题目描述 1 二维数组中是否存在该值

在一个二维数组中（每个一维数组的长度相同），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。

```
public boolean Find(int target, int [][] array)
```

从右上角开始遍历，若 target 大于该值，则向下一行，若小于该值，则向左一列。

题目描述 2 字符串空格替换成%20

请实现一个函数，将一个字符串中的每个空格替换成“%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。

```
public String replaceSpace(StringBuffer str)
```

将当前 StringBuffer 长度设置为 oldLength+spaceNum*2，从后向前遍历，遇到空格替换。

题目描述 3 链表反转，输出 ArrayList

输入一个链表，按链表值从尾到头的顺序返回一个 ArrayList

```
public ArrayList<Integer> printListFromTailToHead(ListNode
listNode)
```

递归：当该节点不为空时，先以该函数调用下一个节点，再将该节点添加到链表中。

```
this.printListFromTailToHead(listNode.next);
```

题目描述 4 根据先中序遍历重建二叉树

输入某二叉树的前序遍历和中序遍历的结果，请重建出该二叉树。假设输入的前序遍历和中序遍历的结果中都不含重复的数字。例如输入前序遍历序列{1,2,4,7,3,5,6,8}和中序遍历序列{4,7,2,1,5,3,8,6}，则重建二叉树并返回。

```
public TreeNode reConstructBinaryTree(int [] pre,int [] in)
```

调用可以递归的子函数

```
public TreeNode reConstructBinaryTree(int[] pre,int startPre,int
endPre,int[] in,int startIn,int endIn)
```

该函数返回当前节点值。头大于尾则返回 null。

前序遍历：根-左-右。　中序遍历：左-根-右。

题目描述 5 两个栈实现队列

用两个栈来实现一个队列，完成队列的 Push 和 Pop 操作。队列中的元素为 int 类型。

```
Stack<Integer> stack1 = new Stack<Integer>();
Stack<Integer> stack2 = new Stack<Integer>();
public void push(int node)
public int pop()
```

Push:Stack1 用来 push。

Pop:Stack1 和 stack2 都为空时，抛出异常。

Stack2 为空，stack1 不为空时，将 stack1 中的值全部移到 stack2。

弹出 stack2 的值。

题目描述 6 旋转数组的最小元素

把一个数组最开始的若干个元素搬到数组的末尾，我们称之为数组的旋转。 输入一个非减排序的数组的一个旋转，输出旋转数组的最小元素。 例如数组{3,4,5,1,2}为{1,2,3,4,5}的一个旋转，该数组的最小值为 1。 NOTE：给出的所有元素都大于 0，若数组大小为 0，请返回 0。

public int minNumberInRotateArray(int [] array)

设置两个指针为最左端和最右端，当 low<high 时：

采用二分法解答这个问题，mid = low + (high - low)/2，需要考虑三种情况：

(1)array[mid] > array[high]: low = mid + 1

(2)array[mid] == array[high]: high = high - 1

出现这种情况的 array 类似 [1,0,1,1,1] 或者[1,1,1,0,1]，此时最小数字不好判断在 mid 左边还是右边,这时只好一个一个试 ，

(3)array[mid] < array[high]: high = mid

*注意这里有个坑：如果待查询的范围最后只剩两个数，那么 mid 一定会指向下标靠前的数字,比如 array = [4,6]:

array[low] = 4 ;array[mid] = 4 ; array[high] = 6 ;

如果 high = mid - 1，就会产生错误，因此 high = mid，但情形(1)中 low = mid + 1 就不会错误。return array[low]和 array[high]都可以。

题目描述 7 斐波那契数列第 n 项

大家都知道斐波那契数列，现在要求输入一个整数 n，请你输出斐波那契数列的第 n 项（从 0 开始，第 0 项为 0）。

n<=39

public int Fibonacci(int n)

递归：return Fibonacci(n-1)+Fibonacci(n-2)。

题目描述 8 青蛙跳台阶 1-2 跳法数

一只青蛙一次可以跳上 1 级台阶，也可以跳上 2 级。求该青蛙跳上一个 n 级的台阶总共有多少种跳法（先后次序不同算不同的结果）

递归：return JumpFloor(target-1)+JumpFloor(target-2)。

题目描述 9 青蛙跳台阶 1-跳法数

一只青蛙一次可以跳上 1 级台阶，也可以跳上 2 级……它也可以跳上 n 级。求该青蛙跳上一个 n 级的台阶总共有多少种跳法。

递归：return JumpFloorII(target-1)*2。

题目描述 10 2*1 覆盖 2*n 矩形方法数

我们可以用 2*1 的小矩形横着或者竖着去覆盖更大的矩形。请问用 n 个 2*1 的小矩形无重叠地覆盖一个 2*n 的大矩形，总共有多少种方法？

递归：return RectCover(target-1)+RectCover(target-2)。

题目描述 11 二进制表示中 1 的个数

输入一个整数，输出该数二进制表示中 1 的个数。其中负数用补码表示

public int NumberOf1(int n)

n=(n-1)&n 时会减少一个 1。

题目描述 12 幂运算

给定一个 double 类型的浮点数 base 和 int 类型的整数 exponent。求 base 的 exponent 次方。

```
public double Power(double base, int exponent)
```

