牛客网华为机试练习题

动态规划问题详解

前言

在找工作笔试刷题的过程中，对于动态规划问题不熟悉，找了很多资料，最终发现知乎上的一个回答不错，这里对其进行简单总结。

原回答链接如下：https://www.zhihu.com/question/23995189

生活中的动态规划

先来看看生活中经常遇到的事吧——假设您是个土豪，身上带了足够的1、5、10、20、50、100元面值的钞票。现在您的目标是凑出某个金额w，需要用到尽量少的钞票。

依据生活经验，我们显然可以采取这样的策略：能用100的就尽量用100的，否则尽量用50的……依次类推。在这种策略下，666=6×100+1×50+1×10+1×5+1×1，共使用了10张钞票。

这种策略称为“贪心”：假设我们面对的局面是“需要凑出w”，贪心策略会尽快让w变得更小。能让w少100就尽量让它少100，这样我们接下来面对的局面就是凑出w-100。长期的生活经验表明，贪心策略是正确的。

但是，如果我们换一组钞票的面值，贪心策略就也许不成立了。如果一个奇葩国家的钞票面额分别是1、5、11，那么我们在凑出15的时候，贪心策略会出错：

- 15=1×11+4×1 （贪心策略使用了5张钞票）
- 15=3×5 （正确的策略，只用3张钞票）

为什么会这样呢？贪心策略错在了哪里？

鼠目寸光。

刚刚已经说过，贪心策略的纲领是：“尽量使接下来面对的w更小”。这样，贪心策略在w=15的局面时，会优先使用11来把w降到4；但是在这个问题中，凑出4的代价是很高的，必须使用4×1。如果使用了5，w会降为10，虽然没有4那么小，但是凑出10只需要两张5元。

在这里我们发现，贪心是一种**只考虑眼前情况**的策略。

那么，现在我们怎样才能避免鼠目寸光呢？

如果直接暴力枚举凑出w的方案，明显复杂度过高。太多种方法可以凑出w了，枚举它们的时间是不可承受的。我们现在来尝试找一下性质。

重新分析刚刚的例子。w=15时，我们如果取11，接下来就面对w=4的情况；如果取5，则接下来面对w=10的情况。我们发现这些问题都有相同的形式：“给定w，凑出w所用的最少钞票是多少张？”接下来，我们用f(n)来表示“凑出n所需的最少钞票数量”。

那么，如果我们取了11，最后的代价（用掉的钞票总数）是多少呢？

明显$cost = f(4) + 1 = 4 + 1 = 5$，它的意义是：利用11来凑出15，付出的代价等于f(4)加上自己这一张钞票。现在我们暂时不管f(4)怎么求出来。

依次类推，马上可以知道：如果我们用5来凑出15，cost就是$f(10) + 1 = 2 + 1 = 3$。

那么，现在w=15的时候，我们该取那种钞票呢？**当然是各种方案中，cost值最低的那一个**！

- 取11：$cost = f(4) + 1 = 4 + 1 = 5$
- 取5：$cost = f(10) + 1 = 2 + 1 = 3$
- 取1：$cost = f(14) + 1 = 4 + 1 = 5$

显而易见，cost值最低的是取5的方案。**我们通过上面三个式子，做出了正确的决策**！

这给了我们一个**至关重要**的启示——$f(n)$ 只与$f(n-1), f(n-5), f(n-11)$ 相关；更确切地说：

$$f(n) = min\{f(n-1), f(n-5), f(n-11)\} + 1 \tag{1}$$

这个式子是非常激动人心的。我们要求出$f(n)$，只需要求出几个更小的f值；既然如此，我们从小到大把所有的f(i)求出来不就好了？注意一下边界情况即可。

以n=15为例，说明过程：

- n=0，自然f(0) =0;
- n=1,f(1)=f(0)+1=1
- n=2,f(2)=f(1)+1=2
- n=3,f(3)=f(2)+1=2+1=3
- n=4,f(4)=f(3)+1=3+1=4
- n=5,f(5)有2种情况，
    - f(5)=f(4)+1=4+1=5，选5张1元的；
    - f(5)=f(0)+1=0+1=1,选一张5元的
    - 很明显，应当选择f(5)=f(0)+1=1,选一张5元的方案
- n=6,f(6)也有两种方案，
    - f(6)=f(5)+1=2,选一张1元和一张5元的，5元的先选
    - f(6)=f(1)+1=2,选一张1元和一张5元的，1元的先选
- n=7,f(7)=f(6)+1=2+1=3(选两张1元和一张5元的)，f(7)=f(2)+1=3(选2张1元的和一张5元的)
- n=8,f(8)=f(7)+1=3+1=4（选三张1元和一张5元的）f(8)=f(3)+1=4(选三张1元的和一张5元的)
- n=9,f(9)=f(8)+1=4+1=5(选四张1元和一张5元的)，f(9)=f(4)+1=5（选4张一元的和一张5元的）
- n=10,2种情况：
    - f(10)=f(9)+1=4+1=5（选五张1元和一张5元的）
    - f(10)=f(5)+1=1+1=2（选两张5元的）
    - 最终，f(10)=2
- n=11,3种情况：
    - f(11)=f(10)+1=2+1=3（选两张5元和一张1元的）
    - f(11)=f(6)+1=2+1=3（选两张5元的和一张1元的）
    - f(11)=f(0)+1=1(选1张11元的)
    - 最终，f(11)=1
- n=12,3种情况：
    - f(12)=f(11)+1=1+1=2（选一张11元和一张1元的）
    - f(12)=f（7）+1=3+1=4（选两张5元的和2张1元的）
    - f(12)=f(11)+1=1+1=2（选一张11元和一张1元的)
    - 最终，f(12)=2
- n=13,3种情况：
    - f(13)=f(12)+1=2+1=3（选一张11元和两张1元的）
    - f(13)=f（8）+1=4+1=5（选两张5元的和3张1元的）
    - f(13)=f(3)+1=3+1=4（选一张11元和三张1元的)
    - 最终，f(13)=3
- n=14,3种情况：
    - f(14)=f(13)+1=3+1=4（选一张11元和三张1元的）
    - f(14)=f（9）+1=4+1=5（选两张5元的和四张1元的）
    - f(14)=f(3)+1=3+1=4（选一张11元和三张1元的)
    - 最终，f(14)=4
- n=15,3种情况：

- f(15)=f(14)+1=4+1=5（选一张11元和四张1元的）
- f(15)=f（10）+1=2+1=3（选三张5元的）
- f(15)=f(4)+1=4+1=5（选一张11元和四张1元的)
- 最终，f(15)=3

我们以$O(n)$ 的复杂度解决了这个问题。现在回过头来，我们看看它的原理：

- !$f(n)$ 只与$f(n-1), f(n-5), f(n-11)$的值相关。
- 我们只关心$f(w)$ 的**值**，不关心是怎么凑出w的。

这两个事实，保证了我们做法的正确性。它比起贪心策略，会分别算出取1、5、11的代价，从而做出一个正确决策，这样就避免掉了“鼠目寸光”！

它与暴力的区别在哪里？我们的暴力枚举了“使用的硬币”，然而这属于冗余信息。我们要的是答案，根本不关心这个答案是怎么凑出来的。**譬**如，要求出f(15)，只需要知道f(14),f(10),f(4)的值。**其他信息并不需要。**我们舍弃了冗余信息。我们只记录了对解决问题有帮助的信息——f(n).

我们能这样干，取决于问题的性质：求出f(n)，只需要知道几个更小的f(c)。**我们将求解f(c)称作求解f(n)的“子问题”。**

这就是DP（动态规划，dynamic programming）.

将一个问题拆成几个子问题，分别求解这些子问题，即可推断出大问题的解。

思考题：请稍微修改代码，输出我们凑出w的**方案。**

2. 几个简单的概念

【无后效性】

一旦f(n)确定，“我们如何凑出f(n)”就再也用不着了。

要求出f(15)，只需要知道f(14),f(10),f(4)的值，而f(14),f(10),f(4)是如何算出来的，对之后的问题没有影响。

“未来与过去无关”，这就是**无后效性。**

（严格定义：如果给定某一阶段的状态，则在这一阶段以后过程的发展不受这阶段以前各段状态的影响。）

【最优子结构】

回顾我们对f(n)的定义：我们记“凑出n所需的**最少**钞票数量”为f(n).

f(n)的定义就已经蕴含了“最优”。利用w=14,10,4的**最优**解，我们即可算出w=15的**最优**解。

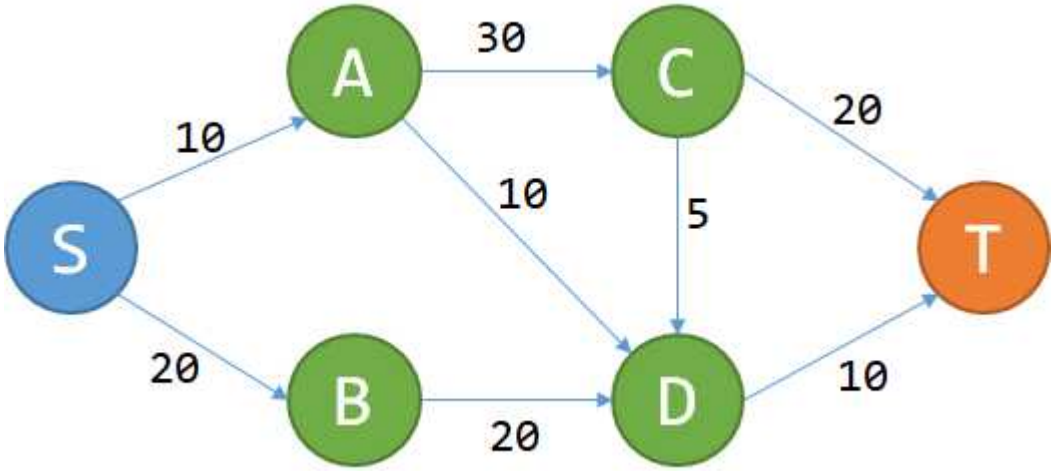
大问题的**最优解**可以由小问题的**最优解**推出，这个性质叫做“最优子结构性质”。

引入这两个概念之后，我们如何判断一个问题能否使用DP解决呢？

能将大问题拆成几个小问题，且满足无后效性、最优子结构性质。

3. DP的典型应用：DAG最短路

问题很简单：给定一个城市的地图，所有的道路都是单行道，而且不会构成环。每条道路都有过路费，问您从S点到T点花费的最少费用。



一张地图。边上的数字表示过路费。

这个问题能用DP解决吗？我们先试着记从S到P的最少费用为f(P).

想要到T，要么经过C，要么经过D。从而$f(T)=min\{f(C)+20,f(D)+10\}$![.

好像看起来可以DP。现在我们检验刚刚那两个性质：

- 无后效性：对于点P，一旦f(P)确定，以后就只关心f(P)的值，不关心怎么去的。
- 最优子结构：对于P，我们当然只关心到P的最小费用，即f(P)。如果我们从S走到T是$S\rightarrow P\rightarrow Q\rightarrow T$，那肯定S走到Q的最优路径是$S\rightarrow P\rightarrow Q$）。对一条最优的路径而言，从S走到**沿途上所有的点（子问题）**的最优路径，都是这条大路的一部分。这个问题的最优子结构性质是显然的。

既然这两个性质都满足，那么本题可以DP。式子明显为：

$$f(P)=min\{f(R)+w_{R\rightarrow P}\} \tag{2}$$

其中R为有路通到P的所有的点，$w_{R\rightarrow P}$为R到P的过路费。

手动分析过程如下：

- f(S)=0
- f(A)=f(S)+10=10;
- f(B)=f(S)+20=20;
- f(C)=f(A)+30=10+30=40;
- f(D)=min(f(A)+10,f(C)+5,f(B)+20)=min(20,45,40)=20,
- f(T)=min(f(C)+20,f(D)+10)=min(60,30)=30

4. 对DP原理的一点讨论

【DP的核心思想】

DP为什么会快？　　无论是DP还是暴力，我们的算法都是在**可能解空间**内，寻找**最优解**。

来看钞票问题。暴力做法是枚举所有的可能解，这是最大的可能解空间。　　DP是枚举**有希望成为答案的解**。这个空间比暴力的小得多。

也就是说：DP自带剪枝。

DP舍弃了一大堆不可能成为最优解的答案。譬如： 15 = 5+5+5 被考虑了。 15 = 5+5+1+1+1+1+1 从来没有考虑过，因为这不可能成为最优解。

从而我们可以得到DP的核心思想：**尽量缩小可能解空间。**

在暴力算法中，可能解空间往往是指数级的大小；如果我们采用DP，那么有可能把解空间的大小降到多项式级。

一般来说，解空间越小，寻找解就越快。这样就完成了优化。

【DP的操作过程】

一言以蔽之：**大事化小，小事化了。**

将一个大问题转化成几个小问题； 求解小问题； 推出大问题的解。

【如何设计DP算法】

下面介绍比较通用的设计DP算法的步骤。

首先，把我们面对的**局面**表示为x。这一步称为**设计状态。** 对于状态x，记我们要求出的答案(e.g. 最小费用)为f(x).我们的目标是求出f(T). **找出f(x)与哪些局面有关（记为p）**，写出一个式子（称为**状态转移方程**），通过f(p)来推出f(x).

【DP三连】

设计DP算法，往往可以遵循DP三连：

我是谁？——设计状态，表示局面 我从哪里来？ 我要到哪里去？——设计转移

设计状态是DP的基础。接下来的设计转移，有两种方式：一种是考虑我从哪里来（本文之前提到的两个例子，都是在考虑“我从哪里来”）；另一种是考虑我到哪里去，这常见于求出f(x)之后，**更新能从x走到的一些解**。这种DP也是不少的，我们以后会遇到。

总而言之，“我从哪里来”和“我要到哪里去”只需要考虑清楚其中一个，就能设计出状态转移方程，从而写代码求解问题。前者又称pull型的转移，后者又称push型的转移。（这两个词是

妹妹告诉我的，不知道源出处在哪）

思考题：如何把钞票问题的代码改写成“我到哪里去”的形式？ 提示：求出f(x)之后，更新f(x+1),f(x+5),f(x+11).

