# include <iostream>

# include <cstdlib>

using namespace std;

class node

{

public:

int info;

struct node \*left;

struct node \*right;

}\*root;

class BST

{

public:

node \*root;

void insert(node \*,node \*);

void display(node \*, int);

int min(node \*);

int height(node \*);

void mirror(node \*);

void preorder(node \*);

void inorder(node \*);

void postorder(node \*);

void search(node \*,int);

BST()

{

root = NULL;

}

};

int main()

{

int choice, num;

BST bst;

node \*temp;

while (1)

{

cout<<"-----------------"<<endl;

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

cout<<"-----------------"<<endl;

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

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

cout<<"3.Min value find"<<endl;

cout<<"4.Height"<<endl;

cout<<"5.Mirror of node"<<endl;

cout<<"6.Preorder"<<endl;

cout<<"7.Inorder"<<endl;

cout<<"8.Postorder"<<endl;

cout<<"9.No. of nodes in longest path"<<endl;

cout<<"10.Search an element"<<endl;

cout<<"11.Quit"<<endl;

cout<<"Enter your choice : ";

cin>>choice;

switch(choice)

{

case 1:

temp = new node();

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

cin>>temp->info;

bst.insert(bst.root, temp);

break;

case 2:

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

bst.display(bst.root,1);

cout<<endl;

break;

case 3:

cout<<"Min value of tree"<<endl;

cout<<temp->info;

bst.min(bst.root);

cout<<endl;

break;

case 4:

int h;

h=bst.height(bst.root);

cout<<"Height of tree="<<h;

cout<<endl;

break;

case 5:

cout<<"Mirror";

bst.mirror(bst.root);

bst.display(bst.root,1);

break;

case 6:

cout<<" \n Display preorder Binary tree = ";

bst.preorder(bst.root);

cout<<endl;

break;

case 7:

cout<<" \n Display inorder Binary tree = ";

bst.inorder(bst.root);

cout<<endl;

break;

case 8:

cout<<" \n Display postorder Binary tree = ";

bst.postorder(bst.root);

cout<<endl;

break;

case 9:

int nodes;

nodes=bst.height(bst.root);

cout<<"No. of nodes in longest path from root is "<<nodes;

cout<<endl;

break;

case 10:

int searchdata;

cout<<"Enter the element to ne searched:";

cin>>searchdata;

bst.search(bst.root, searchdata);

cout<<endl;

break;

case 11:

exit(1);

default:

cout<<"Wrong choice"<<endl;

}

}

}

void BST::insert(node \*tree, node \*newnode)

{

if (root == NULL)

{

root = new node;

root->info = newnode->info;

root->left = NULL;

root->right = NULL;

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

return;

}

if (tree->info == newnode->info)

{

cout<<"Element already in the tree"<<endl;

return;

}

if (tree->info > newnode->info)

{

if (tree->left != NULL)

{

insert(tree->left, newnode);

}

else

{

tree->left = newnode;

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

(tree->left)->right = NULL;

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

return;

}

}

else

{

if (tree->right != NULL)

{

insert(tree->right, newnode);

}

else

{

tree->right = newnode;

(tree->right)->left = NULL;

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

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

return;

}

}

}

void BST::display(node \*ptr, int level)

{

int i;

if (ptr != NULL)

{

display(ptr->right, level+1);

cout<<endl;

if (ptr == root)

cout<<"Root->: ";

else

{

for (i = 0;i < level;i++)

cout<<" ";

}

cout<<ptr->info;

display(ptr->left, level+1);

}

}

int BST::min(node \*root)

{

node \*temp;

if(root==NULL)

{

cout<<"Tree is empty";

}

else

{

temp=root;

while(temp->left!=NULL)

{

temp=temp->left;

}

return(temp->info);

}

}

int BST::height(node \*root)

{

int htleft,htright;

if(root==NULL)

{

//cout<<"Tree is empty"<<endl;

return(0);

}

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

{

return(1);

}

htleft=height(root->left);

htright=height(root->right);

if(htright>=htleft)

{

return(htright+1);

}

else

{return(htleft+1);

}

}

void BST::mirror(node \*root)

{

node \*temp;

if(root!=NULL)

{

temp=root->left;

root->left=root->right;

root->right=temp;

mirror(root->left);

mirror(root->right);

}

}

void BST::preorder(node \*ptr)

{

if(ptr!=NULL)

{

cout<<ptr->info<<"\t";

preorder(ptr->left);

preorder(ptr->right);

cout<<endl;

}

}

void BST::inorder(node \*ptr)

{

if(ptr!=NULL)

{

inorder(ptr->left);

cout<<ptr->info<<"\t";

inorder(ptr->right);

cout<<endl;

}

}

void BST::postorder(node \*ptr)

{

if(ptr!=NULL)

{

postorder(ptr->left);

postorder(ptr->right);

cout<<ptr->info<<"\t";

cout<<endl;

}

}

void BST::search(node \*ptr, int searchdata)

{

if (ptr->info==searchdata)

{

cout<<"Element Found..."<<endl;

}

else if (ptr->info<searchdata && ptr->right!=NULL)

{

search(ptr->right, searchdata);

}

else if (ptr->info>searchdata && ptr->left!=NULL)

{

search(ptr->left, searchdata);

}

else

{

cout<<"Element not found..."<<endl;

}

}