快速幂算法：

exponent 大于 0 时，e=exponent;exponent 小于 0 时，若 base 为 0 则抛出异常，否则 e=-exponent。

```
while(e!=0) {
    if((e&1)==1) {
        res*=curr;
    }
    curr*=curr;
    e>>=1;
}
```

题目描述 13 数组奇数和偶数左右分离

输入一个整数数组，实现一个函数来调整该数组中数字的顺序，使得所有的奇数位于数组的前半部分，所有的偶数位于数组的后半部分，并保证奇数和奇数，偶数和偶数之间的相对位置不变。

```
public void reOrderArray(int [] array)
```

采用冒泡算法，若左为偶右为奇，则交换位置。

题目描述 14 链表倒数第 k 个结点

输入一个链表，输出该链表中倒数第 k 个结点。

Pre 先走 1-k 步（若下一节点为空，则直接返回 null），然后两个节点一起走。

```
public ListNode FindKthToTail(ListNode head,int k)
```

题目描述 15 反转链表，输出链表头

输入一个链表，反转链表后，输出新链表的表头。

```
public ListNode ReverseList(ListNode head)
ListNode pre=null;
ListNode next=null;
while(head!=null) {
    next=head.next;
    head.next=pre;
    pre=head;
    head=next;
}
    return pre;
```

题目描述 16 两个单调递增链表合并

输入两个单调递增的链表，输出两个链表合成后的链表，当然我们需要合成后的链表满足单调不减规则。

public ListNode Merge(ListNode list1,ListNode list2)

新建一个节点，比较 list1 和 list2，哪个值小，下一个指向谁。记得考虑两者前后为空。

题目描述 17 二叉树 B 是否为 Ade 子结构

输入两棵二叉树 A，B，判断 B 是不是 A 的子结构。（ps：我们约定空树不是任意一个树的子结构）

public boolean HasSubtree(TreeNode root1,TreeNode root2)

调用子函数，root1 和 root2 的根结点相同时。

```
return isSubtree(root1,root2) ||
          HasSubtree(root1.left,root2) ||
          HasSubtree(root1.right,root2);
```

题目描述 18 二叉树镜像

操作给定的二叉树，将其变换为源二叉树的镜像。

public void Mirror(TreeNode root)

通过 temp 交换左右节点，然后递归。

题目描述 19 从外向里打印矩阵

输入一个矩阵，按照从外向里以顺时针的顺序依次打印出每一个数字，例如，如果输入如下 4 X 4 矩阵： 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 则依次打印出数字 1,2,3,4,8,12,16,15,14,13,9,5,6,7,11,10.

public ArrayList<Integer> printMatrix(int [][] matrix)

前下左上遍历，记得最后四个指针的加减。

```
for(int i=left;i<=right;i++)
for(int i=top+1;i<=bottom;i++)
for(int i=right-1;i>=left && top<bottom;i--)
for(int i=bottom-1;i>top && left<right;i--)
```

题目描述 20 栈中最小元素，时间复杂度 O（1）

定义栈的数据结构，请在该类型中实现一个能够得到栈中所含最小元素的 min 函数（时间复杂度应为 O（1））。

public void push(int node)

一个 data 放数据，一个 min 放最小值。

temp 为空或 node<=temp 时加入。

public void pop()

弹出两个值不同时，min 加入弹出值。

public int top()

public int min()

题目描述 21 是否为栈的弹出顺序

输入两个整数序列，第一个序列表示栈的压入顺序，请判断第二个序列是否可能为该栈的弹出顺序。假设压入栈的所有数字均不相等。例如序列 1,2,3,4,5 是某栈的压入顺序，序列 4,5,3,2,1 是该压栈序列对应的一个弹出序列，但 4,3,5,1,2 就不可能是该压栈

序列的弹出序列。（注意：这两个序列的长度是相等的）

```
public boolean IsPopOrder(int [] pushA,int [] popA)
```

新建一个 stack

```
Stack<Integer> s=new Stack<Integer>();
for(int i=0;i<pushA.length;i++) {
    s.push(pushA[i]);
    while(!s.empty() && s.peek()==popA[index]) {
        s.pop();
        index++;
    }
}
return s.empty();
```

题目描述 22 层序打印二叉树

从上往下打印出二叉树的每个节点，同层节点从左至右打印。

```
public ArrayList<Integer> PrintFromTopToBottom(TreeNode root)
```

新建 ArrayList<TreeNode> queue,进行遍历 while(queue.size()!=0)。

题目描述 23 判断是否二叉搜索树后序遍历结果

输入一个整数数组，判断该数组是不是某二叉搜索树的后序遍历的结果。如果是则输出 Yes,否则输出 No。假设输入的数组的任意两个数字都互不相同。

```
public boolean VerifySquenceOfBST(int [] sequence)
```

新建一个确认是否小大中的子函数。从右向左走，直至小于根结点。再从左向右走，若有大于的则返回 false,否则继续调用子函数。

题目描述 24？？ 节点值和为目标值的路径

输入一颗二叉树的跟节点和一个整数，打印出二叉树中结点值的和为输入整数的所有路径。路径定义为从树的根结点开始往下一直到叶结点所经过的结点形成一条路径。(注意：在返回值的 list 中，数组长度大的数组靠前)

```
public ArrayList<ArrayList<Integer>> FindPath(TreeNode root,int
target)
if(root==null) {
    return res;
}
list.add(root.val);
target-=root.val;
if(target==0 && root.left==null && root.right==null) {
    res.add(new ArrayList<Integer>(list));
}
FindPath(root.left,target);
FindPath(root.right,target);
list.remove(list.size()-1);
return res;
```