5. 例题：最长上升子序列

扯了这么多形而上的内容，还是做一道例题吧。

最长上升子序列（LIS）问题：给定长度为n的序列a，从a中抽取出一个子序列，这个子序列需要单调递增。问最长的上升子序列（LIS）的长度。 e.g. 1,5,3,4,6,9,7,8的LIS为1,3,4,6,7,8，长度为6。

如何设计状态（我是谁）？

我们记$f(x)$为以a_x 结尾的LIS长度，那么答案就是$max\{f(x)\}$

状态x从哪里推过来（我从哪里来）？

考虑比x小的每一个p：如果$a_x > a_p$ ，那么f(x)可以取f(p)+1.　　解释：我们把a_x 接在a_p 的后面，肯定能构造一个以a_x 结尾的上升子序列，长度比以a_p 结尾的LIS大1.那么，我们可以写出状态转移方程了：

$$f(x) = \max_{p<x, a_p<a_x} \{f(p)\} + 1 \tag{3}$$

至此解决问题。两层for循环，复杂度$O(n^2)$

手动推导过程如下：

- a=1时，因为a最小，所以f(1)=1
- a=5时，f(5)=f(1)+1=2
- a=3时，因为比5小，所以只能f(3)=f(1)+1=2
- a=4时，因为比5小，比3大，所以f(4)=max(f(1)+1,f(3)+1)=max(2,3)=3
- a=6时，因为目前是最大的，所以f(6)=max(f(1)+1,f(5)+1,f(3)+1,f(4)+1)=max(2,3,3,4)=4
- a=9时，因为是目前最大的，所以f(9)=max(f(1)+1,f(5)+1,f(3)+1,f(4)+1,f(6)+1)=max(2,3,3,4,5)=5
- a=7时，因为仅比9小，所以f(7)=max(f(1)+1,f(5)+1,f(3)+1,f(4)+1,f(6)+1)=max(2,3,3,4,5)=5
- a=8时，因为仅比9小，所以f(8)=max(f(1)+1,f(5)+1,f(3)+1,f(4)+1,f(6)+1,f(7)+1)=max(2,3,3,4,5,6)=6

所以，最长上升字串元素个数是6，对应的字串是1,3,4,6,7,8

下面，针对列出的2个实例，给出java版的解决方案

牛客网-华为机试练习题 01

题目描述

计算字符串最后一个单词的长度，单词以空格隔开。

输入描述:

一行字符串，非空，长度小于5000。

输出描述:

整数N，最后一个单词的长度。

示例1

输入

```
hello world
```

输出

```
5
```

解决代码

```
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String s="";
        while(input.hasNextLine()){
            s=input.nextLine();
            System.out.println(s.length()-1-s.lastIndexOf(" "));
        }
    }
}
```

总结

- 如果需要接受输入，就引用java.util.Scanner,然后实例化,Scanner input = new Scanner(System.in);
- 判断是否输入完，input.hasNextLine()
- 获取每一行输入，s=input.nextLine();
- 获取字符串中某个元素的最后一次出现的位置，lastIndexOf()

牛客网-华为机试练习题 02

题目描述

写出一个程序，接受一个由字母和数字组成的字符串，和一个字符，然后输出输入字符串中含有该字符的个数。不区分大小写。

输入描述:

第一行输入一个有字母和数字以及空格组成的字符串，第二行输入一个字符。

输出描述:

输出输入字符串中含有该字符的个数。

示例1

输入

```
ABCDEF
A
```

输出

```
1
```

解决代码

```
import java.util.Scanner;
public class Main{
    public static void main(String[] args)
        {
            int count = 0;
            Scanner sc=new Scanner(System.in);
            String str = sc.nextLine();
            String br = sc.nextLine();

            String string1=str.toUpperCase();
```

```
            String br1=br.toUpperCase();

            char a = br1.charAt(0);

            for(int i=0;i<string1.length();i++)
            {
                if(string1.charAt(i)==a)
                    count++;
            }
            System.out.println(count);

        }
}
```

总结：

- 如果需要导入，那么首先导入java.util.Scanner库，然后实例化，Scanner sc = new Scanner(System.in);
- public class Main后面不需要加小括号，
- 字符串转大写使用toUpperCase()方法
- 字符串按照序号寻找字符使用charAt（ ）方法
- 如果报下列错误，检测main函数是否出现问题，main全部小写
    - 请检查是否存在数组越界等非法访问情况 case通过率为0.00%

牛客网-华为机试练习题 03

题目描述

明明想在学校中请一些同学一起做一项问卷调查，为了实验的客观性，他先用计算机生成了N个1到1000之间的随机整数（N≤1000），对于其中重复的数字，只保留一个，把其余相同的数去掉，不同的数对应着不同的学生的学号。然后再把这些数从小到大排序，按照排好的顺序去找同学做调查。请你协助明明完成“去重”与“排序”的工作(同一个测试用例里可能会有多组数据，希望大家能正确处理)。

```
Input Param
n               输入随机数的个数
inputArray      n个随机整数组成的数组

Return Value
OutputArray     输出处理后的随机整数
```

注：测试用例保证输入参数的正确性，答题者无需验证。测试用例不止一组。

输入描述:

输入多行，先输入随机整数的个数，再输入相应个数的整数

输出描述:

返回多行，处理后的结果

示例1

输入

```
11
10
```

```
20
40
32
67
40
20
89
300
400
15

输出
10
15
20
32
40
67
89
300
400
```

解决代码：

```
import java.util.Iterator;
import java.util.Scanner;
import java.util.TreeSet;

public class Main {
    public static void main(String[] args) {
        Scanner str  = new Scanner(System.in);
        while(str.hasNextInt()){
            int cn = str.nextInt();
            TreeSet<Integer> ts = new TreeSet<Integer>();
            while(cn-->0 && str.hasNextInt()){
                ts.add(str.nextInt());
            }
            Iterator<Integer> it = ts.iterator();
            while(it.hasNext()){
                System.out.println(it.next());
            }
        }
    }
}
```

总结：

- Treeset的使用
    - 引入，java.util.TreeSet
    - 实例化，TreeSet<Integer> ts = new TreeSet<Integer>()；
    - 添加元素，ts.add()
- Interator的使用
    - 判断是否有下一个，hasNext()
    - 获得下一个，nextInt();
- Scanner的使用
    - 引入，java.util.Scanner；
    - 实例化，Scanner str = new Scanner(System.in)
    - 判断是否有整数，str.hasNextInt()

- 获取下一个整数，str.NextInt()

牛客网-华为机试练习题 04

题目描述

连续输入字符串，请按长度为8拆分每个字符串后输出到新的字符串数组； 长度不是8整数倍的字符串请在后面补数字0，空字符串不处理。

输入描述:

连续输入字符串(输入2次,每个字符串长度小于100)

输出描述:

输出到长度为8的新字符串数组

示例1

输入

```
abc
123456789
```

输出

```
abc00000
12345678
90000000
```

解决代码

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scanner=new Scanner(System.in);
        while(scanner.hasNextLine()){
            String s=scanner.nextLine();
            split(s);
        }
    }

    public static void split(String s){
        while(s.length()>=8){
            System.out.println(s.substring(0, 8));
            s=s.substring(8);
        }
        if(s.length()<8&&s.length()>0){
            s=s+"00000000";
            System.out.println(s.substring(0, 8));
        }
    }


}
```

总结：

首先判断字符串长度是否大于8，如果是，则打印substring(0,8)，然后将substring之后的字符串给s，直至s的长度小于8

这个时候，再将s和8个0组成的字符串相加，

牛客网-华为机试练习题 05

题目描述

写出一个程序，接受一个十六进制的数值字符串，输出该数值的十进制字符串。（多组同时输入 ）

输入描述:

输入一个十六进制的数值字符串。

输出描述:

输出该数值的十进制字符串。

示例1

输入

0xA

输出

10

思路

- 从后往前遍历
- 获得每个位置上的字符对应的数值，然后乘以该字符对应的权值，索引值越小权值越大，索引值越大，权值越小
- 在获得每个位置上字符对应的数值的时候，0-9就直接减去字符0，a-z减去字符z，A-Z减去字符Z

解决代码

```
import java.lang.Math;
import java.util.Scanner;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);

        while(sc.hasNext()){
            String str = sc.nextLine();
            System.out.println(fun(str.substring(2)));
        }
    }

    public static int fun(String s){
        int n=0;
        int count= 0;
        int temp = 0;
```

```
        char ch;

        while(count<s.length())
        {
            ch = s.charAt(s.length()-count-1);
            if(ch>='0'&&ch<='9'){
                temp = ch-'0';
            }else if(ch>='A'&&ch<='Z'){
                temp = ch-'A'+10;
            }else if(ch>='a'&&ch<='z'){
                temp = ch-'a'+10;
            }else{
                break;
            }
            n += temp*Math.pow(16,count);
            count++;
        }

        return n;
    }
}
```

总结：

- 首先，使用substring(2)去掉前面的0x
- 然后使用s.length()-count-1,从最右边的一个数字开始进行遍历，注意先去遍历的顺序
- 对每个字母，如果是0-9，那么就将其减去‘0’，获得它的值，如果是在‘A'和’Z'之间，就减去‘A'并加上10，如果是’a'，同样处理。
- 然后乘以每个位的基，math.pow(16,count)
- Math导入的方法是import java.lang.Math
- 对于循环的问题，首先应该明确，开始条件，结束条件，自增的条件，注意不要忘记count++

牛客网-华为机试练习题 06

题目描述

功能:输入一个正整数，按照从小到大的顺序输出它的所有质数的因子（如180的质数因子为2 2 3 3 5 ）最后一个数后面也要有空格

详细描述：

函数接口说明：

```
public String getResult(long ulDataInput)
```

输入参数：

`long ulDataInput`：输入的正整数

返回值：

`String`

输入描述:

输入一个long型整数

输出描述:

按照从小到大的顺序输出它的所有质数的因子，以空格隔开。最后一个数后面也要有空格。

示例1

输入

180

输出

2 2 3 3 5

解决代码：

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner str = new Scanner(System.in);
        long num = str.nextLong();
        String result = getResult(num);
        System.out.println(result);
    }

public static String getResult(long num){
    int pum = 2;
    String result = "";
    while(num!=1){
        while(num%pum==0){
            num=num/pum;
            result = result + pum+" ";
        }
        pum++;
    }
    return result;
}
}
```

总结

- 思路如下：
- 1.利用Scanner类得到一个长整数sc.nextLong()；
- 2.判断得到的正整数是否是一个质数，否就执行下面步骤，
- (1).分解质因数
- (2).定义一个集合，用于装分解出来的因数，
- (3).对合数取商数和余数
- (4).对余数进行判断是否是质数，是就跳出循环，否就执行循环体，直至是质数
- 3.遍历集合，打印出质因数。
- 这里不考虑4,6,9等的原因是，如果可以被4整除，那么肯定可以被2整除，其余同理。所以不必担心这一点

牛客网-华为机试练习题 07

题目描述

写出一个程序，接受一个正浮点数值，输出该数值的近似整数值。如果小数点后数值大于等于5,向上取整；小于5，则向下取整。

输入描述:

输入一个正浮点数值

输出描述:

输出该数值的近似整数值

示例1

输入

```
5.5
```

输出

```
6
```

解决代码

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner=new Scanner(System.in);
        double d=scanner.nextDouble();
        System.out.println(getReturn(d));
    }

    public static int getReturn(double d) {
        int i=(int)d;
        return  (d-i)>=0.5?i+1:i;
    }
}
```

总结：

- 双精度浮点型是double,不是Double
- 强制类型转换是(int)

牛客网-华为机试练习题 08

题目描述

数据表记录包含表索引和数值，请对表索引相同的记录进行合并，即将相同索引的数值进行求和运算，输出按照key值升序进行输出。

输入描述:

先输入键值对的个数
然后输入成对的index和value值，以空格隔开

输出描述:

输出合并后的键值对（多行）

示例1

输入

```
4
0 1
0 2
1 2
3 4
```

输出

```
0 3
1 2
3 4
```

解决代码：

```
import java.util.Scanner;
import java.util.SortedMap;
import java.util.TreeMap;

public class Main {
    public static void main(String[] args) {
        Scanner str = new Scanner(System.in);
        SortedMap<Integer,Integer> map = new TreeMap<>();
        int n = Integer.parseInt(str.nextLine());
        for (int i = 0;i<n;i++){
            String[] mid = str.nextLine().split("\\s+");
            addPare(map,mid);
        }
        System.out.println(mapToString(map));
    }

    private static String mapToString(SortedMap<Integer, Integer> map) {
        // TODO Auto-generated method stub
        StringBuilder builder = new StringBuilder();
        for(SortedMap.Entry<Integer,Integer>e:map.entrySet()){
            builder.append(e.getKey()).append(" ").append(e.getValue()).append("\r");
        }
        return builder.toString();
    }

    private static void addPare(SortedMap<Integer, Integer> map, String[] mid) {
        // TODO Auto-generated method stub
        int key = Integer.parseInt(mid[0]);
        int value = Integer.parseInt(mid[1]);
        if(map.containsKey(key)){
            map.put(key, map.get(key) + value);
        }else{
            map.put(key, value);
        }
    }
}
```

牛客网-华为机试练习题 09

题目描述

输入一个int型整数，按照从右向左的阅读顺序，返回一个不含重复数字的新的整数。

输入描述:

输入一个int型整数

输出描述:

按照从右向左的阅读顺序，返回一个不含重复数字的新的整数

示例1

输入

```
9876673
```

输出

```
37689
```

解决代码：

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        @SuppressWarnings("resource")
        Scanner scanner=new Scanner(System.in);
        while(scanner.hasNext()){
            String s=scanner.nextLine();
            int len=s.length();
            int[]arr1=new int[10];
            //这里的意思是一共有10个数字
            for(int i=len-1;i>=0;i--){
                //i=len-1,表示从右往左进行遍历，
                if(arr1[s.charAt(i)-48]==0){
                    System.out.print(s.charAt(i)-48);
                    arr1[s.charAt(i)-48]++;
                }
            }
        }
    }
}
```

总结：

- 思路是，创建容量为10的数组
- 将每个字符减去48，作为索引，如果不存在即对应位置的值是0，那么就打印该索引，否则，将对应位置的值自加1.

