//Assignment no 3

#include <iostream>

using namespace std;

class node

{

public:

int data;

node\* left;

node\* right;

};

class bst

{

public:

node\* root;

int cnt;

bst()

{

root=NULL;

cnt=0;

}

void insert();

void inorder(node\* temp);

void preorder(node\* temp);

void smallest();

void mirror(node\* root);

void largest();

int search(int key);

int height(node\* root);

};

void bst::insert()

{

node \*newnode,\*temp;

newnode=new node();

newnode->left=NULL;

newnode->right=NULL;

cout<<"Insert data in new node:";

cin>>newnode->data;

if(root==NULL)

{

root=newnode;

cout<<"Node inserted successfully."<<endl;

}

else

{

int flag=0;

temp=root;

while(flag==0)

{

if(newnode->data < temp->data)

{

if(temp->left==NULL)

{

temp->left=newnode;

cout<<"Node inserted successfully."<<endl;

flag=1;

}

else

{

temp=temp->left;

}

}

else if(newnode->data > temp->data)

{

if(temp->right==NULL)

{

temp->right=newnode;

cout<<"Node inserted successfully."<<endl;

flag=1;

}

else

{

temp=temp->right;

}

}

else

{

cout<<"Data already exists."<<endl;

flag=1;

}

}

}

}

void bst::inorder(node\* temp)

{

if(temp!=NULL)

{

inorder(temp->left);

cout<<temp->data<<" ";

cnt++;

inorder(temp->right);

}

}

void bst::preorder(node\* temp)

{

if(temp!=NULL)

{

cout<<temp->data<<" ";

preorder(temp->left);

preorder(temp->right);

}

}

void bst::smallest()

{

node \*temp;;

temp=root;

while(temp->left!=NULL)

{

temp=temp->left;

}

cout<<"Smallest element is:"<<temp->data<<endl;

}

void bst::largest()

{

node \*temp;;

temp=root;

while(temp->right!=NULL)

{

temp=temp->right;

}

cout<<"Largest element is:"<<temp->data<<endl;

}

void bst::mirror(node\* root)

{

node\* temp;

if(root!=NULL)

{

temp=root->left;

root->left=root->right;

root->right=temp;

}

}

int bst::search(int key)

{

node\* temp;

temp=root;

while(1)

{

if(key<temp->data)

{

if(temp->left != NULL)

{

temp=temp->left;

}

else

return 0;

}

else if(key>temp->data)

{

if(temp->right != NULL)

{

temp=temp->right;

}

else

return 0;

}

else

{

return 1;

}

}

}

int bst::height(node\* root)

{

int rh,lh;

if(root==NULL)

return 0;

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

return 0;

rh=height(root->right);

lh=height(root->left);

if(rh>lh)

return (rh+1);

else

return (lh+1);

}

int main()

{

bst obj;

int ch;

do

{

cout<<endl;

cout<<"\n\*\*MENU\*\*";

cout<<"\n1.Insert";

cout<<"\n2.Display inorder";

cout<<"\n3.Display preorder";

cout<<"\n4.Display Smallest element";

cout<<"\n5.Display Largest element";

cout<<"\n6.Display Mirror";

cout<<"\n7.Search";

cout<<"\n8.Height";

cout<<"\n9.Exit";

cout<<"\nEnter your choice:";

cin>>ch;

switch(ch)

{

case 1:

{

obj.insert();

break;

}

case 2:

{

cout<<"Inorder:";

obj.inorder(obj.root);

break;

}

case 3:

{

cout<<"Preorder:";

obj.preorder(obj.root);

break;

}

case 4:

{

obj.smallest();

break;

}

case 5:

{

obj.largest();

break;

}

case 6:

{

obj.mirror(obj.root);

cout<<"Mirror:";

obj.inorder(obj.root);

break;

}

case 7:

{

int key;

cout<<"Enter the key to be searched:";

cin>>key;

int result=obj.search(key);

if(result==1)

{

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

}

else

{

cout<<"Element Not Found."<<endl;

}

break;

}

case 8:

{

int result=obj.height(obj.root);

cout<<"Height:"<<result<<endl;

break;

}

case 9:

{

cout<<"End of Program."<<endl;

break;

}

default:

{

cout<<"Invalid choice!!"<<endl;

}

}

}while(ch!=9);

return 0;

}

output:

gescoe@gescoe-OptiPlex-3010:~/Desktop/SE-A-55$ g++ Bst.cpp

gescoe@gescoe-OptiPlex-3010:~/Desktop/SE-A-55$ ./a.out

\*\*MENU\*\*

1. Insert
2. Display inorder
3. Display preorder
4. Display Smallest element
5. Display Largest element
6. Display Mirror
7. Search
8. Height
9. Exit

Enter your choice:1

Insert data in new node:5

Node inserted successfully.

\*\*MENU\*\*

1. Insert
2. Display inorder
3. Display preorder
4. Display Smallest element
5. Display Largest element
6. Display Mirror
7. Search
8. Height
9. Exit

Enter your choice:1

Insert data in new node:1

Node inserted successfully.

\*\*MENU\*\*

1. Insert
2. Display inorder
3. Display preorder
4. Display Smallest element
5. Display Largest element
6. Display Mirror
7. Search
8. Height
9. Exit

Enter your choice:1

Insert data in new node:10

Node inserted successfully.

\*\*MENU\*\*

1. Insert
2. Display inorder
3. Display preorder
4. Display Smallest element
5. Display Largest element
6. Display Mirror
7. Search
8. Height
9. Exit

Enter your choice:2

Inorder:1 5 10

\*\*MENU\*\*

1. Insert
2. Display inorder
3. Display preorder
4. Display Smallest element
5. Display Largest element
6. Display Mirror
7. Search
8. Height
9. Exit

Enter your choice:3

Preorder:5 1 10

\*\*MENU\*\*

1. Insert
2. Display inorder
3. Display preorder
4. Display Smallest element
5. Display Largest element
6. Display Mirror
7. Search
8. Height
9. Exit

Enter your choice:4

Smallest element is:1

\*\*MENU\*\*

1. Insert
2. Display inorder
3. Display preorder
4. Display Smallest element
5. Display Largest element
6. Display Mirror
7. Search
8. Height
9. Exit

Enter your choice:5

Largest element is:10

\*\*MENU\*\*

1. Insert
2. Display inorder
3. Display preorder
4. Display Smallest element
5. Display Largest element
6. Display Mirror
7. Search
8. Height
9. Exit

Enter your choice:6

Mirror:10 5 1

\*\*MENU\*\*

1. Insert
2. Display inorder
3. Display preorder
4. Display Smallest element
5. Display Largest element
6. Display Mirror
7. Search
8. Height
9. Exit

Enter your choice:7

Enter the key to be searched:10

Element Not Found.

\*\*MENU\*\*

1. Insert
2. Display inorder
3. Display preorder
4. Display Smallest element
5. Display Largest element
6. Display Mirror
7. Search
8. Height
9. Exit

Enter your choice:8

Height:1

\*\*MENU\*\*

1. Insert
2. Display inorder
3. Display preorder
4. Display Smallest element
5. Display Largest element
6. Display Mirror
7. Search
8. Height
9. Exit

Enter your choice:9

End of Program.

gescoe@gescoe-OptiPlex-3010:~/Desktop/SE-A-55$