#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

typedef int KeyType;

typedef struct node {

KeyType data;

struct node\* lchild, \* rchild;

} BSTNode;

typedef BSTNode\* BSTree;

// 创建二叉搜索树

BSTree CreateBST(KeyType A[], int start, int end) {

int mid = (end + start) / 2;

BSTNode\* bt;

if (end < start) return NULL;

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

bt->data = A[mid];

bt->lchild = CreateBST(A, start, mid-1); //根据所给代码进行修改 原为mid 应改为mid-1

bt->rchild = CreateBST(A, mid + 1, end);

return bt;

}

// 判断二叉树是否为二叉搜索树

bool IsBST(BSTree tree) {

if (tree == NULL) return true;

if (tree->lchild != NULL && tree->lchild->data > tree->data)

return false; // 左子树中有节点的值大于根节点的值

if (tree->rchild != NULL && tree->rchild->data < tree->data)

return false; // 右子树中有节点的值小于根节点的值

if (!IsBST(tree->lchild) || !IsBST(tree->rchild))

return false; // 左子树或右子树不是二叉搜索树

return true;

}

// 插入节点到二叉搜索树

void InsertNode(BSTree\* tree, KeyType key) {

if (\*tree == NULL) {

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

(\*tree)->data = key;

(\*tree)->lchild = (\*tree)->rchild = NULL;

}

else {

if (key < (\*tree)->data)

InsertNode(&(\*tree)->lchild, key);

else if (key > (\*tree)->data)

InsertNode(&(\*tree)->rchild, key);

}

}

// 删除节点从二叉搜索树

int DeleteNode(BSTree\* tree, KeyType key) {

if (\*tree == NULL) return 0; // 节点不存在

if (key < (\*tree)->data)

return DeleteNode(&(\*tree)->lchild, key);

else if (key > (\*tree)->data)

return DeleteNode(&(\*tree)->rchild, key);

else {

// 找到要删除的节点

BSTNode\* p;

if ((\*tree)->lchild && (\*tree)->rchild) {

// 被删除节点有两个子节点

p = (\*tree)->rchild;

while (p->lchild)

p = p->lchild;

(\*tree)->data = p->data; // 用右子树中最小的值代替被删除节点的值

return DeleteNode(&(\*tree)->rchild, p->data); // 在右子树中删除最小值的节点

}

else {

// 被删除节点有一个或没有子节点（可以统一操作）无子节点的node两个两个child都是NULL 不影响操作

p = \*tree;

if ((\*tree)->lchild == NULL)

\*tree = (\*tree)->rchild;

else

\*tree = (\*tree)->lchild;

free(p);

return 1;

}

}

}

// 打印二叉搜索树(延用binary tree)

void PrintBST(BSTree b)

{

if (b != NULL)

{

printf("%d", b->data);

if (b->lchild != NULL || b->rchild != NULL)

{

printf("(");

PrintBST(b->lchild);

if (b->rchild != NULL) printf(",");

PrintBST(b->rchild);

printf(")");

}

}

}