牛客网-华为机试练习题 10

题目描述

编写一个函数，计算字符串中含有的不同字符的个数。字符在ACSII码范围内(0~127)。不在范围内的不作统计。

输入描述:

输入N个字符，字符在ACSII码范围内。

输出描述:

输出范围在(0~127)字符的个数。

示例1

输入

abc

输出

3

解决代码:

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner=new Scanner(System.in);
        while(scanner.hasNext()){
            String s=scanner.nextLine();
            int len=getLen(s);
            System.out.println(len);
        }
    }

public static int  getLen(String s) {
    int[] arr=new int[128];
    for(int i=0;i<s.length();i++){
        arr[s.charAt(i)]=1;
    }
    int len=0;
    for (int i = 0; i < arr.length; i++) {
        if(arr[i]==1){
            len++;
        }
    }
    return len;
}
}
```

总结：

- 遍历每个字符，放入数组中的对应位置，对应位置的值自加1
- 遍历数组，值大于的0 的索引的总个数就是最终的结果

牛客网-华为机试练习题 100

题目描述

自守数是指一个数的平方的尾数等于该数自身的自然数。例如：25^2 = 625，76^2 = 5776，9376^2 = 87909376。请求出n以内的自守数的个数

接口说明

/* 功能: 求出n以内的自守数的个数

输入参数： int n

返回值： n以内自守数的数量。 */

public static int CalcAutomorphicNumbers(int n) { /*在这里实现功能*

return 0; }

输入描述:

int型整数

输出描述:

n以内自守数的数量。

示例1

输入

2000

输出

8

解决代码

```
import java.util.*;
import java.io.*;

public class Main{
    public static void main(String[] args) throws IOException{
        BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        String str = "";
        while((str=bf.readLine()) != null){
            int n = Integer.parseInt(str);
            int count = 0;
            for(int i = 0; i<=n; i++){
                int res = calAutomopphicNumbers(i);
                count = count + res;
            }
            System.out.println(count);
        }
    }

    public static int calAutomopphicNumbers(int n){
        String str = Integer.toString(n);
        String str2 = Integer.toString(n*n);

        if(str.equals(str2.substring(str2.length()-str.length()))){
```

```
            return 1;
        }else{
            return 0;
        }
    }
}
```

牛客网-华为机试练习题 101

题目描述

功能:等差数列 2，5，8，11，14。。。。

输入:正整数N >0

输出:求等差数列前N项和

返回:转换成功返回 0 ,非法输入与异常返回-1

输入描述:

输入一个正整数。

输出描述:
输出一个相加后的整数。

示例1
输入

2
输出

7

解决代码

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {

    public static void main(String[] args) throws NumberFormatException, IOException {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String message=null;
        while((message=br.readLine())!=null){
            int n=Integer.parseInt(message);
            int res=n*2+n*(n-1)*3/2;
            System.out.println(res);
        }
    }
}
```

牛客网-华为机试练习题 102

题目描述

输入整型数组和排序标识，对其元素按照升序或降序进行排序（一组测试用例可能会有多组数据）

接口说明

原型：

void sortIntegerArray(Integer[] pIntegerArray, int iSortFlag);

输入参数：

Integer[] pIntegerArray：整型数组

int iSortFlag：排序标识：0表示按升序，1表示按降序

输出参数：

无

返回值：

void

输入描述:

1、输入需要输入的整型数个数

输出描述:

输出排好序的数字

示例1

输入

```
8
1 2 4 9 3 55 64 25
0
```

输出

```
1 2 3 4 9 25 55 64
```

解决代码:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Arrays;

public class Main {

    public static void main(String[] args) throws IOException {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String message=null;
        while((message=br.readLine())!=null){
            int n=Integer.parseInt(message);
            String[] data=br.readLine().split(" ");
```

```
            int m=Integer.parseInt(br.readLine());
            int[] arr=new int[n];
            for(int i=0;i<data.length;i++){
                arr[i]=Integer.parseInt(data[i]);
            }
            StringBuffer sb=new StringBuffer();
            if(m==0){
                Arrays.sort(arr);
            }else{
                Arrays.sort(arr);
                for(int i=0;i<arr.length/2;i++){
                    int a=arr[i];
                    arr[i]=arr[arr.length-1-i];
                    arr[arr.length-1-i]=a;
                }
            }
            for(int i=0;i<arr.length;i++){
                sb.append(arr[i]+" ");
            }
            System.out.println(sb.toString().trim());
        }
    }
}
```

牛客网-华为机试练习题 103

题目描述

如果统计的个数相同，则按照ASCII码由小到大排序输出 。如果有其他字符，则对这些字符不用进行统计。

实现以下接口： 输入一个字符串，对字符中的各个英文字符，数字，空格进行统计（可反复调用） 按照统计个数由多到少输出统计结果，如果统计的个数相同，则按照ASII码由小到大排序输出 清空目前的统计结果，重新统计 调用者会保证： 输入的字符串以'\0'结尾。

输入描述:

输入一串字符。

输出描述:

对字符中的
各个英文字符（大小写分开统计），数字，空格进行统计，并按照统计个数由多到少输出,如果统计的个数相同，则按照ASII码由小到大排序输出 。如果有其他字符，则对这些字符不用进行统计。

示例1

输入

```
aadddccddc
```

输出

```
dca
```

解决代码:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String line=null;
        while((line=br.readLine())!=null){
            System.out.println(fun(line));
        }
    }

    private static String fun(String str) {
        char[] chs=str.toCharArray();
        int[] num=new int[200];     //必须大于128
        for (char c : chs) {
            int i=(int)c;
            num[i]++;
        }

        int max=0;
        for(int i=0;i<num.length;i++){
            if(max<num[i]){
                max=num[i];
            }
        }

        StringBuffer sb=new StringBuffer();

        while(max!=0){
            for(int i=0;i<num.length;i++){
                if(max==num[i]){
                    sb.append((char)i);
                }
            }
            max--;
        }
        return sb.toString();
    }
}
```

牛客网-华为机试练习题 104

题目描述

题目描述

Redraiment是走梅花桩的高手。Redraiment总是起点不限，从前到后，往高的桩子走，但走的步数最多，不知道为什么？你能替Redraiment研究他最多走的步数吗？

样例输入

6

2 5 1 5 4 5

样例输出

3

提示

Example: 6个点的高度各为 2 5 1 5 4 5 如从第1格开始走,最多为3步, 2 4 5 从第2格开始走,最多只有1步,5 而从第3格开始走最多有3步,1 4 5 从第5格开始走最多有2步,4 5

所以这个结果是3。

接口说明

方法原型：

int GetResult(int num, int[] pInput, List pResult);

输入参数： int num：整数，表示数组元素的个数（保证有效）。 int[] pInput: 数组，存放输入的数字。

输出参数： List pResult: 保证传入一个空的List，要求把结果放入第一个位置。 返回值： 正确返回1，错误返回0

输入描述:

输入多行，先输入数组的个数，再输入相应个数的整数

输出描述:

输出结果

示例1

输入

```
6
2
5
1
5
4
5
```

输出

```
3
```

解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {

    public static void main(String[] args) throws IOException {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String message=null;
        while((message=br.readLine())!=null){
            int n=Integer.parseInt(message);
            String[] data=br.readLine().split(" ");
            int[] arr=new int[n];
```

```
            for(int i=0;i<data.length;i++){
                arr[i]=Integer.parseInt(data[i]);
            }
            int[] brr=new int[n];
            for(int i=0;i<arr.length;i++){
                brr[i]=1;
                for(int j=0;j<i;j++){
                    if(arr[j]<arr[i]){
                        brr[i]=Math.max(brr[i], brr[j]+1);
                    }
                }
            }
            int max=0;
            for(int i=0;i<brr.length;i++){
                if(max<brr[i]){
                    max=brr[i];
                }
            }
            System.out.println(max);
        }
    }
}
```

牛客网-华为机试练习题 105

题目描述

连续输入字符串(输出次数为N,字符串长度小于100)，请按长度为8拆分每个字符串后输出到新的字符串数组，

长度不是8整数倍的字符串请在后面补数字0，空字符串不处理。

首先输入一个整数，为要输入的字符串个数。

例如：

输入：2

abc

12345789

输出：abc00000

12345678

90000000

接口函数设计如下:

/************************ 功能:存储输入的字符创

输入:字符串

输出:无
返回:0表示成功,其它返回-1 ************************/

int AddString(char *strValue); /********************** 功能:获取补位后的二维数组的长度

输入:无

输出:无
返回:二维数组长度 **********/

int GetLength();

/********** 功能:将补位后的二维数组，与输入的二维数组做比较

输入:strInput:输入二维数组,iLen：输入的二维数组的长度

输出:无
返回:若相等,返回0;不相等,返回-1.其它:-1; **********/ int ArrCmp(char strInput[][9],int iLen);

输入描述:

首先输入数字n，表示要输入多少个字符串。连续输入字符串(输出次数为N,字符串长度小于100)。

输出描述:

按长度为8拆分每个字符串后输出到新的字符串数组，长度不是8整数倍的字符串请在后面补数字0，空字符串不处理。

示例1

输入

```
2
abc
123456789
```

输出

```
abc00000
12345678
90000000
```

解决代码:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (in.hasNextLine()) {
            int count = Integer.valueOf(in.nextLine());
            for (int i = 0; i < count; i++) {
                String input = in.nextLine();
                if (input == null || input.length() == 0) {
                    continue;
                }
                StringBuilder sb = new StringBuilder(input.trim());
                if (sb.length() % 8 != 0) {
                    int more = 8 - sb.length() % 8;
                    for (int j = 0; j < more; j++) {
                        sb.append("0");
                    }
                }
                for (int j = 0; j < sb.length(); j += 8) {
                    System.out.println(sb.substring(j, j + 8));
                }
```

```
            }
        }
    }
}
```

牛客网-华为机试练习题 106

题目描述

从输入任意个整型数，统计其中的负数个数并求所有非负数的平均值

输入描述:

输入任意个整数

输出描述:

输出负数个数以及所有非负数的平均值

示例1

输入

```
-13
-4
-7
```

输出

```
3
0.0
```

解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
public class Main {
    public static void main(String[] args)throws IOException{
        BufferedReader bReader = new BufferedReader(new InputStreamReader(System.in));
        String line=null;
        while((line=bReader.readLine())!=null) {
            String[] strary=line.split(" ");
            int negnum=0;
            int notnegnum=0;
            int total=0;
            for(int i=0;i<strary.length;i++){
                int x=Integer.valueOf(strary[i]);
                if(x<0) negnum++;
                else {
                    notnegnum++;
                    total+=x;
                }
            }
            System.out.println(negnum);
            System.out.format("%.1f", (float)total/notnegnum);
            System.out.prinln();
```

```
        }
    }
}
```

牛客网-华为机试练习题 107

题目描述

将一个字符串str的内容颠倒过来，并输出。str的长度不超过100个字符。 如：输入“I am a student”，输出“tneduts a ma I”。

输入参数：

inputString：输入的字符串

返回值：

输出转换好的逆序字符串

输入描述:

输入一个字符串，可以有空格

输出描述:

输出逆序的字符串

示例1

输入

```
I am a student
```

输出

```
tneduts a ma I
```

解决代码：

```
import java.io.*;

public class Main{

    public static void main(String[] args) throws Exception {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String str=br.readLine().toString();
        StringBuffer sb=new StringBuffer(str);
        System.out.println(sb.reverse());

    }

}
```

牛客网-华为机试练习题 108

题目描述

•计算一个数字的立方根，不使用库函数

详细描述：

•接口说明

原型：

public static double getCubeRoot(double input)

输入:double 待求解参数

返回值:double 输入参数的立方根，保留一位小数

输入描述:

待求解参数 double类型

输出描述:

输入参数的立方根 也是double类型

示例1

输入

216

输出

6.0

解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.math.RoundingMode;
import java.text.DecimalFormat;
import java.text.NumberFormat;

public class Main {
    public static boolean isTest = false;

    public static void main(String [] args) throws IOException{
        if(isTest) {
            new Main().test();
        } else {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            String str = null;
            do {
                str = br.readLine();
                if(str != null) {
                    double d = new Main().run(str);
                    String tempD = Double.toString(d);
                    tempD = tempD.substring(0, 4);
                    int last = Integer.parseInt(tempD.substring(3, 4));
                    if(last >= 5) {
                        d += 0.1;
```

```
                    }
                    tempD = Double.toString(d);
                    tempD = tempD.substring(0, 3);
                    System.out.println(tempD);
                }
            } while (str != null);
        }
    }

    public double run(String str) {
        double num = Double.parseDouble(str);
        double dis = 1.0;
        double start = 0.1;
        for(double i=0.1; dis > 0.0; i+=0.01) {
            double temp = i*i*i;
            dis = num-temp;
            start = i;
        }
        return start;
    }

    public void test() {
        String test = "11";
        double d = new Main().run(test);
        String tempD = Double.toString(d);
        tempD = tempD.substring(0, 4);
        int last = Integer.parseInt(tempD.substring(3, 4));
        if(last >= 5) {
            d += 0.1;
        }
        tempD = Double.toString(d);
        tempD = tempD.substring(0, 3);
        System.out.println(tempD);
    }
}
```

牛客网-华为机试练习题 109

题目描述

正整数A和正整数B 的最小公倍数是指 能被A和B整除的最小的正整数值，设计一个算法，求输入A和B的最小公倍数。

输入描述:

输入两个正整数A和B。

输出描述:

输出A和B的最小公倍数。

示例1

输入

```
5
7
```

输出

```
35
```

解决代码：

```
import java.util.Scanner;

public class Main {
    public static int getResult(int m ,int n){
        if(m<n){
            int temp=m;
            m=n;
            n=temp;
        }
        int k;
        while(n!=0){
            k=m%n;
            m=n;
            n=k;
        }
        return m;
    }
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);
        while(reader.hasNext()){
            int m = reader.nextInt();
            int n = reader.nextInt();
            System.out.println(m*n/getResult(m, n));
        }
    }

}
```

牛客网-华为机试练习题 11

题目描述

描述：

- 输入一个整数，将这个整数以字符串的形式逆序输出
- 程序不考虑负数的情况，若数字含有0，则逆序形式也含有0，如输入为100，则输出为001

输入描述:

输入一个int整数

输出描述:

将这个整数以字符串的形式逆序输出

示例1

输入

1516000

输出

0006151

解决代码：

```
import java.util.Stack;
import java.util.Scanner;

public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        Stack stack = new Stack();
        for (int i=0;i<str.length();i++){
            stack.push(str.charAt(i));
        }
        while(!stack.isEmpty()){
            System.out.print(stack.pop());
         }
    }
}
```

```
import java.util.*;
public class Main{
    public static void main(String[]args){
        Scanner scan=new Scanner(System.in);
        while(scan.hasNextLine()){
            char[]chars=scan.nextLine().toCharArray();
            for(int i=chars.length-1;i>=0;i--){
                    System.out.print(chars[i]);
            }
            System.out.println();
        }
    }


}
```

总结：

- 导入堆栈，java.util.Stack;
- 往栈中添加元素，stackpush()
- 遍历栈，判断是否为空，！stack.isEmpty(),
- 栈出元素，stack.pop()
- 字符串转数组，str.toCharArray()

