在一个二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序。请完成一个函数，输入这样的一个二维数组和一个整数，判断数组中是否含有该整数。

//从右上角开始搜索

class Solution {

public:

bool Find(int target, vector<vector<int> > array) {

int rowCount = array.size();//行

int colCount = array[0].size();//列

int i,j;

for(i=0,j=colCount-1;i<rowCount&&j>=0;)

{

if(target == array[i][j])

return true;

if(target > array[i][j])

{

i++;

continue;//continue：终止当前循环中的一次操作，继续执行循环；break：直接跳出当前循环，执行循环外的语句。

}

if(target < array[i][j])

{

j--;

continue;

}

}

return false;

}

};

请实现一个函数，将一个字符串中的空格替换成“%20”。例如，当字符串为We Are Happy.则经过替换之后的字符串为We%20Are%20Happy。

//采用从后向前的替换方式。

//若采用从前往后的替换方式，则找到的第一个空格位置，则其后所有的字符均要后移。

//比如，若第一个空格在第5个位置，指针p值向它，若要将其替换成“%20”，则需要：

// \*p = "%";

// \*(p +1) = "2";

// \*(p +2) = "0";

//但这里，\*(p +1)和\*(p +2)原本就指向字符串中的后续字符，直接按照上述代码操作会丢失这部分的信息，则需要先保存p位置所有的字符（将这些字符后移）。

//若采用从后往前的替换方式，则找到最后一个空格后，只需将该空格后的字符移动到最终位置。故而采取这种方案。

//思路：先获取空格数。再从后往前遍历寻找空格进行替换。

class Solution {

public:

void replaceSpace(char \*str,int length) {

int count = 0;

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

if(str[i] == ' ')

count++;

}

for(int i = length-1; i>=0; i--){

if(str[i] != ' ')

str[i+count\*2] = str[i];

else{

count --;

str[i+count\*2] = '%';

str[i+count\*2+1] = '2';

str[i+count\*2+2] = '0';

}

}

}

};

输入一个链表，从尾到头打印链表每个节点的值。

/\*\*

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* ListNode(int x) :

\* val(x), next(NULL) {

\* }

\* };

\*/

class Solution {

public:

vector<int> printListFromTailToHead(ListNode\* head) {

vector<int> value;

if(head != NULL){

value.insert(value.begin(),head->val);

if(head->next != NULL){

vector<int> tmpVec = printListFromTailToHead(head->next);

if(tmpVec.size() > 0)

value.insert(value.begin(),tmpVec.begin(),tmpVec.end());

}

}

return value;

}

};