题目描述 25 复制随机指针链表

输入一个复杂链表（每个节点中有节点值，以及两个指针，一个指向下一个节点，另一个特殊指针指向任意一个节点），返回结果为复制后复杂链表的 head。（注意，输出结果中请不要返回参数中的节点引用，否则判题程序会直接返回空）

public RandomListNode Clone(RandomListNode pHead)

先复制链表，再复制随机指针，再拆分。

```
RandomListNode head=pHead.next;
pCur=pHead;
while(pCur!=null) {
    RandomListNode cloneNode=pCur.next;
    pCur.next=cloneNode.next;
    if(cloneNode.next!=null) {
        cloneNode.next=cloneNode.next.next;
    }
    pCur=pCur.next;
}
```

题目描述 26 二叉搜索树转换成双向链表

输入一棵二叉搜索树，将该二叉搜索树转换成一个排序的双向链表。要求不能创建任何新的结点，只能调整树中结点指针的指向。

public TreeNode Convert(TreeNode pRootOfTree)

新建两个 realHead 和 head 节点，调用 convertSub(pRootOfTree)子函数进行，先递归左节点再执行再递归右节点。

题目描述 27 字符串所有字符排列

输入一个字符串,按字典序打印出该字符串中字符的所有排列。例如输入字符串 abc,则打印出由字符 a,b,c 所能排列出来的所有字符串 abc,acb,bac,bca,cab 和 cba。

输入描述:

输入一个字符串,长度不超过 9(可能有字符重复),字符只包括大小写字母。

基于回溯法思想：

```
public class Solution {
    public ArrayList<String> Permutation(String str) {
        ArrayList<String> res = new ArrayList<String>();
        if(str.length() == 0) {
            return res;
        }
        fun(str.toCharArray(),res,0);
        Collections.sort(res);
        return res;
    }
    private void fun(char[] ch,ArrayList<String> res,int i){
        if(i == ch.length-1){
            if(!res.contains(new String(ch))){
                res.add(new String(ch));
```

```
                return;
            }
        }
        else {
            for(int j=i;j<ch.length;j++){
                swap(ch,i,j);
                fun(ch,res,i+1);
                swap(ch,i,j);
            }
        }
    }
    private void swap(char[] str, int i, int j) {
        if (i != j) {
            char t = str[i];
            str[i] = str[j];
            str[j] = t;
        }
    }
}
```

题目描述 28 数组次数超过一半的众数

数组中有一个数字出现的次数超过数组长度的一半，请找出这个数字。例如输入一个长度为 9 的数组{1,2,3,2,2,2,5,4,2}。由于数字 2 在数组中出现了 5 次，超过数组长度的一半，因此输出 2。如果不存在则输出 0。

```
public int MoreThanHalfNum_Solution(int [] array)
```

排序后，只有中间的数有可能为这个数字。

题目描述 29 数组最小的 k 个数

输入 n 个整数，找出其中最小的 K 个数。例如输入 4,5,1,6,2,7,3,8 这 8 个数字，则最小的 4 个数字是 1,2,3,4,。

```
public ArrayList<Integer> GetLeastNumbers_Solution(int [] input,
int k)
```

使用最大堆保存这 k 个数，每次只和堆顶比，如果比堆顶小，删除堆顶，新数入堆

```
PriorityQueue<Integer> maxHeap=new
        PriorityQueue<Integer>(k,new Comparator<Integer>() {
        public int compare(Integer o1,Integer o2) {
                return o2-o1;
        }
});
```

题目描述 30 最大连续子序列的和

HZ 偶尔会拿些专业问题来忽悠那些非计算机专业的同学。今天测试组开完会后,他又发话了:在古老的一维模式识别中,常常需要计算连续子向量的最大和,当向量全为正数的时候,问题很好解决。但是,如果向量中包含负数,是否应该包含某个负数,并期望旁边的正数会

弥补它呢？例如:{6,-3,-2,7,-15,1,2,2},连续子向量的最大和为 8(从第 0 个开始，到第 3 个为止)。给一个数组，返回它的最大连续子序列的和，你会不会被他忽悠住？(子向量的长度至少是 1)

```
public int FindGreatestSumOfSubArray(int[] array)
```

对于一个数 A，若是 A 的左边累计数非负，那么加上 A 能使得值不小于 A，认为累计值对整体和是有贡献的。如果前几项累计值负数，则认为有害于总和，total 记录当前值。

题目描述 31 非负整数中 1 出现的次数

求出 1~13 的整数中 1 出现的次数,并算出 100~1300 的整数中 1 出现的次数？为此他特别数了一下 1~13 中包含 1 的数字有 1、10、11、12、13 因此共出现 6 次,但是对于后面问题他就没辙了。ACMer 希望你们帮帮他,并把问题更加普遍化,可以很快的求出任意非负整数区间中 1 出现的次数（从 1 到 n 中 1 出现的次数）

```
public int NumberOf1Between1AndN_Solution(int n)
```

