**Name: Abdul Ghaffar Kalhoro**

**Reg# 194699**

**Class: BSCS-6C**

DATA STRUCTURES AND ALGORITHMS

**assignment-4**

**Task: traversel the Binary tree using recursive definition with:**

**Inorder, PreOrder and postorder**

**Preorder:**

The preorder algorithm work as:

1) Visit the root

2) Traverse the left subtree

3) Traverse the right subtree

**Inorder:**

The inorder algorithm work as:

1) Traverse the left subtree

2) Visit the root

3) Traverse the right subtree

**Postorder:**

The postorder algorithm work as:

1) Traverse the left subtree

2) Traverse the right subtree

3) Visit the root

**Source code**

/\*

Name: Abdul Ghaffar Kalhoro

reg # 194699

BSCS-6C

Printing Binary Tree with:

Preorder, Inorder, Postorder \*/

#include<iostream>

using namespace std;

struct Node {

int data;

struct Node \*left;

struct Node \*right;

};

// Function to Insert Node in a Binary Search Tree

Node\* Insert(Node \*root, int value) {

if (root == NULL) {

root = new Node();

root->data = value;

root->left = root->right = NULL;

}

else if (value <= root->data)

root->left = Insert(root->left, value);

else

root->right = Insert(root->right, value);

return root;

}

// Preorder display function

void Preorder\_func(struct Node \*root) {

// base condition

if (root == NULL) return;

cout<<root->data<<", ";//printing data

Preorder\_func(root->left); // use root->left as a parameter

Preorder\_func(root->right); // use root->right as a parameter

}

//Inorder display function

void Inorder\_func(Node \*root) {

//base condition

if (root == NULL) return;

Inorder\_func(root->left); // use root->left as a parameter of inorder function

cout << root->data << ", "; //print the data

Inorder\_func(root->right); // use root->right as a parameter of Inorder function

}

//Postorder display function

void Postorder\_func(Node \*root) {

if (root == NULL) return;

Postorder\_func(root->left); // use root->left as a parameter of postorder

Postorder\_func(root->right); // use root->right as a parameter of postorder

cout << root->data << ", "; // Print the data

}

//displaying the layout of the tree

void display(){

cout << endl << "The tree is given as: "<<endl;

cout << " 7"<<endl;

cout << " / \\"<<endl;

cout << " 1 9" << endl;

cout << " / \\ / \\" << endl;

cout << " 0 3 8 10" << endl;

cout << " / \\" << endl;

cout << "\t 2 5" << endl;

cout << "\t / \\ " << endl;

cout << "\t 4 6" << endl;

}

int main() {

Node\* root = NULL;

root = Insert(root, 7);

root = Insert(root, 1);

root = Insert(root, 0);

root = Insert(root, 3);

root = Insert(root, 2);

root = Insert(root, 5);

root = Insert(root, 4);

root = Insert(root, 6);

root = Insert(root, 9);

root = Insert(root, 8);

root = Insert(root, 10);

display();

cout <<endl<<endl<<"The printing orders of the given tree are"<< endl;

cout << endl<<"Preorder: ";

Preorder\_func(root);//calling the preorder function

cout << "\n";

cout << "Inorder: ";

Inorder\_func(root); //using inorder function

cout << "\n";

cout << "Postorder: ";

Postorder\_func(root); //postorder function calling

cout << "\n"; cout << "\n"; cout << "\n";

getchar();

getchar();

return 0;

}

**OUTPUT**

