**Assignment Number: 1**

**Problem Statement:** Program To Perform Operations On Binary Tree

**Name:** Vaishnavi Sachin Karanjawane

**Roll No:** 04

**Class:** Second Year

**Division:** C

**Code**

#include <iostream>

using namespace std;

//Creation Of Node

struct node

{

int data;

node \*left = NULL;

node \*right = NULL;

};

node \*newnode(int data)

{

struct node \*n = (struct node \*)malloc(sizeof(struct node));

n->data = data;

n->left = NULL;

n->right = NULL;

return n;

}

node \*createtree()

{

node \*root = newnode(20);

root->left = newnode(10);

root->right = newnode(30);

root->left->left = newnode(5);

cout << "\nTree Is:" << "\n\t\t\t" << root->data;

cout << "\n\t\t\t/\\";

cout << "\n\t\t " << root->left->data << " " << root->right->data;

cout << "\n\t\t /";

cout << "\n\t\t " << root->left->left->data;

return root;

}

void preorder(node \*root)

{

if (root != NULL)

{

cout << root->data << "\t";

preorder(root->left);

preorder(root->right);

}

}

void inorder(node \*root)

{

if(root!=NULL)

{

inorder(root->left);

cout << root->data << "\t";

inorder(root->right);

}

}

void postorder(node \*root)

{

if(root!=NULL)

{

postorder(root->left);

postorder(root->right);

cout << root->data << "\t";

}

}

int height(node \* root)

{

if (root==NULL)

return 0;

else

{

int lh,rh;

lh=height(root->left);

rh=height(root->right);

int tot=max(lh,rh)+1;

return tot;

}

}

void mirror(node \*root)

{

if(root==NULL)

return ;

else

{

node \*temp;

mirror(root->left);

mirror(root->right);

temp=root->left;

root->left=root->right;

root->right=temp;

}

}

void leafnode(node \*root)

{

if(root==NULL)

return;

if(root->left==NULL && root->right==NULL)

{

cout<<root->data<<"\t";

return ;

}

if(root->left)

leafnode(root->left);

if(root->right)

leafnode(root->right);

}

int countnode(node \* root)

{

int ln,rn;

if(root==NULL)

return 0;

ln=countnode(root->left);

rn=countnode(root->right);

return ln+rn+1;

}

int main()

{

int ch;

char ch1[10];

cout << "1. Creation Of New Tree \n2. Tree Traversal \n3. Height Of Tree \n4. Mirror Image Of Tree \n5. Print Leaf Nodes Of Tree\n6. Total Number Of Nodes";

node \*root = NULL;

do

{

cout << "\n\nEnter Your Choice: ";

cin >> ch;

switch (ch)

{

case 1:

root = createtree();

break;

case 2:

cout << "\nPreorder: \n";

preorder(root);

cout << "\n\nInorder: \n";

inorder(root);

cout << "\n\nPostorder: \n";

postorder(root);

break;

case 3:

cout<<"\n\nHeight Of Tree Is: "<<height(root);

break;

case 4:

mirror(root);

cout << "\nMirror Image Of Tree Is:" << "\n\t\t\t" << root->data;

cout << "\n\t\t\t/\\";

cout << "\n\t\t " << root->left->data << " " << root->right->data;

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

cout << "\n\t\t\t " << root->right->right->data;

break;

case 5:

cout<<"\nLeaf Nodes: ";

leafnode(root);

break;

case 6:

cout<<"\nTotal Number Of Nodes Are: "<<countnode(root);

break;

default:

cout<<"Invalid Choice!";

}

cout << "\n\nDo You Want To Continue? ";

cin >> ch1;

} while (strcmp(ch1,"Yes")==0||strcmp(ch1,"yes")==0);

return 0;

}

**Output**