通过 i 从低到高遍历每一位。若为 0：res+=before*i;
若为 1： res+=before*i+after+1;否则：res+=(before+1)*i;

题目描述 32 数组拼接最小数

输入一个正整数数组，把数组里所有数字拼接起来排成一个数，打印能拼接出的所有数字中最小的一个。例如输入数组{3，32，321}，则打印出这三个数字能排成的最小数字为 321323。

放入链表中并排序，自定义比较器。

```
Collections.sort(list,new Comparator<Integer>() {
    public int compare(Integer i1,Integer i2) {
        String s1=i1+""+i2;
        String s2=i2+"" +i1;
        return s1.compareTo(s2);
    }
});
```

题目描述 33 第 N 个丑数

把只包含质因子 2、3 和 5 的数称作丑数（Ugly Number）。例如 6、8 都是丑数，但 14 不是，因为它包含质因子 7。 习惯上我们把 1 当做是第一个丑数。求按从小到大的顺序的第 N 个丑数。

```
public int GetUglyNumber_Solution(int index)
```

分为三个队列指针。List 先加入 1，当队列长度小于 index 时执行循环。，

题目描述 34 第一个只出现一次字符的位置

在一个字符串(0<=字符串长度<=10000，全部由字母组成)中找到第一个只出现一次的字符,并返回它的位置，如果没有则返回 -1（需要区分大小写）。

```
public int FirstNotRepeatingChar(String str)
```

新建'z'+1 长度的数组，记录出现次数。

题目描述 35 数组逆序对数量

在数组中的两个数字，如果前面一个数字大于后面的数字，则这两个数字组成一个逆序

对。输入一个数组,求出这个数组中的逆序对的总数 P。并将 P 对 1000000007 取模的结果输出。 即输出 P%1000000007。

归并排序的改进，把数据分成前后两个数组，合并数组，合并时。出现前面的数组值 array[i]大于后面数组值 array[j]时；则前面数组 array[i]~array[mid]都是大于 array[j]的，count += mid+1 - i。

```
public class Solution {
    public int InversePairs(int [] array) {
        if(array==null || array.length==0) {
            return 0;
        }
        int[] copy=new int[array.length];
        int count=getCount(array,copy,0,array.length-1);
        return count;
    }
    private int getCount(int[] array,int[] copy,int low,int high)
{
        if(low==high) {
            return 0;
        }
        int mid=(low+high)/2;
        int leftCount=getCount(array,copy,low,mid)%1000000007;
        int rightCount=getCount(array,copy,mid+1,high)%1000000007;
        int count=0;
        int i=mid;
        int j=high;
        int indexCopy=high;
        while(i>=low && j>mid) {
            if(array[i]>array[j]) {
                count+=j-mid;
                copy[indexCopy--]=array[i--];
                if(count>=1000000007) {
                    count%=1000000007;
                }
            }
            else {
                copy[indexCopy--]=array[j--];
            }
        }
        while(i>=low) {
            copy[indexCopy--]=array[i--];
        }
        while(j>mid) {
            copy[indexCopy--]=array[j--];
        }
```

```
        for(int s=low;s<=high;s++) {
            array[s]=copy[s];
        }
        return (leftCount+rightCount+count)%1000000007;
    }
}
```

题目描述 36 链表公共节点

输入两个链表，找出它们的第一个公共结点。

```
public ListNode FindFirstCommonNode(ListNode pHead1, ListNode
pHead2)
```

长度相同：直接遍历到结果。

长度不同，第一次循环遍历差值，第二次循环遍历共同节点。

```
while(p1!=p2) {
    p1=(p1==null?pHead2:p1.next);
    p2=(p2==null?pHead1:p2.next);
}
```

题目描述 37 数字在排序数组中的次数

统计一个数字在排序数组中出现的次数。

```
public int GetNumberOfK(int [] array , int k)
```

通过二分法获取第一次 `getFirstK` 和最后一次 `getLastK` 出现的索引，都不为`-1` 时返回长度。注意 `mid-1>=0 && array[mid-1]==k` 和 `mid+1<length && array[mid+1]==k` 两种特殊情况。

题目描述 38 二叉树的深度（最长路径）

输入一棵二叉树，求该树的深度。从根结点到叶结点依次经过的结点（含根、叶结点）形成树的一条路径，最长路径的长度为树的深度。

```
public int TreeDepth(TreeNode root)
```

方法一：采用递归。

方法二：采用类似层序遍历。

题目描述 39 是否平衡二叉树

输入一棵二叉树，判断该二叉树是否是平衡二叉树。

```
public boolean IsBalanced_Solution(TreeNode root)
```

调用子函数 `getDepth`，其中调用子函数计算左右子树深度，为`-1` 则直接返回，否则计算二者差与 1 的大小（差的绝对值不超过 1）。

题目描述 40 整型数组中只出现一次（其他两次）

一个整型数组里除了两个数字之外，其他的数字都出现了偶数次。请写程序找出这两个只出现一次的数字。

```
public void FindNumsAppearOnce(int [] array,int num1[] , int
num2[])
```

