#include<iostream>

using namespace std;

struct node {

int data;

node\* right;

node\* left;

};

//Binary Search Tree

class BST {

void insert(node\*, int);

bool search(int, node\*);

void inorder(node\*);

void preorder(node\*);

void postorder(node\*);

public:

node\* root;

BST()

{

root = NULL;

}

int max();

int min();

void insert(int);

bool search(int);

void inorder();

void preorder();

void postorder();

};

void BST::insert(int data) {

if (root != NULL) {

insert(root, data);

}

else {

root = new node;

root->data = data;

root->left = NULL;

root->right = NULL;

}

}

void BST::insert(node\* n, int data) {

if (data < n->data) {

if (n->left == NULL) {

n->left = new node;

n->left->data = data;

n->left->left = NULL;

n->left->right = NULL;

}

else {

insert(n->left, data);

}

}

else if (data >= n->data) {

if (n->right == NULL) {

n->right = new node;

n->right->data = data;

n->right->right = NULL;

n->right->left = NULL;

}

else {

insert(n->right, data);

}

}

}

void BST::inorder() {

if (root == NULL) {

cout << "List is empty" << endl;

}

else {

inorder(root);

}

}

void BST::inorder(node\* n) {

if (n != NULL) {

inorder(n->left);

cout << n->data << endl;

inorder(n->right);

}

}

void BST::preorder() {

if (root == NULL) {

cout << "List is empty" << endl;

}

else {

preorder(root);

}

}

void BST::preorder(node\* n) {

if (n != NULL) {

//cout.width(10);cout<< n->data<<endl;

//cout.width(10); cout << std::left;;

//preorder(n->left);

//cout.width(10); cout << std::right;

//preorder(n->right);

}

}

void BST::postorder() {

if (root == NULL) {

cout << "List is empty" << endl;

}

else {

postorder(root);

}

}

void BST::postorder(node\* n) {

if (n != NULL) {

cout << n->data << endl;

postorder(n->right);

postorder(n->left);

}

}

bool BST::search(int data) {

if (root == NULL) {

cout << "List is empty" << endl;

}

else {

return search(data,root);

}

}

bool BST::search(int data, node\* n) {

bool answer = false;

if (n == NULL) {

return false;

}

if (data == n->data) {

return true;

}

else if (data < n->data) {

answer = search(data, n->left);

}

else {

answer = search(data, n->right);

}

return answer;

}

void main() {

BST bst;

bst.insert(10);

bst.insert(8);

bst.insert(9);

bst.insert(23);

bst.insert(16);

cout << bst.search(111) << endl;

//bst.inorder();

//bst.preorder();

//bst.postorder();

}

//Ev tapshirigi axtarilan elementi silen method yazin,\SADECE SILMEDE DIQQETLI OLAN //AGAC POZULMASIN