**Lab 10**

**Name: Nimra**

**Class: BESE-7B**

**Registration No. 199281**

**LAB TASKS**

#include<iostream>

# include <cstdlib>

using namespace std;

struct bstNode{

int data;

bstNode \*leftChild;

bstNode \*rightChild;

} \*root;

class BST {

public:

void insert(bstNode \*newNode, bstNode \*tree){

if (root == NULL){

root = new bstNode;

root->leftChild = NULL;

root->rightChild = NULL;

root->data = newNode->data;

cout << "Root Node is Added" << endl;

return;

}

if (newNode->data < tree->data)

{

if (tree->leftChild == NULL)

{

{

tree->leftChild = newNode;

(tree->leftChild)->leftChild = NULL;

(tree->leftChild)->rightChild = NULL;

cout << "Node is Added To Left Side" << endl;

return;

}

}

else

{

insert(newNode, tree->leftChild);

}

}

else if (newNode->data > tree->data)

{

if (tree->rightChild == NULL)

{

tree->rightChild = newNode;

(tree->rightChild)->leftChild = NULL;

(tree->rightChild)->rightChild = NULL;

cout << "Node is Added To Right Child" << endl;

return;

}

else

{

insert(newNode, tree->rightChild);

}

}

}

void insertModified(bstNode \*newNode, bstNode \*tree){

if (root == NULL){

root = new bstNode;

root->leftChild = NULL;

root->rightChild = NULL;

root->data = newNode->data;

cout << "Root Node is Added" << endl;

return;

}

if (newNode->data < tree->data)

{

if (tree->leftChild == NULL)

{

{

tree->leftChild = newNode;

(tree->leftChild)->leftChild = NULL;

(tree->leftChild)->rightChild = NULL;

cout << "Node is Added To Left Side" << endl;

return;

}

}

else

{

insert(newNode, tree->leftChild);

}

}

else if (newNode->data >= tree->data)

{

if (tree->rightChild == NULL)

{

tree->rightChild = newNode;

(tree->rightChild)->leftChild = NULL;

(tree->rightChild)->rightChild = NULL;

cout << "Node is Added To Right Child" << endl;

return;

}

else

{

insert(newNode, tree->rightChild);

}

}

}

void display(bstNode \*ptr) //inorderDisplay

{

if (ptr != NULL){

display(ptr->leftChild);

cout << ptr->data << ", ";

display(ptr->rightChild);

}

}

int find\_height(bstNode \*high)

{

if (high == NULL) {

return 0;

}

else

{

int leftHeight = find\_height(high->leftChild);

int rightHeight = find\_height(high->rightChild);

if (leftHeight > rightHeight) {

return leftHeight + 1;

}

else {

return rightHeight + 1;

}

}

}

int minValue(struct bstNode\* node) {

struct bstNode\* current = node;

while (current->leftChild != NULL) {

current = current->leftChild;

}

return(current->data);

}

};

int main()

{

BST bst;

bstNode \*temp;

int select;

while (1)

{

cout << "Operations on BST" << endl;

cout << "1.Insert Element " << endl;

cout << "2.Display" << endl;

cout << "3.Insert Element Modified" << endl;

cout << "4.Minimum Value" << endl;

cout << "5.Height" << endl;

cout << "6.Quit" << endl;

cout << "Enter your choice : ";

cin >> select;

switch (select)

{

case 1:

temp = new bstNode;

cout << "Enter the number to be inserted : ";

cin >> temp->data;

bst.insert(temp, root);

case 2:

cout << "Display BST:" << endl;

bst.display(root);

cout << endl;

break;

case 3:

temp = new bstNode;

cout << "Enter the number to be inserted : ";

cin >> temp->data;

bst.insertModified(temp,root);

case 4:

cout << "The Minimum Value is: " << bst.minValue(root) << endl;

break;

case 5:

int h;

h = bst.find\_height(root);

cout << "The height is: " << h << endl;

break;

case 6:

exit(1);

break;

default:

cout << "Cannot be selected" << endl;

}

}

return 0;

}