#include <iostream>

using namespace std;

class Node

{

public:

int no;

Node \* left;

Node \* right;

Node \* temp;

Node()

{

left=NULL;

right=NULL;

}

Node(int no)

{

this->no=no;

left=NULL;

right=NULL;

}

};

class Dictionary

{

public:

Node \* root;

int count=0;

Dictionary()

{

root=NULL;

}

void createfunc();

Node\* insertfunc(Node \*,int);

void Inorder(Node \*);

void Leaf(Node \*);

int Tree\_Height(Node \*);

void Mirror(Node \*);

};

int Menu()

{

int k;

cout<<"PRESS"<<endl;

cout<<"1) To Add Elements"<<endl;

cout<<"2) Get Inorder"<<endl;

cout<<"3) Get Height of Tree"<<endl;

cout<<"4) Get Leaf"<<endl;

cout<<"5) Mirror of BST"<<endl;

cout<<"6) Exit"<<endl;

cin>>k;

cout<<endl;

return k;

}

void Dictionary::createfunc()

{

while (1)

{

switch (Menu())

{

case 1:

int numb;

cout<<"Enter Number"<<endl;

cin>>numb;

root = insertfunc(root,numb);

break;

case 2:

Inorder(root);

break;

case 3:

cout<<Tree\_Height(root)<<endl;

break;

case 4:

Leaf(root);

cout<<count<<endl;

break;

case 5:

Mirror(root);

Inorder(root);

Mirror(root);

break;

case 6:

exit(1);

break;

default:

break;

}

}

}

Node \* Dictionary::insertfunc(Node \* temproot,int t)

{

Node \* temp = new Node(t);

if(temproot==NULL)

{

return temp;

}

if(t < temproot->no)

{

temproot->left = insertfunc(temproot->left,t);

}

else

{

temproot->right= insertfunc(temproot->right,t);

}

return temproot;

}

void Dictionary ::Inorder(Node \*root)

{

if (root == NULL)

{

return;

}

Inorder(root->left);

cout<<root->no<<endl;

Inorder(root->right);

}

int count=0;

void Dictionary::Leaf(Node \*root)

{

if(root==NULL)

{

return;

}

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

{

count++;

}

Leaf(root->left);

Leaf(root->right);

}

int Dictionary::Tree\_Height(Node \* root)

{

int lheight=0,rheight=0;

if(root==NULL)

{

return 0;

}

else

{

lheight=Tree\_Height(root->left);

rheight=Tree\_Height(root->right);

if(lheight>rheight)

{

return (lheight+1);

}

else

{

return (rheight+1);

}

}

}

void Dictionary::Mirror(Node \* root)

{

if(root==NULL)

{

return;

}

else

{

Mirror(root->left);

Mirror(root->right);

Node \* temp = root->right;

root->right = root->left ;

root->left = temp;

}

}

int main()

{

Dictionary d;

d.createfunc();

return 0;

}