通过 `sum^=array[i]`确定两个数的不同位，然后分类进行异或。

问题：数组 a 中只有一个数出现一次，其他数都出现了 2 次，找出这个数字。

```
res = res ^ a[i];
```

问题：数组 a 中只有一个数出现一次，其他数字都出现了 3 次，找出这个数字。

在总数的二进制位上相加，若不为三的倍数，则进行或运算。

```
for(int i = 0; i < len; i++){
    for(int j = 0; j < 32; j++){
        bits[j] = bits[j] + ( (a[i]>>>j) & 1);
    }
}
for(int i = 0; i < 32; i++){
    if(bits[i] % 3 !=0){
        res = res | (1 << i);
    }
}
```

题目描述 41 所有和为 S 的连续序列

小明很喜欢数学,有一天他在做数学作业时,要求计算出 9~16 的和,他马上就写出了正确答案是 100。但是他并不满足于此,他在想究竟有多少种连续的正数序列的和为 100(至少包括两个数)。没多久,他就得到另一组连续正数和为 100 的序列:18,19,20,21,22。现在把问题交给你,你能不能也很快的找出所有和为 S 的连续正数序列? Good Luck!

输出描述:

输出所有和为 S 的连续正数序列。序列内按照从小至大的顺序，序列间按照开始数字从小到大的顺序。

```
public ArrayList<ArrayList<Integer> > FindContinuousSequence(int sum)
```

n 为奇数时：(n & 1) == 1 && sum % n == 0；

n 为偶数时：(sum % n) * 2 == n；序列中间两个数的平均值是序列的平均值，而这个平均值的小数部分为 0.5。

```
for(int n=(int)Math.sqrt(2*sum);n>=2;n--) {
    if(((n&1)==1 && sum%n==0) || (sum%n)*2==n) {
        ArrayList<Integer> list=new ArrayList<Integer>();
        for(int k=(sum/n)-(n-1)/2,j=0;j<n;k++,j++) {
            list.add(k);
        }
        res.add(list);
    }
}
```

题目描述 42 递增数组中两个数的和为 S，选择乘积最小的，小的数先输出

输入一个递增排序的数组和一个数字 S，在数组中查找两个数，使得他们的和正好是 S，如果有多对数字的和等于 S，输出两个数的乘积最小的。

输出描述:

对应每个测试案例，输出两个数，小的先输出。

```
public ArrayList<Integer> FindNumbersWithSum(int [] array,int
```

sum)
左右两个指针，与 sum 相比，较小时左指针++；较大时右指针--；否则返回。

题目描述 43 字符串循环左移 N 位

汇编语言中有一种移位指令叫做循环左移（ROL），现在有个简单的任务，就是用字符串模拟这个指令的运算结果。对于一个给定的字符序列 S，请你把其循环左移 K 位后的序列输出。例如，字符序列 S=”abcXYZdef”,要求输出循环左移 3 位后的结果，即“XYZdefabc”。是不是很简单？OK，搞定它！

public String LeftRotateString(String str,int n)

调用旋转子函数 reverse(char[] ch,int start,int end)。

转换成数组形式变换：先旋转 0~n-1，再旋转 n~length-1，再旋转 0~length-1。

题目描述 44 翻转句子中的单词

牛客最近来了一个新员工 Fish，每天早晨总是会拿着一本英文杂志，写些句子在本子上。同事 Cat 对 Fish 写的内容颇感兴趣，有一天他向 Fish 借来翻看，但却读不懂它的意思。例如，“student. a am I”。后来才意识到，这家伙原来把句子单词的顺序翻转了，正确的句子应该是“I am a student.”。Cat 对一一的翻转这些单词顺序可不在行，你能帮助他么？

public String ReverseSentence(String str)

调用旋转子函数 reverse(char[] ch,int start,int end)。

先旋转 0~length-1，再遍历旋转 blank+1~i-1,最后旋转 blank+1~length-1。

题目描述 45 纸牌抽顺子（两张大小王为任意数）

LL 今天心情特别好,因为他去买了一副扑克牌,发现里面居然有 2 个大王,2 个小王(一副牌原本是 54 张^_^)...他随机从中抽出了 5 张牌,想测测自己的手气,看看能不能抽到顺子,如果抽到的话,他决定去买体育彩票,嘿嘿!! “红心 A,黑桃 3,小王,大王,方片 5”,“Oh My God!”不是顺子.....LL 不高兴了,他想了想,决定大\小 王可以看成任何数字,并且 A 看作 1,J 为 11,Q 为 12,K 为 13。上面的 5 张牌就可以变成“1,2,3,4,5”(大小王分别看作 2 和 4),“So Lucky!”。LL 决定去买体育彩票啦。 现在,要求你使用这幅牌模拟上面的过程,然后告诉我们 LL 的运气如何， 如果牌能组成顺子就输出 true，否则就输出 false。为了方便起见,你可以认为大小王是 0。

public boolean isContinuous(int [] numbers)

1. 除 0 外没有重复的数。
2. max-min<5。

设最小初始值为 14，最大初始值为-1，flag 为 0。

先判断是否在 0~13 间；若为 0 则继续，判断是否重复，再更新最大最小值进行判断。

题目描述 46 m 个小朋友，报数 n-1 的小朋友出，求最后剩下的小朋友

每年六一儿童节,牛客都会准备一些小礼物去看望孤儿院的小朋友,今年亦是如此。HF 作为牛客的资深元老,自然也准备了一些小游戏。其中,有个游戏是这样的:首先,让小朋友们围成一个大圈。然后,他随机指定一个数 m,让编号为 0 的小朋友开始报数。每次喊到 m-1 的那个小朋友要出列唱首歌,然后可以在礼品箱中任意的挑选礼物,并且不再回到圈中,从他的下一个小朋友开始,继续 0...m-1 报数....这样下去....直到剩下最后一个小朋友,可以不用表演,并且拿到牛客名贵的“名侦探柯南”典藏版(名额有限哦!!^_^)。请你

