 1.525 seconds
```