牛客网-华为机试练习题 12

题目描述

写出一个程序，接受一个字符串，然后输出该字符串反转后的字符串。例如：

输入描述:

输入N个字符

输出描述:

输出该字符串反转后的字符串

示例1

输入

abcd

输出

dcba

解决代码：

```
import java.util.Stack;
import java.util.Scanner;

public class Main{
    public static void main(String[] args){

        Stack stack = new Stack();
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        for(int i=0;i<str.length();i++){
            stack.push(str.charAt(i));
        }
        while(!stack.isEmpty()){
            System.out.print(stack.pop());
        }
    }
}
```

牛客网-华为机试练习题 13

题目描述

将一个英文语句以单词为单位逆序排放。例如“I am a boy”，逆序排放后为“boy a am I” 所有单词之间用一个空格隔开，语句中除了英文字母外，不再包含其他字符

接口说明

```
/**
 * 反转句子
 *
 * @param sentence 原句子
 * @return 反转后的句子
 */
public String reverse(String sentence);
```

输入描述:

将一个英文语句以单词为单位逆序排放。

输出描述:

得到逆序的句子

示例1

输入

```
I am a boy
```

输出

```
boy a am I
```

解决代码

```
import java.util.Stack;
import java.util.Scanner;

public class Main{
    public static void main(String[] args){

        Stack stack = new Stack();
        Scanner sc = new Scanner(System.in);
        while(sc.hasNext()){
            String str = sc.nextLine();
            String[] s = str.split(" ");
            for(int i=0;i<s.length;i++){
                stack.push(s[i]);
            }
            while(!stack.isEmpty()){
                System.out.print(stack.pop()+" ");
            }
            System.out.println();
        }
    }
}
```

```
import java.util.*;
public class Main{
    public static void main(String[] args)
        {
        Scanner scan=new Scanner(System.in);
        String str=scan.nextLine();
        String res=reverse(str);
        System.out.println(res);
    }
    public static String reverse(String str)
        {
        String[] s=str.split(" ");
        StringBuilder sb=new StringBuilder();
        for(int i=s.length-1;i>0;i--)
            {
            sb.append(s[i]);
            sb.append(" ");
        }
```

```
            sb.append(s[0]);
            return sb.toString();
        }
    }
```

牛客网-华为机试练习题 14

题目描述

给定n个字符串，请对n个字符串按照字典序排列。

输入描述:

输入第一行为一个正整数n(1≤n≤1000),下面n行为n个字符串(字符串长度≤100),字符串中只含有大小写字母。

输出描述:

数据输出n行，输出结果为按照字典序排列的字符串。

示例1

输入

```
9
cap
to
cat
card
two
too
up
boat
boot
```

输出

```
boat
boot
cap
card
cat
to
too
two
up
```

解决代码：

```
import java.util.Arrays;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int num = sc.nextInt();
        String [] ss = new String[num];
        for (int i = 0; i < num; i++) {
            ss[i]=sc.next();
```

```
        }
        Arrays.sort(ss);
        for (int i = 0; i < ss.length; i++) {
            System.out.println(ss[i]);

       }
    }
}
```

总结：

- 排序可以直接用Arrays.sort方法
- 导入Arrays方法，java.util.Arrays
- 需要注意sc.next和sc.nextLine()的区别；

牛客网-华为机试练习题 15

题目描述

输入一个int型的正整数，计算出该int型数据在内存中存储时1的个数。

输入描述:

输入一个整数（int类型）

输出描述:

这个数转换成2进制后，输出1的个数

示例1

输入

5

输出

2

解决代码：

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args){
        Scanner sc  = new Scanner(System.in);
        int num = sc.nextInt();
        int count =0;
        while(num>0){
            num = num& num-1;
            count++;
        }
        System.out.println(count);
    }
}
```

牛客网-华为机试练习题 16

题目描述

王强今天很开心，公司发给N元的年终奖。王强决定把年终奖用于购物，他把想买的物品分为两类：主件与附件，附件是从属于某个主件的，下表就是一些主件与附件的例子：

主件	附件
电脑	打印机，扫描仪
书柜	图书
书桌	台灯，文具
工作椅	无

如果要买归类为附件的物品，必须先买该附件所属的主件。每个主件可以有 0 个、 1 个或 2 个附件。附件不再有从属于自己的附件。王强想买的东西很多，为了不超出预算，他把每件物品规定了一个重要度，分为 5 等：用整数 1 **~** 5 表示，第 5 等最重要。他还从因特网上查到了每件物品的价格（都是 10 元的整数倍）。他希望在不超过 N 元（可以等于 N 元）的前提下，使每件物品的价格与重要度的乘积的总和最大。

- 设第 j 件物品的价格为 v[j] ，重要度为 w[j] ，共选中了 k 件物品，编号依次为 j 1 ， j 2 ，……， j k ，则所求的总和为：

v[j 1]*w[j 1]+v[j 2]*w[j 2]+ … +v[j k]*w[j k] 。（其中 * 为乘号）

- 请你帮助王强设计一个满足要求的购物单。

输入描述:

输入的第 1 行，为两个正整数，用一个空格隔开：N m

（其中 N （ <32000 ）表示总钱数， m （ <60 ）为希望购买物品的个数。）

从第 2 行到第 m+1 行，第 j 行给出了编号为 j-1 的物品的基本数据，每行有 3 个非负整数 v p q

（其中 v 表示该物品的价格（ v<10000 ）， p 表示该物品的重要度（ 1 ~ 5 ）， q 表示该物品是主件还是附件。如果 q=0 ，表示该物品为主件，如果 q>0 ，表示该物品为附件， q 是所属主件的编号）

输出描述:

输出文件只有一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值（ <200000 ）。

示例1

输入

```
1000 5
800 2 0
400 5 1
300 5 1
400 3 0
500 2 0
```

输出

```
2200
```

解决代码:

```
import java.util.Scanner;

//加了限制条件的背包问题
public class Main {
    public static int getMaxValue(int[] val, int[] weight, int[] q, int n, int w) {
        int[][] dp = new int[n + 1][w + 1];
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= w; j++) {
                if (q[i-1] == 0) {  // 主件
                    if (weight[i - 1] <= j) // 用j这么多钱去买 i 件商品 可以获得最大价值
                        dp[i][j] = Math.max(dp[i - 1][j], dp[i - 1][j- weight[i - 1]]+ val[i
- 1]);
                }
                else { //附件
                    if (weight[i - 1] + weight[q[i - 1]] <= j) //附件的话 加上主件一起算
                        dp[i][j] = Math.max(dp[i - 1][j], dp[i - 1][j- weight[i - 1]]+ val[i
- 1]);
                }
            }
        }
        return dp[n][w];
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        while (input.hasNextInt()) {
            int n = input.nextInt(); // 总钱数
            int m = input.nextInt(); // 商品个数
            int[] p = new int[m];
            int[] v = new int[m];
            int[] q = new int[m];
            for (int i = 0; i < m; i++) {
                p[i] = input.nextInt();        // 价格
                v[i] = input.nextInt() * p[i]; // 价值
                q[i] = input.nextInt();        // 主or附件
            }
            System.out.println(getMaxValue(v, p, q, m, n));
        }
    }
}
```

总结：

- 背包问题要好好看一下
- 可以在本子上写下变量的含义

牛客网-华为机试练习题 17

题目描述

开发一个坐标计算工具， A表示向左移动，D表示向右移动，W表示向上移动，S表示向下移动。从（0,0）点开始移动，从输入字符串里面读取一些坐标，并将最终输入结果输出到输出文件里面。

输入：

合法坐标为A(或者D或者W或者S) + 数字（两位以内）坐标之间以;分隔。

非法坐标点需要进行丢弃。如AA10; A1A; ; YAD; 等。

下面是一个简单的例子 如：

A10;S20;W10;D30;X;A1A;B10A11;;A10;

处理过程：

起点（0,0）

\+ A10 = （-10,0）

\+ S20 = (-10,-20)

\+ W10 = (-10,-10)

\+ D30 = (20,-10)

\+ X = 无效

\+ A1A = 无效

\+ B10A11 = 无效

\+ 一个空 不影响

\+ A10 = (10,-10)

结果 （10， -10）

输入描述:

一行字符串

输出描述:

最终坐标，以,分隔

示例1

输入

A10;S20;W10;D30;X;A1A;B10A11;;A10;

输出

10,-10

解决代码：

```
import java.util.Scanner;
import java.lang.Integer;

public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        while(sc.hasNextLine())
        {
            String str = sc.nextLine();
```

```
            String[] arr = str.split(";");
            int x=0;
            int y=0;
            for(int i=0;i<arr.length;i++)
            {
                //判断是否是合法字符
                try{
                    int b = Integer.parseInt(arr[i].substring(1));
                    char dir = arr[i].charAt(0);
                    if(dir=='A'){
                        x-=b;
                    }
                    if(dir=='W'){
                        y+=b;
                    }
                    if(dir=='S'){
                        y-=b;
                    }
                    if(dir=='D'){
                        x+=b;
                    }
                }catch(Exception e){
                    continue;
                }
            }
            System.out.println(x+","+y);
    }
    }
}
```

```
import java.util.*;
public class Main{
    public static void main(String[]args){
        Scanner scan=new Scanner(System.in);
        while(scan.hasNextLine()){
            int x=0;
            int y=0;
            String[]tokens=scan.nextLine().split(";");
            for(int i=0;i<tokens.length;i++){
                try{
                    int b=Integer.parseInt(tokens[i].substring(1,tokens[i].length()));
                    if(tokens[i].charAt(0)=='A'){
                        x-=b;
                    }
                     if(tokens[i].charAt(0)=='W'){
                        y+=b;
                    }
                     if(tokens[i].charAt(0)=='S'){
                        y-=b;
                    }
                     if(tokens[i].charAt(0)=='D'){
                        x+=b;
                    }


                }catch(Exception e){
                    continue;
                }
            }
            System.out.println(x+","+y);
```

```
        }
    }
}
```

牛客网-华为机试练习题 18

题目描述

请解析IP地址和对应的掩码，进行分类识别。要求按照A/B/C/D/E类地址归类，不合法的地址和掩码单独归类。

所有的IP地址划分为 A,B,C,D,E五类

A类地址1.0.0.0~126.255.255.255;

B类地址128.0.0.0~191.255.255.255;

C类地址192.0.0.0~223.255.255.255;

D类地址224.0.0.0~239.255.255.255；

E类地址240.0.0.0~255.255.255.255

私网IP范围是：

10.0.0.0～10.255.255.255

172.16.0.0～172.31.255.255

192.168.0.0～192.168.255.255

子网掩码为二进制下前面是连续的1，然后全是0。（例如：255.255.255.32就是一个非法的掩码）

输入描述:

多行字符串。每行一个IP地址和掩码，用~隔开。

输出描述:

统计A、B、C、D、E、错误IP地址或错误掩码、私有IP的个数，之间以空格隔开。

示例1

输入

```
10.70.44.68~255.254.255.0
1.0.0.1~255.0.0.0
192.168.0.2~255.255.255.0
19..0.~255.255.255.0
```

输出

```
1 0 1 0 0 2 1
```

解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] args)throws IOException{
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        int a=0,b=0,c=0,d=0,e=0;
        int err=0;
        int pri=0;
        String str;
        String[] ip_mask;
        String[] ip;
        String[] mask;
        int i;
        while((str=br.readLine())!=null){
            ip_mask=str.split("~");
            ip=ip_mask[0].split("\\.");
            mask=ip_mask[1].split("\\.");
            //count error mask
                if(checkMask(ip_mask[1])){//mask correct
                    //count ip
                    if(checkIp(ip)){
                        i=Integer.parseInt(ip[0]);
                        if(i>=1&&i<=126){//a
                            a++;
                            if(i==10){
                                pri++;
                            }
                        }else if(i>=128&&i<=191){//b
                            b++;
                            if(i==172&&Integer.parseInt(ip[1])>=16&&Integer.parseInt(ip[1])
<=31){
                                pri++;
                            }
                        }else if(i>=192&&i<=223){//c
                            c++;
                            if(i==192&&Integer.parseInt(ip[1])==168){
                                pri++;
                            }
                        }else if(i>=224&&i<=239){//d
                            d++;
                        }else if(i>=240&&i<=255){//e
                            e++;
                        }
                    }else{
                        err++;
                    }
                }else{
                    err++;
                }
        }
        //output
        System.out.println(a+" "+b+" "+c+" "+d+" "+e+" "+err+" "+pri);
    }
public static boolean checkMask(String mask)
    {

        //check mask
        String[] mask_arr=mask.split("\\.");
```

```
        if(mask_arr[0].equals("255"))
        {
            if(mask_arr[1].equals("255"))
            {
                if(mask_arr[2].equals("255"))
                {
                    if(

 mask_arr[3].equals("254")||mask_arr[3].equals("252")||mask_arr[3].equals("248")||

 mask_arr[3].equals("240")||mask_arr[3].equals("224")||mask_arr[3].equals("192")||
                            mask_arr[3].equals("128")||mask_arr[3].equals("0"))
                        return true;
                    else
                        return false;
                }
                else
if(mask_arr[2].equals("254")||mask_arr[2].equals("252")||mask_arr[2].equals("248")||

 mask_arr[2].equals("240")||mask_arr[2].equals("224")||mask_arr[2].equals("192")||
                        mask_arr[2].equals("128")||mask_arr[2].equals("0"))
                {
                    if(mask_arr[3].equals("0"))
                        return true;
                    else
                        return false;
                }
                else
                    return false;

            }
            else
if(mask_arr[1].equals("254")||mask_arr[1].equals("252")||mask_arr[1].equals("248")||

 mask_arr[1].equals("240")||mask_arr[1].equals("224")||mask_arr[1].equals("192")||
                    mask_arr[1].equals("128")||mask_arr[1].equals("0"))
            {
                if(mask_arr[2].equals("0")&&mask_arr[3].equals("0"))
                    return true;
                else
                    return false;
            }
            else
            {
                return false;
            }

        }
        else
if(mask_arr[0].equals("254")||mask_arr[0].equals("252")||mask_arr[0].equals("248")||

 mask_arr[0].equals("240")||mask_arr[0].equals("224")||mask_arr[0].equals("192")||
                mask_arr[0].equals("128")||mask_arr[0].equals("0"))
        {
            if(mask_arr[1].equals("0")&&mask_arr[2].equals("0")&&mask_arr[3].equals("0"))
                return true;
            else
                return false;
        }
        else
```

```
        {
            return false;
        }

    }
    static boolean checkIp(String[] ip){

 if(ip.length==4&&!ip[0].equals("")&&!ip[1].equals("")&&!ip[2].equals("")&&!ip[3].equals(""))
{
            return true;
        }
        return false;
    }
}
```