试着想下,哪个小朋友会得到这份礼品呢？(注：小朋友的编号是从 0 到 n-1)
public int LastRemaining_Solution(int n, int m)
新建索引 i=-1，步数 step=0，剩余人数 count=n。
count 不为 0 时循环。i>=n 时循环，若为-1 则继续，判断 temp 是否为 m。

题目描述 47 求 1 到 n 的和

求 1+2+3+...+n，要求不能使用乘除法、for、while、if、else、switch、case 等关键字及条件判断语句（A?B:C）。
public int Sum_Solution(int n)
需利用逻辑与的短路特性实现递归终止。

```
int sum=n;
boolean flag=(n>0)&&((sum+=Sum_Solution(n-1))>0);
return sum;
```

题目描述 48 求两个整数的和

写一个函数，求两个整数之和，要求在函数体内不得使用+、-、*、/四则运算符号。
public int Add(int num1,int num2)
两个数异或：相当于每一位相加，而不考虑进位；
两个数相与，并左移一位：相当于求得进位；
将上述两步的结果相加

```
while(num2!=0) {
    int sum=num1^num2;
    int carray=(num1&num2)<<1;
    num1=sum;
    num2=carray;
}
return num1;
```

题目描述 49 字符串转换成整数，不合法返回 0

将一个字符串转换成一个整数(实现 Integer.valueOf(string)的功能，但是 string 不符合数字要求时返回 0)，要求不能使用字符串转换整数的库函数。 数值为 0 或者字符串不是一个合法的数值则返回 0。
输入描述:
输入一个字符串,包括数字字母符号,可以为空。
public int StrToInt(String str)
新建 start 指针和 symbol 记录正负。

题目描述 50 返回任意重复的元素，元素数值均小于长度

在一个长度为 n 的数组里的所有数字都在 0 到 n-1 的范围内。 数组中某些数字是重复的，但不知道有几个数字是重复的。也不知道每个数字重复几次。请找出数组中任意一个重复的数字。 例如，如果输入长度为 7 的数组{2,3,1,0,2,5,3}，那么对应的输出是第一个重复的数字 2。
Parameters:
numbers: an array of integers

length: the length of array numbers
这里要特别注意~返回任意重复的一个，赋值 duplication[0]
Return value: true if the input is valid, and there are some duplications in the array number
otherwise false
public boolean duplicate(int numbers[],int length,int [] duplication)
题目里写了数组里数字的范围保证在 0 ~ n-1 之间，所以可以利用现有数组设置标志，当一个数字被访问过后，可以设置对应位上的数 + n，之后再遇到相同的数时，会发现对应位上的数已经大于等于 n 了，那么直接返回这个数即可。

题目描述 51 构建数组，B[i]= A[0]*A[1]*...*A[i-1]*A[i+1]*...*A[n-1]，不使用除法

给定一个数组 A[0,1,...,n-1],请构建一个数组 B[0,1,...,n-1],其中 B 中的元素 B[i]=A[0]*A[1]*...*A[i-1]*A[i+1]*...*A[n-1]。不能使用除法。
public int[] multiply(int[] A)
从前面乘一遍，从后面（用 temp 记录乘数）乘一遍。
for(int i=1;i<length;i++)
for(int j=length-2;j>=0;j--)

题目描述 52 判断与包括'.'、'*'的正则表达式是否匹配

请实现一个函数用来匹配包括'.'和'*'的正则表达式。模式中的字符'.'表示任意一个字符，而'*'表示它前面的字符可以出现任意次（包含 0 次）。 在本题中，匹配是指字符串的所有字符匹配整个模式。例如，字符串"aaa"与模式"a.a"和"ab*ac*a"匹配，但是与"aa.a"和"ab*a"均不匹配

题目描述 53 判断字符串是否表示数值

请实现一个函数用来判断字符串是否表示数值（包括整数和小数）。例如，字符串"+100","5e2","-123","3.1416"和"-1E-16"都表示数值。 但是"12e","1a3.14","1.2.3","+-5"和"12e+4.3"都不是。
正则表达式：
XY：表示 X 后面跟着 Y，这里 X 和 Y 分别是正则表达式的一部分
(X):子表达式，将 X 看做是一个整体
[abc] ：表示可能是 a，可能是 b，也可能是 c。
[^abc]：表示不是 a,b,c 中的任意一个
[a-zA-Z]：表示是英文字母
[0-9]:表示是数字
?：表示出现 0 次或 1 次
+：表示出现 1 次或多次
*：表示出现 0 次、1 次或多次

题目描述 54 找出字符流中第一个只出现一次的字符

请实现一个函数用来找出字符流中第一个只出现一次的字符。例如，当从字符流中只读出前两个字符"go"时，第一个只出现一次的字符是"g"。当从该字符流中读出前六个字符“google"时，第一个只出现一次的字符是"l"。

