#include<iostream>

#include<stack>

#include<queue>

using namespace std;

struct node

{

int data;

node\* left;

node\* right;

};

struct node\* create\_node()

{

node\* temp;

temp=new node;

cout<<"ENTER THE DATA:";

cin>>temp->data;

temp->left=NULL;

temp->right=NULL;

return temp;

}

struct node\* create\_bst()

{

node \*temp,\*ptr,\*root=NULL;

char ch;

do

{

temp=create\_node();

if(root==NULL)

{

root=temp;

}

else

{

ptr=root;

while(ptr!=NULL)

{

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

{

if(ptr->left==NULL)

{

ptr->left=temp;

//cout<<ptr->data<<endl;

break;

}

else

ptr=ptr->left;

}

else

{

if(ptr->right==NULL)

{

ptr->right=temp;

//cout<<ptr->data<<endl;

break;

}

else

ptr=ptr->right;

}

}

}

cout<<"\nWANT TO CONTINUE(Y/N):";

cin>>ch;

}while(ch=='y' or ch=='Y');

return root;

}

void preorder(node \*root)

{

node \*curr;

if(root==NULL)

return;

stack<node\*> s;

s.push(root);

while(!s.empty())

{

curr=s.top();

cout<<curr->data<<" ";

s.pop();

if(curr->right!=NULL)

s.push(curr->right);

if(curr->left!=NULL)

s.push(curr->left);

}

cout<<endl;

}

void inorder(node \*root)

{

stack<node\*> s;

node \*curr=root;

while(curr!=NULL || !s.empty())

{

while(curr!=NULL)

{

s.push(curr);

curr=curr->left;

}

curr=s.top();

s.pop();

cout<<curr->data<<" ";

curr=curr->right;

}

cout<<endl;

}

void postorder(node \*root)

{

if(root==NULL)

return ;

stack<node\*> s1,s2;

node \*curr=root;

s1.push(curr);

while (!s1.empty())

{

curr=s1.top();

s1.pop();

s2.push(curr);

if(curr->left!=NULL)

s1.push(curr->left);

if(curr->right!=NULL)

s1.push(curr->right);

}

while(!s2.empty())

{

curr=s2.top();

cout<<curr->data<<" ";

s2.pop();

}

cout<<endl;

}

//TOTAL NO. OF LEAF NODE OF BST

int getleaf\_node(struct node\* root)

{

if (root==NULL)

return 0;

queue<node\*> q;

int count = 0;

q.push(root);

while (!q.empty())

{

struct node \*temp = q.front();

q.pop();

if (temp->left != NULL)

q.push(temp->left);

if (temp->right != NULL)

q.push(temp->right);

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

{

cout<<temp->data<<endl;

count++;

}

}

return count;

}

// HEIGHT OF A BST

int height(node \*root)

{

int d1,d2;

if(root==NULL)

{

return 0;

}

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

return 0;

d1=height(root->left);

d2=height(root->right);

if(d1>=d2)

return d1+1;

else

return d2+1;

}

//MIRROR IMAGE OF GIVEN BST

node\* mirrorImage(node\* root) {

if (root == NULL) {

return root;

}

node\* left = mirrorImage(root->left);

node\* right = mirrorImage(root->right);

root->left = right;

root->right = left;

return root;

}

int main()

{

node \*b;

int f;

char ch;

cout<<"\n\t\tENTER THE ELEMENT IN BST:\n"<<endl;

b=create\_bst();

cout<<"\n\nTHE BINARY SEARCH TREE:";

cout<<"\n1.PREORDER TRAVERSAL:";

preorder(b);

cout<<"\n2.INORDER TRAVERSAL:";

inorder(b);

cout<<"\n3.POSTORDER TRAVERSAL:";

postorder(b);

cout<<"\n\t\t..MENU..\n\t1.CALCULATE LEAF NODES\n\t2.CALCULATE HEIGHT OF BST\n\t3.DISPLAY MIRROR IMAGE FOR BST\n\t4.EXIT\n";

do

{

cout<<"\nENTER THE CHOICE:";

cin>>f;

switch(f)

{

case 1:

cout<<"\nTOTAL NO. OF LEAF NODES IN BST:"<<getleaf\_node(b)<<endl;

break;

case 2:

cout << "\nThe height of the BST is: " << height(b) <<endl;

break;

case 3:

cout<<"\nMIRROR IMAGE OF BST(PREORDER TRAVERSAL):\n"<<endl;

b=mirrorImage(b);

preorder(b);

break;

case 4:

exit(1);

break;

}

cout<<"\n\nWANT TO CONTINUE(Y/N):";

cin>>ch;

}while(ch=='y' || ch=='Y');

return 0;

}