总结：

- BufferedReader br=new BufferedReader(new InputStreamReader(System.in));是读取数据的另外一种形式；

牛客网-华为机试练习题 19

题目描述

开发一个简单错误记录功能小模块，能够记录出错的代码所在的文件名称和行号。

处理：

1、 记录最多8条错误记录，循环记录，对相同的错误记录（净文件名称和行号完全匹配）只记录一条，错误计数增加；

2、 超过16个字符的文件名称，只记录文件的最后有效16个字符；

3、 输入的文件可能带路径，记录文件名称不能带路径。

输入描述:

一行或多行字符串。每行包括带路径文件名称，行号，以空格隔开。

输出描述:

将所有的记录统计并将结果输出，格式：文件名 代码行数 数目，一个空格隔开，如：

示例1

输入

```
E:\V1R2\product\fpgadrive.c  1325
```

输出

```
fpgadrive.c 1325 1
```

解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.LinkedHashMap;
```

```
import java.util.Map;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        String messageLine = null;
        //使用有序的LinkedHashMap存储信息
        Map<String,Integer> map = new LinkedHashMap<String,Integer>();
        //循环读入数据
        while ((messageLine = bufferedReader.readLine()) != null){
            String[] error_Message = messageLine.split(" ");
            String error = error_Message[0];
            String line_no = error_Message[1];
            //取文件名
            String file_Name = error.substring(error.lastIndexOf("\\") + 1);
            //处理长度超过16的情况
            if(file_Name.length() > 16){
                file_Name = file_Name.substring(file_Name.length() - 16);
            }

            //将错误信息添加到map中
            String error_Name = file_Name+" "+line_no;
            if (map.containsKey(error_Name)){
                map.put(error_Name,map.get(error_Name)+1);
            } else {
                map.put(error_Name,1);
            }
        }

        //输出错误信息,最多8条(后八条)
        int count = 0;
        for (String key:map.keySet()){
            count++;
            if(count > (map.size() -8))
            System.out.println(key + " " + map.get(key));
        }
    }

}
```

总结：

- 字典：
    - 引用：java.util.Map
    - 创建：map<String,Integer> map = new Map<String,Integer>();
    - 添加元素:map.put()
    - 是否包含key: map.containsKey(key)
    - 获得元素：map.get(key)

牛客网-华为机试练习题 20

题目描述

密码要求:

1.长度超过8位

2.包括大小写字母.数字.其它符号,以上四种至少三种

3.不能有相同长度超2的子串重复

说明:长度超过2的子串

输入描述：

一组或多组长度超过2的子符串。每组占一行

输出描述：

如果符合要求输出：OK，否则输出NG

示例1

输入

021Abc9000 021Abc9Abc1 021ABC9000 021$bc9000

输出

OK NG NG OK

解决代码：

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
        while(sc.hasNextLine())
        {
            String str = sc.nextLine();
            boolean flag = Solution(str);
            if(flag){
                System.out.println("OK");
            }else{
                System.out.println("NG");
            }
        }
    }

    public static boolean Solution(String str){
        if(str.length()<=8)
        {
            return false;
        }
        int num = 0;
        int chars = 0;
        int other =0;
        for(int i=0;i<str.length();i++){
            if(str.charAt(i)>='0' &&str.charAt(i)<='9'){
                num = num+1;
            }
            if(str.charAt(i)>='a' &&str.charAt(i)<='z'){
```

```
                chars+=1;
            }
            if(str.charAt(i)>='A' &&str.charAt(i)<='Z'){
                chars+=1;
            }else{
                other+=1;
            }
        }
        if(num==0||chars==0||other==0 ){
            return false;
        }else {
            for(int i=0;i<str.length()-3;i++){
                String str1 = str.substring(i,i+3);
                String str2 = str.substring(i+3);
                if(str2.contains(str1)){
                    return false;
                }
            }
            return true;
    }
    }
}
```

总结

- 判断相同长度超过2的字串重复

```
for(int i=0;i<str.length()-3;i++){
                String str1 = str.substring(i,i+3);
                String str2 = str.substring(i+3);
                if(str2.contains(str1)){
                    return false;
                }
            }
```

牛客网-华为机试练习题 21

题目描述:简单密码

密码是我们生活中非常重要的东东，我们的那么一点不能说的秘密就全靠它了。哇哈哈. 接下来渊子要在密码之上再加一套密码，虽然简单但也安全。

假设渊子原来一个BBS上的密码为zvbo9441987,为了方便记忆，他通过一种算法把这个密码变换成YUANzhi1987，这个密码是他的名字和出生年份，怎么忘都忘不了，而且可以明目张胆地放在显眼的地方而不被别人知道真正的密码。

他是这么变换的，大家都知道手机上的字母： 1--1， abc--2, def--3, ghi--4, jkl--5, mno--6, pqrs--7, tuv--8 wxyz--9, 0--0,就这么简单，渊子把密码中出现的小写字母都变成对应的数字，数字和其他的符号都不做变换，

声明：密码中没有空格，而密码中出现的大写字母则变成小写之后往后移一位，如：X，先变成小写，再往后移一位，不就是y了嘛，简单吧。记住，z往后移是a哦。

输入描述:

输入包括多个测试数据。输入是一个明文，密码长度不超过100个字符，输入直到文件结尾

输出描述:

输出渊子真正的密文

示例1

输入

YUANzhi1987

输出

zvbo9441987

思路

做题自然要审题，把题目的要点给列出来。

- 大写字符变成小写，然后往后移一位
  - 大写换小写，ascii码加上32，后移一位，再加1
  - 如果是Z，换成a
- 小写字符按照对应规则换成数字
- 数字不变

解决代码：

```
import java.util.Scanner;
import java.lang.Integer;

public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();

        char[] res = new char[str.length()];
        for(int i=0;i<str.length();i++){
            char temp = str.charAt(i);
            char temp1;
            if(temp>='A' && temp<'Z'){
                temp1 = (char)(temp+33);
            }else if(temp=='Z'){
                temp1 = 'a';
            }else if(temp>='a' && temp<='z'){
                temp1 = Solution(temp);
            }else{
                temp1 = temp;
            }
            res[i] = temp1;
        }
        for (int i=0;i<res.length;i++){
            System.out.print(res[i]);
        }
    }
    public static char Solution(char temp){
        if(temp>='a' && temp<='c'){
            return '2';
        }else if(temp>='d' && temp<='f'){
            return '3';
```

```
        }else if(temp>='g' && temp<='i'){
            return '4';
        }else if(temp>='j' && temp<='l'){
            return '5';
        }else if(temp>='m' && temp<='o'){
            return '6';
         }else if(temp>='p' && temp<='s'){
            return '7';
         }else if(temp>='t' && temp<='v'){
            return '8';
          }else {
            return '9';
        }
    }
}
```

总结:

- ascii

符号	ascii
A	65
a	97
0	48

牛客网-华为机试练习题 22

题目描述

有这样一道智力题：“某商店规定：三个空汽水瓶可以换一瓶汽水。小张手上有十个空汽水瓶，她最多可以换多少瓶汽水喝？”答案是5瓶，方法如下：先用9个空瓶子换3瓶汽水，喝掉3瓶满的，喝完以后4个空瓶子，用3个再换一瓶，喝掉这瓶满的，这时候剩2个空瓶子。然后你让老板先借给你一瓶汽水，喝掉这瓶满的，喝完以后用3个空瓶子换一瓶满的还给老板。如果小张手上有n个空汽水瓶，最多可以换多少瓶汽水喝？

输入描述:

输入文件最多包含10组测试数据，每个数据占一行，仅包含一个正整数n（1<=n<=100），表示小张手上的空汽水瓶数。n=0表示输入结束，你的程序不应当处理这一行。

输出描述:

对于每组测试数据，输出一行，表示最多可以喝的汽水瓶数。如果一瓶也喝不到，输出0。

示例1
输入

```
3
10
81
0
```

输出

```
1
```

```
5
40
```

思路：

这种问题最好就是多列几个寻找规律。

- n=1,只有1个空瓶子，一瓶也喝不到，
- n=2,有2个空瓶子，借一瓶，还一个空瓶，可以喝1瓶
- n=3,有3个空瓶子，换一瓶，手里余一个空瓶子，喝一瓶
- n=4,有4个空瓶子，换一瓶，手里余2个空瓶子，借一瓶，喝两瓶
- n=5,有5个空瓶子，借一瓶，换2瓶，手里余1个瓶子，喝两瓶
- n=6,有6个空瓶子，换2瓶，2个空瓶子，借一瓶，手里没有空瓶子，喝3瓶，

所以，最终是n/2

解决代码：

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        while(sc.hasNext()){
            int data = sc.nextInt();
            if(data==0){
                break;
            }
            int res = data/2;
            System.out.println(res);
        }
    }
}
```

总结：

- 要注意sc.hasNext和sc.hasNextLine的区别

牛客网-华为机试练习题 23

题目描述

实现删除字符串中出现次数最少的字符，若多个字符出现次数一样，则都删除。输出删除这些单词后的字符串，字符串中其它字符保持原来的顺序。

输入描述:

字符串只包含小写英文字母， 不考虑非法输入，输入的字符串长度小于等于20个字节。

输出描述:

删除字符串中出现次数最少的字符后的字符串。

示例1

输入

abcdd

输出

dd

解决代码：

```
import java.util.*;
public class Main{
    public static void main(String[]args){
        Scanner scan=new Scanner(System.in);
        while(scan.hasNextLine()){
            String str=scan.nextLine();
            if(str.length()>20){
                continue;
            }
            int []max=new int[26];
            char[]ch=str.toCharArray();
            int min=Integer.MAX_VALUE;
            for(int i=0;i<ch.length;i++){
                max[ch[i]-'a']++;
                min=min>max[ch[i]-'a']?max[ch[i]-'a']:min;
            }


            for(int i=0;i<max.length;i++){
                  if(max[i]==min){

                     str=str.replaceAll(String.valueOf((char)(i+97)), "");
                }
            }
                System.out.println(str);
            }
    }
}
```

```
import java.util.Scanner;

public class Main{
    public static void main(String [] args){
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        int[] res = new int [26];
        int min = 22;
        for(int i=0;i<str.length();i++){
            res[str.charAt(i)-'a']++;
            //min = min>res[str.charAt(i)-97]?res[str.charAt(i)-97]:min;
        }
        for(int i=0;i<res.length;i++){
            if(res[i]>0){
                min = min>res[i]?res[i]:min;
            }
        }
```

```
        for(int i=0;i<str.length();i++){
            if(res[str.charAt(i)-'a']>min){
                System.out.print(str.charAt(i));
            }
        }
    }
}
```

牛客网-华为机试练习题 24

题目描述

计算最少出列多少位同学，使得剩下的同学排成合唱队形

说明：

N位同学站成一排，音乐老师要请其中的(N-K)位同学出列，使得剩下的K位同学排成合唱队形。 合唱队形是指这样的一种队形：设K位同学从左到右依次编号为1，2…，K，他们的身高分别为T1，T2，…，TK， 则他们的身高满足存在i（1<=i<=K）使得T1<T2<……<Ti-1Ti+1>……>TK。 你的任务是，已知所有N位同学的身高，计算最少需要几位同学出列，可以使得剩下的同学排成合唱队形。

输入描述:

整数N

输出描述:

最少需要几位同学出列

示例1
输入

```
8
186 186 150 200 160 130 197 200
```

输出

```
4
```

解决代码：

```
import java.util.Arrays;
import java.util.Scanner;


public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Scanner in = new Scanner(System.in);
        String tempString = null;
        String[] tempss = null;
        int[] nums = null;
        int n = -1,result = -1;

        while(in.hasNext()){
```

```
        tempString = in.nextLine().trim();
        n = Integer.parseInt(tempString.trim());

        tempString = in.nextLine().trim();
        tempss = tempString.split(" ");
        nums = new int[tempss.length];
        for(int i = 0 ; i< tempss.length;i++)
            nums[i] = Integer.parseInt(tempss[i].trim());
        result = process(nums);
        System.out.println(n - result);
    }

}

private static int[] largest(int[] nums) {
    // TODO Auto-generated method stub
    int[] temp = new int[nums.length];
    int lastLoc = -1,begin = -1,end = -1,curLoc = -1;

    int[] preLen = new int[nums.length];
    Arrays.fill(preLen, 0);
    preLen[0] = 1;
    Arrays.fill(temp, -1);
    temp[0] = nums[0];
    lastLoc = 0;
    for(int i = 1;i<nums.length;i++){

        if(nums[i] > temp[lastLoc]){
            lastLoc ++;
            temp[lastLoc] = nums[i];
            preLen[i] = lastLoc+1;
            continue;
        }

        begin = 0;end = lastLoc;
        while(begin <= end){
            curLoc = (begin + end)/2;
            if(temp[curLoc] < nums[i]){
                begin = curLoc + 1 ;
            }else if(temp[curLoc] > nums[i]){
                end = curLoc - 1;
            }else{
                break;
            }
        }
        preLen[i] = begin+1;
        if(temp[begin] >= nums[i])
            temp[begin] = nums[i];

    }

    return preLen;
}

private static int process(int[] nums){
    int[] preLen = null,postLen = null;
    preLen = largest(nums);

    int[] tempNums = new int[nums.length];
    int i = nums.length-1;
```

```
        for(int n:nums)
            tempNums[i--] = n;
//      System.out.println(Arrays.toString(nums));
//      System.out.println(Arrays.toString(tempNums));
        postLen = largest(tempNums);
        int k = 0;
        for(i = 0;i < preLen.length;i++){
            k = Math.max(preLen[i]+postLen[nums.length-1-i], k);
        }

//      System.out.println(Arrays.toString(preLen));
//      System.out.println(Arrays.toString(postLen));

        return k-1;