输出描述：

如果当前字符流没有存在出现一次的字符，返回#字符。

```
public void Insert(char ch)
public char FirstAppearingOnce()
```

新建 Stringbuffer 记录字符，新建 int[]记录出现次数，找次数时先转换成数组。

题目描述 55 链表的入口结点

给一个链表，若其中包含环，请找出该链表的环的入口结点，否则，输出 null。

```
public ListNode EntryNodeOfLoop(ListNode pHead)
```

第一步，找环中相汇点。分别用 slow,fast 指向链表头部，slow 每次走一步，fast 每次走二步，直到 slow==fast 找到在环中的相汇点。

第二步，找环的入口。接上步，当 slow==fast 时，fast 所经过节点数为 2x，slow 所经过节点数为 x,设环中有 n 个节点，fast 比 slow 多走一圈(n 圈也可以)有 2x=n+x;n=x;可以看出 slow 实际走了一个环的步数，再让 fast 指向链表头部，slow 位置不变，slow，fast 每次走一步直到 slow==fast；此时 slow 指向环的入口。

fast 和 slow 一起走，相交后，fast 返回远点并一起一步走，相交时返回。

题目描述 56 删除链表中重复的结点

在一个排序的链表中，存在重复的结点，请删除该链表中重复的结点，重复的结点不保留，返回链表头指针。 例如，链表 1->2->3->3->4->4->5 处理后为 1->2->5

```
public ListNode deleteDuplication(ListNode pHead)
```

新建 first 节点。并有 last 指向 first 和 cur 指向 pHead 进行遍历。

题目描述 57 中序遍历的下一个结点

给定一个二叉树和其中的一个结点，请找出中序遍历顺序的下一个结点并且返回。注意，树中的结点不仅包含左右子结点，同时包含指向父结点的指针。

```
public TreeLinkNode GetNext(TreeLinkNode pNode)
```

中序遍历：左根右。

当有右结点时，返回右结点中的最左结点。

当没有右结点时，寻找第一个当前节点是父节点左孩子的节点，返回其父结点。

题目描述 58 二叉树是否镜像

请实现一个函数，用来判断一颗二叉树是不是对称的。注意，如果一个二叉树同此二叉树的镜像是同样的，定义其为对称的。

```
boolean isSymmetrical(TreeNode pRoot)
```

调用 comRoot(TreeNode left,TreeNode right)用于判断左右子树是否相等，并其中递归。

题目描述 59 之字形打印二叉树

请实现一个函数按照之字形打印二叉树，即第一行按照从左到右的顺序打印，第二层按照从右至左的顺序打印，第三行按照从左到右的顺序打印，其他行以此类推。

public ArrayList<ArrayList<Integer> > Print(TreeNode pRoot)
采用两个堆来装奇数行和偶数行。根据 layer 的奇偶分别遍历，记得 layer 每次后++。

题目描述 60 上下左右打印二叉树

从上到下按层打印二叉树，同一层结点从左至右输出。每一层输出一行。
ArrayList<ArrayList<Integer> > Print(TreeNode pRoot)
方法一：层序遍历，使用 queue.size()来计数。
方法二：使用深度递归的方式，调用子函数 depth(TreeNode root,int depth,ArrayList<ArrayList<Integer>> res)，若为空则返回，若深度大于尺寸则新建一个队列，进行添加节点的值，并递归子函数遍历左右节点。

```
depth(pRoot,1,res);
private void depth(TreeNode root,int
depth,ArrayList<ArrayList<Integer>> res) {
    if(root==null) {
        return;
    }
    if(depth>res.size()) {
        res.add(new ArrayList<Integer>());
    }
    res.get(depth-1).add(root.val);
    depth(root.left,depth+1,res);
    depth(root.right,depth+1,res);
}
```

题目描述 61 序列化二叉树和反序列化二叉树

请实现两个函数，分别用来序列化和反序列化二叉树

```
String Serialize(TreeNode root)
TreeNode Deserialize(String str)
```

```
public class Solution {
    int index=-1;
    String Serialize(TreeNode root) {
        StringBuffer sb=new StringBuffer();
        if(root==null) {
            sb.append("#,");
            return sb.toString();
        }
        sb.append(root.val+",");
        sb.append(Serialize(root.left));
        sb.append(Serialize(root.right));
        return sb.toString();
  }
    TreeNode Deserialize(String str) {
        index++;
        int len=str.length();
        if(index>len) {
            return null;
        }
        String[] strr=str.split(",");
        TreeNode node=null;
        if(!strr[index].equals("#")) {
            node=new TreeNode(Integer.valueOf(strr[index]));
            node.left=Deserialize(str);
            node.right=Deserialize(str);
        }
        return node;
  }
}
```

题目描述 62* 二叉搜索树第 k 小的结点

给定一棵二叉搜索树，请找出其中的第 k 小的结点。例如，（5，3，7，2，4，6，8） 中，按结点数值大小顺序第三小结点的值为 4。

```
TreeNode KthNode(TreeNode pRoot, int k)
```

必须要对每一个递归调用返回值进行判断 `if(node != null){return node;}`，判断返回值是否为 `null`，如果为 `null` 就说明在没找到，要继续执行 `index++ ;` `if(index == k){...}`的寻找过程，

