本关任务：编写一个函数实现判定二叉树是否为二叉排序树 函数原型：status JudgeBiTree(BiTree T)； 功能说明：判定以二叉链表存储的二叉树是否为二叉排序树，是则返回YES，不是返回NO。

为了完成本关任务，你需要掌握：1.二叉排序树，2.二叉链表。 有关定义：

1. #define YES 1
2. #define NO 0
3. typedef int status;
4. typedef int KeyType;
5. typedef struct {
6. KeyType key;
7. char others[20];
8. } TElemType; //二叉树结点类型定义
9. typedef struct BiTNode{ //二叉链表结点的定义
10. TElemType data;
11. struct BiTNode \*lchild,\*rchild;
12. } BiTNode, \*BiTree;

**编程要求**

根据提示，在右侧编辑器补充代码，完成函数JudgeBiTree(BiTree T)的编写

**测试说明**

平台会自动读取输入数据，对你编写的代码进行测试，并输出结果。 下列输入为结点在满二叉树中的编号和结点值的序列，当结点在满二叉树中的编号为0时表示输入结束。例如： 6 4 d 表示值为（4，d）的结点在满二叉树中的编号为6； 0 0 null代表结束标记。输出判断根据该序列创建的二叉树是否为二叉排序树的结果。

测试输入：1 1 a 2 2 b 3 3 c 6 4 d 7 5 e 0 0 null 预期输出： NO

测试输入：1 4 a 2 2 b 3 6 c 4 1 d 5 3 e 6 5 f 7 7 g 0 0 null 预期输出： YES

#include "def.h"

#include "stu.h"

typedef struct {

    int pos;

    TElemType data;

} DEF;

status CreateBiTree(BiTree &T,DEF definition[])

{

    int i=0,j;

    static BiTNode \*p[100];

    while (j=definition[i].pos)

    {

         p[j]=(BiTNode \*)malloc(sizeof(BiTNode));

       p[j]->data=definition[i].data;

       p[j]->lchild=NULL;

       p[j]->rchild=NULL;

       if (j!=1)

            if (j%2)   p[j/2]->rchild=p[j];

           else      p[j/2]->lchild=p[j];

       i++;

    }

    T=p[1];

    return OK;

}

int main()

{

DEF definition[100];

BiTree T;

int ans,i=0;

do {

    scanf("%d%d%s",&definition[i].pos,&definition[i].data.key,definition[i].data.others);

} while (definition[i++].pos);

CreateBiTree(T,definition);

if (JudgeBiTree(T))

       printf("YES");

else    printf("NO");

return 1;

}

#include "stdio.h"

#include "stdlib.h"

#include "stu.h"

int main()

{

int ans,\*a,n,i;

scanf("%d",&n);

a=(int \*)malloc(sizeof(int)\*n);

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

    scanf("%d",&a[i]);

printf("%d",MidValue(a,n));

free(a);

return 1;

}