    }

}
```

牛客网-华为机试练习题 25

题目描述

编写一个程序，将输入字符串中的字符按如下规则排序。
规则 1 ：英文字母从 A 到 Z 排列，不区分大小写。
如，输入： Type 输出： epTy
规则 2 ：同一个英文字母的大小写同时存在时，按照输入顺序排列。
如，输入： BabA 输出： aABb
规则 3 ：非英文字母的其它字符保持原来的位置。
如，输入： By?e 输出： Be?y
样例：
输入：
A Famous Saying: Much Ado About Nothing(2012/8).
输出：
A aaAAbc dFgghh : iimM nNn oooos Sttuuuy (2012/8).
示例1
输入
A Famous Saying: Much Ado About Nothing (2012/8).
输出
A aaAAbc dFgghh: iimM nNn oooos Sttuuuy (2012/8).

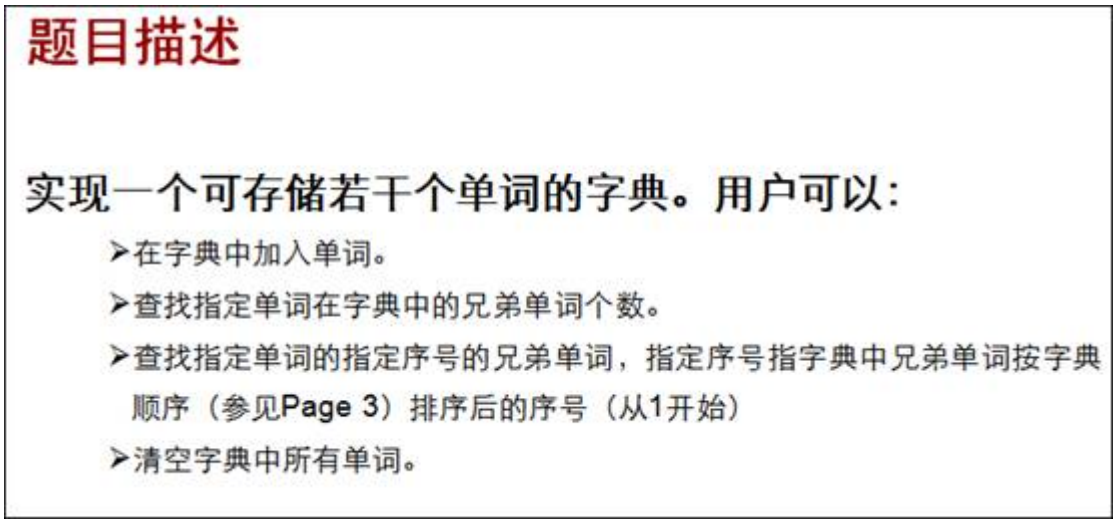
解决代码：

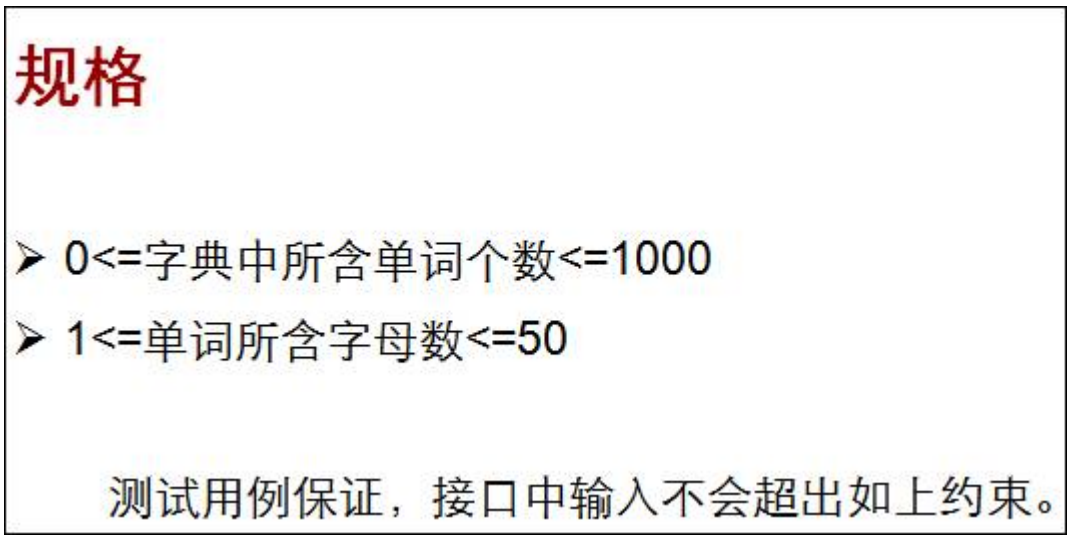
```
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Main{
    public static void main(String []args) throws IOException{
        BufferedReader bf=new BufferedReader(new InputStreamReader(System.in));
        String str;
        while((str=bf.readLine())!=null){
          StringBuffer builder = new StringBuffer();
          for(int i=0;i<26;i++){
           char c=(char)(i+'A');
              for(int j=0;j<str.length();j++){
                  char sc=str.charAt(j);
                  if(c==sc||c==sc-32){
```

```
                    builder.append(sc);
                }
            }
        }
         for(int i=0;i<str.length();i++){
             char c=str.charAt(i);
             if(!(c>='a'&&c<='z')&&!(c>='A'&&c<='Z')){
                 builder.insert(i,c);
             }
         }
          System.out.println(builder.toString());
       }
     bf.close();
    }
}
```

牛客网-华为机试练习题 26

题目描述





输入描述:

先输入字典中单词的个数，再输入n个单词作为字典单词。
输入一个单词，查找其在字典中兄弟单词的个数
再输入数字n

输出描述:

根据输入，输出查找到的兄弟单词的个数
示例1
输入

```
3       abc     bca     cab     abc     1
```

输出

```
2       bca
```

解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.TreeSet;


public class Main {

    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String input = null;
        while((input = br.readLine()) != null){
            String[] s = input.split(" ");
            int n = Integer.valueOf(s[0]);
            int k = Integer.valueOf(s[s.length-1]);
            String str = s[s.length-2];
            int m = 0;
            String[] ss = new String[n];
            if( n>=0&&n<=1000){
                for(int i = 1; i < s.length-1; i++){
                    if(s[i].length()>=1&&s[i].length()<=50){
                        if(isBrother(str, s[i])){
                            ss[m] = s[i];
                            m++;
                            //System.out.println(str+"===="+s[i]+"==="+m+"==="+i);
```

```
                    }


                }else{
                    System.out.println("单词长度越界");
                    System.exit(0);
                }
            }
        }else{
            System.out.println("n越界");
            System.exit(0);
        }

        System.out.println(m);
        //System.out.println("k="+k+"---m="+m);
        if(k<=m){
        sort(ss, k);

        }
    }
}

private static void sort(String[] ss, int k) {
    // TODO Auto-generated method stub
    int min;
    for(int i = 0; i < ss.length; i++){
        min = i;
        for(int j = i+1; j < ss.length; j++){
            if(ss[i]!=null && ss[j]!=null){
                if(ss[j].compareTo(ss[min])<0){
                    min = j;
                }
            }

        }
        String temp = ss[min];
        ss[min] = ss[i];
        ss[i] = temp;

    }

        System.out.println(ss[k-1]);


}

private static boolean isBrother(String str, String string) {
    // TODO Auto-generated method stub
    int[] s1 = new int[26];
    for(int i = 0; i < str.length(); i++){
        s1[(int)str.charAt(i) - 97]++;
    }
    int[] s2 = new int[26];
    for(int i = 0; i < string.length(); i++){
        s2[(int)string.charAt(i) - 97]++;
    }

    if(str.length() == string.length() && !str.equals(string)){
        for(int i = 0; i < 26; i++){
```

```
                if(s1[i] == s2[i]){

                }else{
                    return false;
                }
            }
        }else{
            return false;
        }
        return true;
    }
}
```

总结

- 兄弟单词的处理，那么就比较每个元素的个数
- 字符串比较实用str.compareTo(String)<0

牛客网-华为机试练习题 27

题目描述:素数伴侣

题目描述
若两个正整数的和为素数，则这两个正整数称之为“素数伴侣”，如2和5、6和13，它们能应用于通信加密。现在密码学会请你设计一个程序，从已有的N（N为偶数）个正整数中挑选出若干对组成“素数伴侣”，挑选方案多种多样，例如有4个正整数：2，5，6，13，如果将5和6分为一组中只能得到一组“素数伴侣”，而将2和5、6和13编组将得到两组“素数伴侣”，能组成“素数伴侣”最多的方案称为“最佳方案”，当然密码学会希望你寻找出“最佳方案”。
输入:
有一个正偶数N（N≤100），表示待挑选的自然数的个数。后面给出具体的数字，范围为[2,30000]。
输出:
输出一个整数K，表示你求得的“最佳方案”组成“素数伴侣”的对数。

输入描述:

输入说明
1 输入一个正偶数n
2 输入n个整数

输出描述:

求得的“最佳方案”组成“素数伴侣”的对数。
示例1
输入
4
2 5 6 13
输出
2

解决代码

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) throws Exception {
        // 1.高效读数据
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str = null;
```

```
        while ((str = br.readLine()) != null) {
            int n = Integer.parseInt(str);
            long[] arr = new long[n];
            String[] numStr = br.readLine().split(" ");// str->str数组
            for (int i = 0; i < arr.length; i++) {
                arr[i] = Integer.parseInt(numStr[i]);
            }

            // 2.分奇偶
            ArrayList<Long> evens = new ArrayList<Long>();
            ArrayList<Long> odds = new ArrayList<Long>();
            for (int i = 0; i < n; i++) {
                if (arr[i] % 2 == 0) {
                    evens.add(arr[i]);
                } else {
                    odds.add(arr[i]);
                }
            }

            if (n == 22) {
                System.out.println(8);
            } else if (n == 12) {
                System.out.println(4);
            } else {
                if(evens.size()<odds.size()){
                    System.out.println(evens.size());
                }
                else{
                    System.out.println(odds.size());
                }
            }

            // 3.得到从偶数集合和奇数集合各抽取一个数字组成素数的最大组合数

        }
    }

}
```

牛客网-华为机试练习题 28

题目描述

1、对输入的字符串进行加解密，并输出。

2加密方法为：

当内容是英文字母时则用该英文字母的后一个字母替换，同时字母变换大小写,如字母a时则替换为B；字母Z时则替换为a；

当内容是数字时则把该数字加1，如0替换1，1替换2，9替换0；

其他字符不做变化。

3、解密方法为加密的逆过程。

接口描述：

实现接口，每个接口实现1个基本操作：

`void Encrypt (char aucPassword[], char aucResult[])`：在该函数中实现字符串加密并输出

说明：

1、字符串以\0结尾。

2、字符串最长100个字符。

int unEncrypt (char result[], char password[])：在该函数中实现字符串解密并输出

说明：

1、字符串以\0结尾。

2、字符串最长100个字符。

输入描述:

输入说明
输入一串要加密的密码
输入一串加过密的密码

输出描述:

输出说明
输出加密后的字符
输出解密后的字符

示例1

输入

```
abcdefg
BCDEFGH
```

输出

```
BCDEFGH
abcdefg
```

解决代码

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
        String str1 = sc.nextLine();
        String str2 = sc.nextLine();
        for(int i=0;i<str1.length();i++){
            System.out.print(solution_encode(str1.charAt(i)));
        }
        System.out.print('\n');

        for(int i=0;i<str2.length();i++){
            System.out.print(solution_decode(str2.charAt(i)));
        }
    }

    public static char solution_encode(char ch){
```

```
        if(ch>='0' && ch<='8'){
            return (char)(ch+1);
        }else if(ch=='9')
            return '0';
        if(ch>='A' && ch<'Z'){
            return (char)(ch+33);
        }
        if(ch=='Z'){
            return 'a';
        }
        if(ch>='a' && ch<'z'){
            return (char)(ch-32+1);
        }
        if(ch=='z'){
            return 'A';
        }
        return ch;
    }

    public static char solution_decode(char ch){
        if(ch=='0'){
            return '9';
        }
        if(ch>'0' && ch<='9'){
            return (char)(ch-1);
        }
        if(ch=='a'){
            return 'Z';
        }
        if(ch>'a' && ch<='z'){
            return (char)(ch-32-1);
        }
        if(ch=='A'){
            return 'z';
        }
        if(ch>'A' && ch<='Z'){
            return (char)(ch+32-1);
        }
        return ch;
    }
}
```

```
import java.util.*;
import java.io.*;

public class Main{
    public static void main(String[] args) throws IOException{
        BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        String str = "";
        while((str=bf.readLine())!=null){
            String str1 = Encrypt(str);
            System.out.println(str1);
            str=bf.readLine();
            String str2=unEncrypt(str);
            System.out.println(str2);
        }
    }

    private static String Encrypt(String line){
        char[] cha = line.toCharArray();
```

```
    StringBuilder sb= new StringBuilder();
    for(int i=0;i<cha.length;i++){
        if (cha[i]>='a'&&cha[i]<='z'){
            if(cha[i]=='z'){
                sb.append('A');
            }else{
                sb.append((char)(cha[i]+1-32));
            }
        }else if(cha[i]>='A'&&cha[i]<='Z'){
            if(cha[i]=='Z'){
                sb.append('a');
            }else{
                sb.append((char)(cha[i]+32+1));
            }
        }else if(cha[i]>='0'&&cha[i]<='9'){
            if(cha[i]=='9'){
                sb.append('0');
            }else{
                sb.append(cha[i]-48+1);
               // sb.append((char)(cha[i]+1));
            }
        }else{
            sb.append(cha[i]);
        }
    }
    return sb.toString();
}


private static String unEncrypt(String line){
    char[] cha = line.toCharArray();
    StringBuilder sb= new StringBuilder();
    for(int i=0;i<cha.length;i++){
        if (cha[i]>='a'&&cha[i]<='z'){
            if(cha[i]=='a'){
                sb.append('Z');
            }else{
                sb.append((char)(cha[i]-1-32));
            }
        }else if(cha[i]>='A'&&cha[i]<='Z'){
            if(cha[i]=='A'){
                sb.append('z');
            }else{
                sb.append((char)(cha[i]+32-1));
            }
        }else if(cha[i]>='0'&&cha[i]<='9'){
            if(cha[i]=='0'){
                sb.append('9');
            }else{
                sb.append(cha[i]-48-1);
              // System.out.println((char)(cha[i]-1));
                 //System.out.println(cha[i]-48-1);
            }
        }else{
            sb.append(cha[i]);
        }
    }
    return sb.toString();
}
```

```
}
```

牛客网-华为机试练习题 30

题目描述：字符串合并

按照指定规则对输入的字符串进行处理。

详细描述：

将输入的两个字符串合并。

对合并后的字符串进行排序，要求为：下标为奇数的字符和下标为偶数的字符分别从小到大排序。这里的下标意思是字符在字符串中的位置。

对排序后的字符串进行操作，如果字符为‘0’--‘9’或者‘A’--‘F’或者‘a’--‘f’，则对他们所代表的16进制的数进行BIT倒序的操作，并转换为相应的大写字符。如字符为‘4’，为0100b，则翻转后为0010b，也就是2。转换后的字符为‘2’； 如字符为‘7’，为0111b，则翻转后为1110b，也就是e。转换后的字符为大写‘E’。

