#include <iostream>

#include<vector>

#include <malloc.h>

using namespace std;

#define T int

typedef struct BSTNode {

T data;

struct BSTNode\*leftChild;

struct BSTNode\*rightChild;

}BSTNode;

typedef BSTNode\* BST;

void InsertBST(BST &t, T x) {

if (t == NULL) {

//申请结点

BSTNode\*s = (BSTNode\*)malloc(sizeof(BSTNode));

s->data = x;

s->leftChild = s->rightChild = NULL;

//插入节点

t = s;

return;

}

else if (x < t->data) {

InsertBST(t->leftChild, x);

}

else if (x > t->data) {

InsertBST(t->rightChild, x);

}

}

T Min(BST t) {

while (t->leftChild != NULL) {

t = t->leftChild;

}

return t->data;

}

T Max(BST t) {

while (t->rightChild != NULL) {

t = t->rightChild;

}

return t->data;

}

void InOrder(BST t) {

if (t != NULL) {

InOrder(t->leftChild);

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

InOrder(t->rightChild);

}

}

BSTNode\* Search(BST t,T key) {

if (t == NULL || t->data == key) {

return t;

}

if (key < t->data) {

Search(t->leftChild, key);

}

else {

Search(t->rightChild, key);

}

}

void DestroyBST(BST&t) {

if (t != NULL) {

DestroyBST(t->leftChild);

DestroyBST(t->rightChild);

}

free(t);

t = NULL;

}

void RemoveBST(BST&t,T key) {

if (t == NULL) {

return;

}

BSTNode\*p = t;

if (key < p->data) {

RemoveBST(t->leftChild, key);

}

else if (key > p->data) {

RemoveBST(t->rightChild, key);

}

else {

//既不大于也不小于

//说明找到了，开始进行删除操作

if (p->leftChild == NULL && p->rightChild == NULL) {

free(p);

t = NULL; //这句不能少

}

else if (p->leftChild != NULL && p->rightChild == NULL) {

t = p->leftChild;

free(p);

}

else if (p->leftChild == NULL && p->rightChild != NULL) {

t = p->rightChild;

free(p);

}

else {

//具有左右子树结点

p = t->leftChild;

while (p->rightChild != NULL) {

p = p->rightChild;

}

t->data = p->data;

RemoveBST(t->leftChild, p->data);

}

}

}

int main() {

vector<int>ar = { 16,3,7,11,9,26,18,14,15 };

int n = ar.size();

BST t = NULL;

for (int i = 0; i < n; i++) {

InsertBST(t, ar[i]);

}

cout << "最小值为：" << Min(t) << endl;

cout << "最大值为：" << Max(t) << endl;

InOrder(t);

cout << endl;

RemoveBST(t, 26);

InOrder(t);

return 0;

}