Binary search tree assignment 6

#include<iostream>

#include<queue>

using namespace std;

class node {

public:

int data;

node\* left;

node\* right;

node(int d) {

this -> data = d;

this -> left = NULL;

this -> right = NULL;

}

};

node\* insertIntoBST(node\* root , int d){

if(root == NULL){

root = new node(d);

return root;

}

if (d > root->data)

root -> right = insertIntoBST(root->right,d);

else

root -> left = insertIntoBST(root -> left, d);

}

void takeInput(node\* &root){

int data;

cin>> data;

while( data != -1){

root = insertIntoBST(root , data);

cin >> data;

}

}

int height(node\* &root){

if(root == NULL)

return 0;

int left\_subtree = 1 + height(root->left);

int right\_subtree = 1 + height(root->right);

int maxHeight = max(left\_subtree,right\_subtree);

return maxHeight;

}

bool search(node\* &root,int element){

if (root == NULL)

return 0;

if(root -> data == element)

return 1;

if(root->data > element)

return search(root->left, element);

else

return search(root->right, element);

}

int minimum(node\* &root){

if(root -> left == NULL)

return (root -> data);

return min(minimum(root->left),root->data);

cout <<"\n";

}

void mirror(node\* &root){

if(root == NULL)

return ;

mirror(root->left);

mirror(root->right);

swap(root->left,root->right);

}

void traversal(node\* &root){

if(root == NULL)

return;

//inorder traversal mai first left then data print then we have to move right

traversal(root->left);

cout << root->data <<"\t";

traversal (root->right);

}

int main(){

node\* root = NULL;

while(true){

cout<<"--------------Binary search tree----------------"<<endl;

cout << "1.Insert node\n2.Find number of nodes in longest path from root\n3.Minimum data value found in the tree\n4.Change the treee so that role of left and right pointers are swapped at every node\n5.Search a Node\n6.Exit"<< endl;

int choice ;

cin >> choice ;

if(choice == 1)

takeInput(root);

else if (choice == 2){

cout << "Maximum height :: ";

cout << height(root) << "\n";

}

else if (choice == 3)

cout << minimum(root);

else if (choice == 4){

mirror(root);

traversal(root);

cout<<"\n";

}

else if(choice == 5){

int element;

cout << "Enter the element to search :: ";

cin>>element;

if (search(root,element) )

cout << "Element found\n";

else

cout << "Element not found\n";

}

else if(choice == 6)

break;

else

cout<< "Invalid input";

}

}