举例：输入str1为"dec"，str2为"fab"，合并为“decfab”，分别对“dca”和“efb”进行排序，排序后为“abcedf”，转换后为“5D37BF”

接口设计及说明：

```
/*

功能:字符串处理

输入:两个字符串,需要异常处理

输出:合并处理后的字符串，具体要求参考文档

返回:无

*/

void ProcessString(char* str1,char *str2,char * strOutput)

{

}
```

输入描述:

输入两个字符串

输出描述:

输出转化后的结果

示例1

输入

dec fab

输出

5D37BF

解决代码

```
import java.util.Arrays;
import java.util.Scanner;
/**
 * Created by Administrator on 2015/12/22.
 */
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
         while (sc.hasNext()){
             StringBuffer rs=new StringBuffer();
             char[] strmx=(sc.next()+sc.next()).toCharArray();
             String string1="";
             String string2="";
             for(int i=0;i<strmx.length;i++){
                 if(i%2==0){
                     string1+=strmx[i];
                 }else {
                     string2+=strmx[i];
                 }
             }
             char[] str1=string1.toCharArray();
             char[] str2=string2.toCharArray();
             Arrays.sort(str1);
             Arrays.sort(str2);
             String strx="";
              int k=0;
             for(int i=0;i<Math.min(str1.length,str2.length);i++){
                 strx+=str1[i];
                 strx+=str2[i];
                 if(i==Math.min(str1.length,str2.length)-1){
                     k=i;
                 }
             }
             if(str1.length>str2.length){
                 strx+=str1[k+1];
             }else if(str1.length<str2.length) {
                 strx+=str2[k+1];
             }
             char[] str=strx.toCharArray();
             for (int i=0;i<str.length;i++){
                 if(String.valueOf(str[i]).matches("[A-Fa-f]")){
                     String
res=revser(Integer.toBinaryString(Integer.valueOf(Character.toLowerCase(str[i])) - 87));
                     int x=Integer.parseInt(res,2);
```

```
                    rs.append(Nx(x));
                }else if(String.valueOf(str[i]).matches("[0-9]")){
                    String res="";
                    String
hex=Integer.toBinaryString(Integer.parseInt(String.valueOf(str[i])));
                    if(hex.length()<4){
                        for(int j=0;j<4-hex.length();j++){
                            res+="0";
                        }
                    }
                    String resx=revser(res+hex);
                    int x=Integer.parseInt(resx, 2);
                    rs.append(Nx(x));
                }else {
                    rs.append(str[i]);
                }
            }
            System.out.println(rs);

        }

    }
    public  static String revser(String srx){
        StringBuffer sb=new StringBuffer();
        return sb.append(srx).reverse().toString();
    }
    public static String Nx(int x){
        if(x==10){
            return "A";
        }else if(x==11){
            return "B";
        }else if(x==12){
            return "C";
        }else if(x==13){
            return "D";
        }else if(x==14){
            return "E";
        }else if(x==15){
            return "F";
        }
      return String.valueOf(x);
    }
}
```

牛客网-华为机试练习题 31

题目描述：单词倒排

对字符串中的所有单词进行倒排。

说明：

1、每个单词是以26个大写或小写英文字母构成；

2、非构成单词的字符均视为单词间隔符；

3、要求倒排后的单词间隔符以一个空格表示；如果原字符串中相邻单词间有多个间隔符时，倒排转换后也只允许出现一个空格间隔符；

4、每个单词最长20个字母；

输入描述:

输入一行以空格来分隔的句子

输出描述:

输出句子的逆序

示例1

输入

复制

```
I am a student
```

输出

复制

```
student a am I
```

解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line;
        while ((line=br.readLine())!=null){
            StringBuffer sb=new StringBuffer();
            for (int i=0;i<line.length();i++){
                char c=line.charAt(i);
                if ((c>='A'&&c<='Z')||(c>='a'&&c<='z')){
                    sb.append(c);
                }else {
                    sb.append(' ');
                }
            }
            String[] str=sb.toString().trim().split(" ");
            StringBuffer s2=new StringBuffer();
            for (int j=str.length-1;j>=0;j--){
                if (!(str[j].equals(" "))){
                    s2.append(str[j]);
                }
                if (j>0){
                    s2.append(" ");
                }
            }
            System.out.println(s2.toString());
        }
    }
}
```

牛客网-华为机试练习题 32

题目描述:字符串运用-密码截取

Catcher是MCA国的情报员，他工作时发现敌国会用一些对称的密码进行通信，比如像这些ABBA，ABA，A，123321，但是他们有时会在开始或结束时加入一些无关的字符以防止别国破解。比如进行下列变化 ABBA->12ABBA,ABA->ABAKK,123321->51233214 。因为截获的串太长了，而且存在多种可能的情况（abaaab可看作是aba,或baaab的加密形式），Cathcer的工作量实在是太大了，他只能向电脑高手求助，你能帮Catcher找出最长的有效密码串吗？

输入描述:

输入一个字符串

输出描述:

返回有效密码串的最大长度

示例1
输入
ABBA
输出
4

思路

- 在每个字符的左右加上一个#，然后统计每个字符的最长回文半径

  ABBA
  #A#B#B#A#
  010141010

- 最大的回文半径就是最终的结果

解决代码：

```
//同楼上一位答主一样，以每个字符（奇数长度的回文串）或者字符间空隙
//（偶数长度的回文串）分别向左向右扩充，记录遇到的最大长度
import java.util.Scanner;


public class Main{
    public static int process(String str){
        int len=str.length();
        if(len<1){
            return 0;
        }
        int max=1;//只要字符创长度大于1，则最短的回文串长度为1
        //考虑奇数个数的回文串
        for(int i=1;i<len-1;i++){
            int k=i-1,j=i+1;
            int count=0;
            while(k>=0 && j<=len-1){
                if(str.charAt(k--)==str.charAt(j++)){
                    count++;
                }else{
                    break;
                }
            }
            max= (max>=(count*2+1)) ? max:(count*2+1);
        }
        //现在考虑偶数回文串的情况，主要考虑字符之间的位置
```

```
        for(int i=1;i<len-1;i++){
            int k=i-1,j=i;
            int count=0;
            while(k>=0 && j<=len-1){
                if(str.charAt(k--)==str.charAt(j++)){
                    count++;
                }else{
                    break;
                }
            }
            max= (max>=count*2) ? max : (count*2);
        }
        return max;
    }
    public static void main(String[] args){
        Scanner in=new Scanner(System.in);
        while(in.hasNext()){
            String str=in.next();
            System.out.println(process(str));
        }
        in.close();
    }
}
```

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        while(sc.hasNextLine()){
            String str = sc.nextLine();
            int num = Solution(str);
            System.out.println(num);
        }

    }

    public static int Solution(String str){
        StringBuilder sbuf = new StringBuilder();
        sbuf.append('#');
        for(int i=0;i<str.length();i++){
            sbuf.append(str.charAt(i));
            sbuf.append('#');
        }
        String new_str = sbuf.toString();
        char[] arr_str = new_str.toCharArray();
        int[] radius = new int[arr_str.length];
        radius[0] = 0;
        radius[radius.length-1]=0;
        for(int i=1;i<arr_str.length-1;i++){
            int count = 1;
            while(i>=count && (i+count)<arr_str.length) {
                if (arr_str[i - count] == arr_str[i + count]) {
                    count++;
                } else {
                    break;
                }
            }
            radius[i]=count-1;
```

```
        }
        int res =0;
        for(int i=0;i<radius.length;i++){
            if(radius[i]>res)
                res = radius[i];
        }
        return res;
    }
}
```

牛客网-华为机试练习题 33

题目描述：整数与IP地址间的转换

原理：ip地址的每段可以看成是一个0-255的整数，把每段拆分成一个二进制形式组合起来，然后把这个二进制数转变成一个长整数。
举例：一个ip地址为10.0.3.193

每段数字	相对应的二进制数
10	00001010
0	00000000
3	00000011
193	11000001

组合起来即为：00001010 00000000 00000011 11000001,转换为10进制数就是：167773121，即该IP地址转换后的数字就是它了。

的每段可以看成是一个0-255的整数，需要对IP地址进行校验

输入描述:

输入
1 输入IP地址
2 输入10进制型的IP地址

输出描述:

输出
1 输出转换成10进制的IP地址
2 输出转换后的IP地址

示例1

输入
10.0.3.193
167969729

输出
167773121
10.3.3.193

解决代码：

```
import java.util.*;
import java.io.*;

public class Main {
    public static void Change1(String str) {
        String[] data1 = str.split("\\.");

        data1[0] = Integer.toBinaryString(Integer.parseInt(data1[0]));
        data1[1] = Integer.toBinaryString(Integer.parseInt(data1[1]));
```

```
        data1[2] = Integer.toBinaryString(Integer.parseInt(data1[2]));
        data1[3] = Integer.toBinaryString(Integer.parseInt(data1[3]));

        while(data1[0].length()<8) data1[0] = "0"+data1[0];
        while(data1[1].length()<8) data1[1] = "0"+data1[1];
        while(data1[2].length()<8) data1[2] = "0"+data1[2];
        while(data1[3].length()<8) data1[3] = "0"+data1[3];


        long sum = 0;
        for(int i=0;i<data1.length;i++) {
            for(int j=0;j<data1[0].length();j++) {
                sum = sum*2+(data1[i].charAt(j)-'0');
            }
        }
        System.out.println(sum);

    }
    public static void Change2(String str) {
        long data2 = Long.parseLong(str);
        String bindata2 = Long.toBinaryString(data2);
        String[] data = new String[4];
        data[0] = bindata2.substring(0,bindata2.length()-3*8);
        data[1] = bindata2.substring(data[0].length(),data[0].length()+8);
        data[2] =
bindata2.substring(data[0].length()+data[1].length(),data[0].length()+data[1].length()+8);
        data[3] = bindata2.substring(bindata2.length()-8,bindata2.length());

        System.out.print(Integer.valueOf(data[0],2)+".");
        System.out.print(Integer.valueOf(data[1],2)+".");
        System.out.print(Integer.valueOf(data[2],2)+".");
        System.out.println(Integer.valueOf(data[3],2));
    }
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        String str;
        while((str=br.readLine())!=null) {
            Change1(str);
            str=br.readLine();
            Change2(str);
        }
    }
}
```

总结

- public static void main(String[] args) throws IOException {BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

需要加throws IOException

- Long.toBinaryString可以将数字处理成二进制字符串
- Integer.valueOf可以获得字符串对应的数字。

牛客网-华为机试练习题 34

题目描述:图片整理

Lily上课时使用字母数字图片教小朋友们学习英语单词，每次都需要把这些图片按照大小（ASCII码值从小到大）排列收好。请大家给Lily帮忙，通过C语言解决。

输入描述:

Lily使用的图片包括"A"到"Z"、"a"到"z"、"0"到"9"。输入字母或数字个数不超过1024。

输出描述:

Lily的所有图片按照从小到大的顺序输出

示例1

输入

```
Ihave1nose2hands10fingers
```

输出

```
0112Iaadeeefghhinnnorsssv
```

解决代码：

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        while(sc.hasNextLine()){
            String str = sc.nextLine();
            char[] arr = str.toCharArray();
            char[] res = sort(arr);
            System.out.println(res);
        }
    }

    public static char[] sort(char[] arr){
        for(int i=0;i<arr.length;i++){
            for(int j=0;j<arr.length-1-i;j++){
                if(arr[j+1]<arr[j]){
                    char temp = arr[j+1];
                    arr[j+1]=arr[j];
                    arr[j] = temp;
                }
            }
        }
        return arr;
    }
}
```

总结：

- 要注意处理多个case的 输入
- 冒泡排序边界不需要加等号
- 字符串转成数组的语句是char[] arr = str.toCharArray();

牛客网-华为机试练习题 35

题目描述：蛇形矩阵

蛇形矩阵是由1开始的自然数依次排列成的一个矩阵上三角形。

样例输入

5

样例输出

1 3 6 10 15

2 5 9 14

4 8 13

7 12

11

输入描述:

输入正整数N（N不大于100）

输出描述:

输出一个N行的蛇形矩阵。

示例1

输入

4

输出

```
1 3 6 10
2 5 9
4 8
7
```

解决代码：

```
import java.util.Scanner;
public class Main {
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        while(sc.hasNext()){
            int n=sc.nextInt();
            StringBuilder builder = new StringBuilder();
            for (int i = 1; i <= n; i++) {
                for (int j = 1, start = (i - 1) * i / 2 + 1, step = i + 1; j <= n - i + 1;
j++, start += step, step++) {
                    builder.append(start).append(' ');
                }
                // 设置换行符
                builder.setCharAt(builder.length()-1, '\n');
```

```
            }
            System.out.print(builder.toString());
        }
    }

}
```

总结：

- 纵列第一个元素的值是(i-1)/2+1,
- 横列元素的初始步长是i+1，步长每运算一次自加1

牛客网-华为机试练习题 36

题目描述：字符串加密

有一种技巧可以对数据进行加密，它使用一个单词作为它的密匙。下面是它的工作原理：首先，选择一个单词作为密匙，如TRAILBLAZERS。如果单词中包含有重复的字母，只保留第1个，其余几个丢弃。现在，修改过的那个单词属于字母表的下面，如下所示：

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

T R A I L B Z E S C D F G H J K M N O P Q U V W X Y

上面其他用字母表中剩余的字母填充完整。在对信息进行加密时，信息中的每个字母被固定于顶上那行，并用下面那行的对应字母一一取代原文的字母(字母字符的大小写状态应该保留)。因此，使用这个密匙，Attack AT DAWN(黎明时攻击)就会被加密为Tpptad TP ITVH。

请实现下述接口，通过指定的密匙和明文得到密文。

输入描述:

先输入key和要加密的字符串

输出描述:

返回加密后的字符串

示例1

输入

