Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ19ECSBL1)

Name: Nihal Nisar Shaikh SAP ID: 60002200155 Batch:E1-4

Experiment no: 8 Date: 1-12-2022

Aim: Write a program to create a binary tree and traverse it in in order, pre order and post order

Programming Language: C/C++/Java

Theory: A Tree is a non-linear data structure and a hierarchy consisting of a collection of nodes such that each node of the tree stores a value and a list of references to other nodes called as its children.

Creating tree structure:

Each node in a linked list consists of three fields INFO, LEFT, and RIGHT pointer

- LEFT [k] contains the location of the left child of node N.
- INFO [k] contains the data at the node N.
- RIGHT [k] contains the location of right child of node N.

Tree traversal: Traversing a tree means visiting and outputting the value of each node in a particular order. The major importance of tree traversal is that there are multiple ways of carrying out traversal operations unlike linear data structures like arrays, bitmaps, matrices where traversal is done in a linear order. Each of these methods of traversing a tree have a particular order they follow:

- For Inorder, traverse from the left subtree to the root then to the right subtree.
- For Preorder, traverse from the root to the left subtree then to the right subtree.
- For Post order,traverse from the left subtree to the right subtree then to the root.

Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ19ECSBL1)

Name: Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

Code:

```
#include <stdlib.h>
struct node
{
    int item;
    struct node *left;
    struct node *right;
};
void preorderTraversal(struct node *root)
{
    if (root == NULL)
        return;
    printf("%d ->", root->item);
    preorderTraversal(root->left);
    preorderTraversal(root->right);
}
// Inorder traversal
void inorderTraversal(struct node *root)
{
    if (root == NULL)
        return;
    inorderTraversal(root->left);
    printf("%d ->", root->item);
    inorderTraversal(root->right);
}
// preorderTraversal traversal
// postorderTraversal traversal
void postorderTraversal(struct node *root)
{
    if (root == NULL)
        return;
    postorderTraversal(root->left);
    postorderTraversal(root->right);
    printf("%d ->", root->item);
}
// Create a new Node
struct node *createNode(value)
{
    struct node *newNode = malloc(sizeof(struct node));
    newNode->item = value;
    newNode->left = NULL;
    newNode->right = NULL;
```

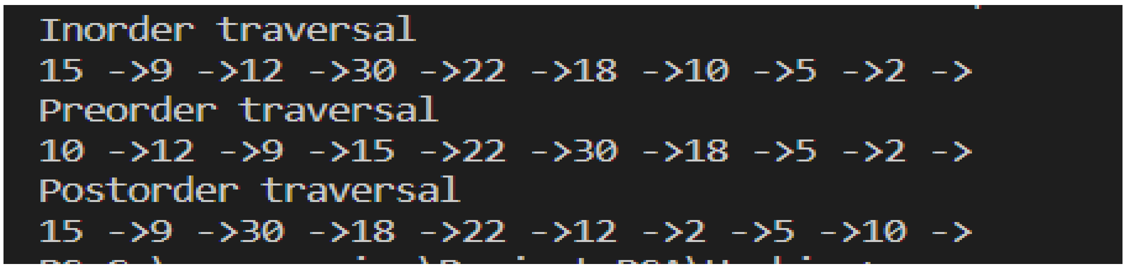
```c
    return newNode;
}
// Insert on the left of the node
struct node *insertLeft(struct node *root, int value)
{
    root->left = createNode(value);
    return root->left;
}
// Insert on the right of the node
struct node *insertRight(struct node *root, int value)
{
    root->right = createNode(value);
    return root->right;
}
int main()
{
    struct node *root = createNode(10);
    insertLeft(root, 12);
    insertRight(root, 5);
    insertLeft(root->left, 9);
    insertRight(root->left, 22);
    insertLeft(root->left->left, 15);
    insertLeft(root->left->right, 30);
    insertRight(root->left->right, 18);
    insertRight(root->right, 2);

    printf("Inorder traversal \n");
    inorderTraversal(root);
    printf("\nPreorder traversal \n");
    preorderTraversal(root);
    printf("\nPostorder traversal \n");
    postorderTraversal(root);
}
```