如果返回不为空，则说明在递归调用判断子节点的时候已经找到符合要求的节点了，则将找到的节点

逐层向上传递返回。直到返回到第一次调用 `KthNode` 的地方。

如果不对递归调用的返回值做判断，即不执行 if(node != null){return node;}，那所找到符合 要求的节点只能返回到上一层，不能返回到顶层（可以自己调试，然后观察方法栈的调用变化）

```
public class Solution {
    int index=0;
    TreeNode KthNode(TreeNode pRoot, int k) {
        if(pRoot!=null) {
            TreeNode node=KthNode(pRoot.left,k);
            if(node!=null) {
                return node;
            }
            index++;
            if(index==k) {
                return pRoot;
            }
            node=KthNode(pRoot.right,k);
            if(node!=null) {
                return node;
            }
        }
        return null;
    }
}
```

题目描述 63 数据流的中位数

如何得到一个数据流中的中位数？如果从数据流中读出奇数个数值，那么中位数就是所有数值排序之后位于中间的数值。如果从数据流中读出偶数个数值，那么中位数就是所有数值排序之后中间两个数的平均值。我们使用 Insert()方法读取数据流，使用 GetMedian()方法获取当前读取数据的中位数。

```
public void Insert(Integer num)
public Double GetMedian()
```

Count 初始化为 0，新建大根堆和小根堆。
count 为偶数时，进入大根堆，弹出，进入小根堆，count++（小根堆多）；
Count 为奇数时，进入小根堆，弹出，进入大根堆，count++；
Count 为奇数时，返回 Doublt（小根堆堆顶）。
Count 为偶数时，返回 Double（二者的和）/2，2 要放外面，否则空指针异常；

题目描述 64 所有滑动窗口里的最大值

给定一个数组和滑动窗口的大小，找出所有滑动窗口里数值的最大值。例如，如果输入数组{2,3,4,2,6,2,5,1}及滑动窗口的大小 3，那么一共存在 6 个滑动窗口，他们的最大值分别为{4,4,6,6,6,5}； 针对数组{2,3,4,2,6,2,5,1}的滑动窗口有以下 6 个：{[2,3,4],2,6,2,5,1}， {2,[3,4,2],6,2,5,1}， {2,3,[4,2,6],2,5,1}，{2,3,4,[2,6,2],5,1}， {2,3,4,2,[6,2,5],1}， {2,3,4,2,6,[2,5,1]}。

用一个双端队列保存索引，队列第一个位置保存当前窗口的最大值，当窗口滑动一次
1.判断当前最大值是否过期。

2.新增加的值从队尾开始比较，把所有比他小的值丢掉。

```
import java.util.ArrayList;
import java.util.LinkedList;
public class Solution {
    public ArrayList<Integer> maxInWindows(int [] num, int size) {
        ArrayList<Integer> res=new ArrayList<Integer>();
        if(num==null || num.length==0 || size<=0 ||
num.length<size) {
            return res;
        }
        LinkedList<Integer> qmax=new LinkedList<Integer>();
        for(int i=0;i<num.length;i++) {
            while(!qmax.isEmpty() && num[qmax.peekLast()]<num[i]) {
                qmax.pollLast();
            }
            qmax.add(i);
            if(qmax.peekFirst()==i-size) {
                qmax.pollFirst();
            }
            if(i>=size-1) {
                res.add(num[qmax.peekFirst()]);
            }
        }
        return res;
    }
}
```

题目描述 65 二维矩阵是否包括字符串路径

请设计一个函数，用来判断在一个矩阵中是否存在一条包含某字符串所有字符的路径。路径可以从矩阵中的任意一个格子开始，每一步可以在矩阵中向左，向右，向上，向下移动一个格子。如果一条路径经过了矩阵中的某一个格子，则之后不能再次进入这个格子。 例如 a b c e s f c s a d e e 这样的 3 X 4 矩阵中包含一条字符串"bcced"的路径，但是矩阵中不包含"abcb"路径，因为字符串的第一个字符 b 占据了矩阵中的第一行第二个格子之后，路径不能再次进入该格子。

```
public boolean hasPath(char[] matrix, int rows, int cols, char[]
str)
```

回溯算法：二次遍历，若该点为字符串起点，返回 true；

调用子函数 `private boolean helper(char[] matrix,int rows,int cols,int i,int j,char[] str,int k,boolean[] flag)`，当不属于字符串时记得最后置零。

题目描述 66 机器人可到达格子数，位数相加小于 k

地上有一个 m 行和 n 列的方格。一个机器人从坐标 0,0 的格子开始移动，每一次只能向左，右，上，下四个方向移动一格，但是不能进入行坐标和列坐标的数位之和大于 k 的格子。

例如，当 k 为 18 时，机器人能够进入方格（35,37），因为 3+5+3+7 = 18。但是，它不能进入方格（35,38），因为 3+5+3+8 = 19。请问该机器人能够达到多少个格子？

```
public int movingCount(int threshold, int rows, int cols)
```

回溯算法：调用子函数 `helper(int i,int j,int rows,int cols, int threshold, boolean[][] flag)`，以及计算位数和的子函数 `sumnum(int i)`。

定义树：

```
public class TreeNode {
    int val = 0;
    TreeNode left = null;
    TreeNode right = null;
    public TreeNode(int val) {
        this.val = val;
    }
}
```