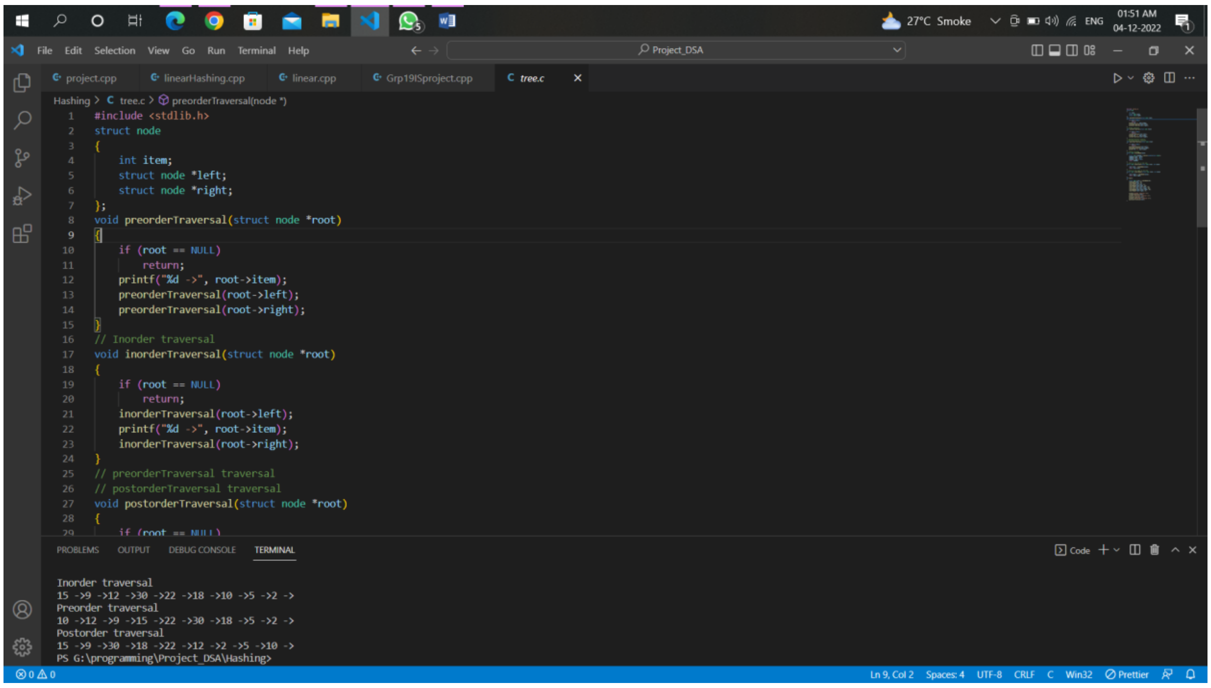
Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ19ECSBL1)

Name: Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

Output:



```
Inorder traversal
15 ->9 ->12 ->30 ->22 ->18 ->10 ->5 ->2 ->
Preorder traversal
10 ->12 ->9 ->15 ->22 ->30 ->18 ->5 ->2 ->
Postorder traversal
15 ->9 ->30 ->18 ->22 ->12 ->2 ->5 ->10 ->
```



```c
#include <stdlib.h>
struct node
{
    int item;
    struct node *left;
    struct node *right;
};
void preorderTraversal(struct node *root)
{
    if (root == NULL)
        return;
    printf("%d ->", root->item);
    preorderTraversal(root->left);
    preorderTraversal(root->right);
}
// Inorder traversal
void inorderTraversal(struct node *root)
{
    if (root == NULL)
        return;
    inorderTraversal(root->left);
    printf("%d ->", root->item);
    inorderTraversal(root->right);
}
// preorderTraversal traversal
// postorderTraversal traversal
void postorderTraversal(struct node *root)
{
    if (root == NULL)
```

```
Inorder traversal
15 ->9 ->12 ->30 ->22 ->18 ->10 ->5 ->2 ->
Preorder traversal
10 ->12 ->9 ->15 ->22 ->30 ->18 ->5 ->2 ->
Postorder traversal
15 ->9 ->30 ->18 ->22 ->12 ->2 ->5 ->10 ->
PS G:\programming\Project_DSA\Hashing>
```

Name: Nihal Nisar Shaikh SAP ID: 60002200155 Batch:E1-4

Result and Conclusion:

Algorithm Inorder(tree)

1. Traverse the left subtree, i.e., call Inorder(left->subtree)
2. Visit the root.
3. Traverse the right subtree, i.e., call Inorder(right->subtree)

Algorithm Preorder(tree)

1. Visit the root.
2. Traverse the left subtree, i.e., call Preorder(left->subtree)
3. Traverse the right subtree, i.e., call Preorder(right->subtree)

Algorithm Postorder(tree)

1. Traverse the left subtree, i.e., call Postorder(left->subtree)
2. Traverse the right subtree, i.e., call Postorder(right->subtree)
3. Visit the root