```
nihao
ni
```
````##### 输出


le

##### 解决代码：
```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.List;
```
```

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) throws IOException {
        BufferedReader br =new BufferedReader(new InputStreamReader(System.in));
        String sr="";
        while((sr=br.readLine())!=null) {
            StringBuilder bu=new StringBuilder();
            String sr1=br.readLine();
            HashSet<Character> set=new LinkedHashSet<Character>();
            for(int i=0;i<sr.length();i++)
                set.add(sr.charAt(i));
            for(int i=0;i<26;i++)
                set.add((char) (i+'a'));
            char ch[]=new char[set.size()];
            Iterator iter=set.iterator();
            for(int i=0;i<ch.length&&iter.hasNext();i++)
                ch[i]=(Character) iter.next();
            for(int i=0;i<sr1.length();i++) {
                if(Character.isLowerCase(sr1.charAt(i)))
                    bu.append(Character.toLowerCase(ch[sr1.charAt(i)-'a']));
                else
                    bu.append(Character.toUpperCase(ch[sr1.charAt(i)-'A']));
            }
            System.out.println(bu);
        }
    }
}
```

# 牛客网-华为机试练习题 37

## 题目描述：统计每个月兔子的总数

有一只兔子，从出生后第3个月起每个月都生一只兔子，小兔子长到第三个月后每个月又生一只兔子，假如兔子都不死，问每个月的兔子总数为多少？

```
    /**
     * 统计出兔子总数。
     *
     * @param monthCount 第几个月
     * @return 兔子总数
     */
    public static int getTotalCount(int monthCount)
    {
        return 0;
    }
```

## 输入描述:

输入int型表示month

## 输出描述:

输出兔子总数int型

示例1

输入

9

输出

34

### 解决代码：

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
         while (sc.hasNextInt()) {
        int month = sc.nextInt();
        System.out.println(count(month));
         }
        sc.close();
    }
    public static int count(int month){
        if(month==1||month==2){
            return 1;
        }
        if(month == 0 ){
            return 0;
        }
        else{
        return count(month-1)+count(month-2);
        }
    }
}
```

## 牛客网-华为机试练习题 38

### 题目描述：求小球落地5次后所经历的路程和第5次反弹的高度

假设一个球从任意高度自由落下，每次落地后反跳回原高度的一半; 再落下, 求它在第5次落地时，共经历多少米?第5次反弹多高？

### 输入描述:

输入起始高度，int型

### 输出描述:

分别输出第5次落地时，共经过多少米第5次反弹多高

示例1

输入

1

输出

2.875
0.03125

### 解决代码：

```
import java.util.Scanner;
public class Main {

    public static double getJourney(int high)
    {
        double highSum=0;
        double high1=high;
        for(int i=1;i<=5;i++){
            highSum+=high1+high1/2;
            high1/=2;
        }
        highSum-=high1;
        return highSum;
    }
    public static double getTenthHigh(int high)
    {
        double highSum=0;
        double high1=high;
        for(int i=1;i<=5;i++){
            high1/=2;
        }

        return high1;
    }

    public static void main(String[]args){
        Scanner in=new Scanner(System.in);
        while(in.hasNext()){
            int high=in.nextInt();
            System.out.println(getJourney(high));
            System.out.println(getTenthHigh(high));
        }
    }
}
```

## 牛客网-华为机试练习题 39

### 题目描述：判断两个IP是否属于同一子网

子网掩码是用来判断任意两台计算机的IP地址是否属于同一子网络的根据。
子网掩码与IP地址结构相同，是32位二进制数，其中网络号部分全为“1”和主机号部分全为“0”。利用子网掩码可以判断两台主机是否中同一子网中。若两台主机的IP地址分别与它们的子网掩码相“与”后的结果相同，则说明这两台主机在同一子网中。

示例：
I P 地址  192.168.0.1
子网掩码  255.255.255.0

转化为二进制进行运算：

I P 地址  11010000.10101000.00000000.00000001
子网掩码  11111111.11111111.11111111.00000000

AND运算
        11000000.10101000.00000000.00000000

转化为十进制后为：
        192.168.0.0



I P 地址  192.168.0.254
子网掩码  255.255.255.0


转化为二进制进行运算：

I P 地址  11010000.10101000.00000000.11111110
子网掩码  11111111.11111111.11111111.00000000

AND运算
        11000000.10101000.00000000.00000000

转化为十进制后为：
        192.168.0.0

通过以上对两台计算机IP地址与子网掩码的AND运算后，我们可以看到它运算结果是一样的。均为192.168.0.0，所以这二台计算机可视为是同一子网络。

```
/*
* 功能: 判断两台计算机IP地址是同一子网络。
* 输入参数：     String Mask: 子网掩码，格式：“255.255.255.0”；
*                String ip1: 计算机1的IP地址，格式：“192.168.0.254”；
*                String ip2: 计算机2的IP地址，格式：“192.168.0.1”；
*

\* 返回值：       0：IP1与IP2属于同一子网络；      1：IP地址或子网掩码格式非法；     2：IP1与IP2不属于同一子网络
*/
public int checkNetSegment(String mask, String ip1, String ip2)
{
    /*在这里实现功能*/
    return 0;
}
```

## 输入描述:

输入子网掩码、两个ip地址

## 输出描述:

得到计算结果

示例1

输入

255.255.255.0 192.168.224.256 192.168.10.4

输出

1

### 解决代码：

```
import java.util.Arrays;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNext()) {
            String[] mask = sc.next().split("\\.");
            String[] ip1 = sc.next().split("\\.");
            String[] ip2 = sc.next().split("\\.");
            for (int i = 0; i < 4; i++) {
                if (Integer.valueOf(mask[i]) < 0 || Integer.valueOf(mask[i]) > 255 ||
Integer.valueOf(ip1[i]) < 0
                        || Integer.valueOf(ip1[i]) > 255 || Integer.valueOf(ip2[i]) < 0
                        || Integer.valueOf(ip2[i]) >
255||mask.length!=4||ip1.length!=4||ip2.length!=4) {
                    System.out.println(1);// IP地址或子网掩码格式非法
                    break;
                }
                else if ((Integer.valueOf(mask[i]) & Integer.valueOf(ip1[i])) ==
(Integer.valueOf(mask[i])
                        & Integer.valueOf(ip2[i]))) {
                    System.out.println(0);// IP1与IP2属于同一子网
                }
                else {
                    System.out.println(2);// IP1与IP2不属于同一子网络
                    break;
                }
            }

        }
    }

}
```

## 牛客网-华为机试练习题 40

### 题目描述

输入一行字符，分别统计出包含英文字母、空格、数字和其它字符的个数。

```
    /**
```

```
     * 统计出英文字母字符的个数。
     *
     * @param str 需要输入的字符串
     * @return 英文字母的个数
     */
    public static int getEnglishCharCount(String str)
    {
        return 0;
    }

    /**
     * 统计出空格字符的个数。
     *
     * @param str 需要输入的字符串
     * @return 空格的个数
     */
    public static int getBlankCharCount(String str)
    {
        return 0;
    }

    /**
     * 统计出数字字符的个数。
     *
     * @param str 需要输入的字符串
     * @return 英文字母的个数
     */
    public static int getNumberCharCount(String str)
    {
        return 0;
    }

    /**
     * 统计出其它字符的个数。
     *
     * @param str 需要输入的字符串
     * @return 英文字母的个数
     */
    public static int getOtherCharCount(String str)
    {
        return 0;
    }
```

## 输入描述:

输入一行字符串，可以有空格

## 输出描述:

统计其中英文字符，空格字符，数字字符，其他字符的个数

示例1

输入

```
1qazxsw23 edcvfr45tgbn hy67uj m,ki89ol.\\/;p0-=\\][
```

输出

```
26
3
10
12
```

## 解决代码：

```
import java.util.Scanner;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        while(sc.hasNextLine()){
            String str = sc.nextLine();
            System.out.println(getEnglishCharCount(str));
            System.out.println(getBlankCharCount(str));
            System.out.println(getNumberCharCount(str));
            System.out.println(getOtherCharCount(str));
        }
    }

    public static int getEnglishCharCount(String str){
        int num =0;
        for(int i=0;i<str.length();i++){
            char temp = str.charAt(i);
            if((temp>='A' && temp<='Z')||(temp>='a' && temp<='z')){
                num++;
            }
        }
        return num;
    }

    public static int getBlankCharCount(String str){
        int num =0;
        for(int i=0;i<str.length();i++){
            char temp = str.charAt(i);
            if(temp==' '){
                num++;
            }
        }
        return num;

    }

    public static int getNumberCharCount(String str){
        int num =0;
        for(int i=0;i<str.length();i++){
            char temp = str.charAt(i);
            if(temp>='0' && temp<='9'){
                num++;
            }
        }
        return num;

    }
    public static int getOtherCharCount(String str)
    {
        int num =0;
        for(int i=0;i<str.length();i++){
```

```
            char temp = str.charAt(i);
            if(!(temp>='A' && temp<='Z')&&!(temp>='a' && temp<='z')&&temp!=' '&&!(temp>='0'
&& temp<='9')){
                num++;
            }
        }
        return num;
    }
}
```

# 牛客网-华为机试练习题 41

## 题目描述：称砝码

现有一组砝码，重量互不相等，分别为m1,m2,m3…mn；
每种砝码对应的数量为x1,x2,x3...xn。现在要用这些砝码去称物体的重量(放在同一侧)，问能称出多少种不同的重量。
注：
称重重量包括0
方法原型：**public** **static** **int** fama(**int** n, **int**[] weight, **int**[] nums)

## 输入描述:

输入包含多组测试数据。

对于每组测试数据：

第一行：n --- 砝码数(范围[1,10])

第二行：m1 m2 m3 ... mn --- 每个砝码的重量(范围[1,2000])

第三行：x1 x2 x3 .... xn --- 每个砝码的数量(范围[1,6])

## 输出描述:

利用给定的砝码可以称出的不同的重量数

示例1

输入

```
2
1 2
2 1
```

输出

```
5
```

## 解决代码：

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNextLine()){
            int n = Integer.parseInt(sc.nextLine()); // n 种砝码
            String str1 = sc.nextLine();
```

```
            String str2 = sc.nextLine();
            System.out.println(getNums(n, str1, str2));
        }
        sc.close();
    }

    private static int getNums(int n, String str1, String str2) {
        // TODO Auto-generated method stub



        String[] strings1 = str1.split(" ");
        String[] strings2 = str2.split(" ");
        int[] m = new int[n];
        int[] x = new int[n];
        int sum = 0; // 总的重量
        for (int i = 0; i < n; i++) {
            m[i] = Integer.valueOf(strings1[i]); // 每种砝码的重量
            x[i] = Integer.valueOf(strings2[i]); // 每种砝码的数量
            sum += x[i] * m[i];
        }
        boolean[] temp = new boolean[sum+1];
        temp[0] = true;
        temp[sum] = true;
        for (int i = 0; i < n; i++) {//砝码的种类数
            for (int j = 0; j < x[i]; j++) {//每种砝码对应的个数
                for (int k = sum; k >= m[i]; k--) {//总重量往下减
                    if (temp[k - m[i]])//递归思想的应用
                        temp[k] = true;
                }
            }
        }
        int count = 0;
        for (int i = 0; i <= sum; i++) {
            if (temp[i])
                count++;
        }//找到temp[]为true的，这是可以被称出来的；
        return count;
    }
}
```

# 牛客网-华为机试练习题 42

## 题目描述

Jessi初学英语，为了快速读出一串数字，编写程序将数字转换成英文：

如22：twenty two，123：one hundred and twenty three。

说明：

数字为正整数，长度不超过九位，不考虑小数，转化结果为英文小写；

输出格式为twenty two；

非法数据请返回“error”；

关键字提示：and，billion，million，thousand，hundred。

方法原型：public static String parse(long num)

## 输入描述:

输入一个long型整数

## 输出描述:

输出相应的英文写法

示例1

输入

2356 输出

two thousand three hundred and fifty six

## 解决代码

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.HashMap;

public class Main {

    static HashMap<Integer, String> dict = new HashMap<>();
    static {
        dict.put(0, "zero");
        dict.put(1, "one");
        dict.put(2, "two");
        dict.put(3, "three");
        dict.put(4, "four");
        dict.put(5, "five");
        dict.put(6, "six");
        dict.put(7, "seven");
        dict.put(8, "eight");
        dict.put(9, "nine");
        dict.put(10, "ten");
        dict.put(11, "eleven");
        dict.put(12, "twelve");
        dict.put(13, "thirteen");
        dict.put(14, "fourteen");
        dict.put(15, "fifteen");
        dict.put(16, "sixteen");
        dict.put(17, "seventeen");
        dict.put(18, "eighteen");
        dict.put(19, "nineteen");
    }

     public static void main(String[] arsg) throws IOException {
         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         String str;

         while ((str = br.readLine()) != null) {
             if(str.length()>0 && str.length() <= 9 && !str.contains(".")) {
                 int num = Integer.valueOf(str);
```

```
                System.out.println(parse(num));
            }
        }
    }

    public static String parse(int num){//翻译数字为英文
        int len = String.valueOf(num).length();
        if(len == 1){
            return dict.get(num);
        }else if(len == 2){
            if(num<=19) return dict.get(num);
            else if(num<30){
               return num % 10 != 0 ? "twenty " + dict.get(num % 10): "twenty";
            }else if(num<40){
                return num % 10 != 0 ? "thirty " + dict.get(num % 10): "thirty";
            }else if(num<50){
                return num % 10 != 0 ? "forty " + dict.get(num % 10): "forty";
            }else if(num<60){
                return num % 10 != 0 ? "fifty " + dict.get(num % 10): "fifty";
            }else if(num<70){
                return num % 10 != 0 ? "sixty " + dict.get(num % 10): "sixty";
            }else if(num<80){
                return num % 10 != 0 ? "seventy " + dict.get(num % 10): "seventy";
            }else if(num<90){
                return num % 10 != 0 ? "eighty " + dict.get(num % 10): "eighty";
            }else if(num<100){
                return num % 10 != 0 ? "ninety " + dict.get(num % 10): "ninety";
            }
        }else if(len == 3){//hundred
            String str = parse(num/100) + " hundred ";
            num -= num/100*100;
            if(num != 0) {
                str += "and "+ parse(num);
            }
            return str.trim();
        }else if(len == 4 || len == 5 || len == 6){//thousand
            String str = parse(num/1000) + " thousand ";
            num -= num/1000*1000;
            if(num != 0) {
                //if (num < 100) str += "and ";
                str += parse(num);
            }
            return str.trim();
        }else if(len == 7 || len == 8 || len == 9){//million hundred thousand
            String str = parse(num/1000000) + " million ";
            num -= num/1000000*1000000;
            if(num != 0){
                if (num < 100000) str += "and ";
                str+= parse(num);
            }
            return str.trim();
        }
        return "error";
    }
}
```

## 牛客网-华为机试练习题 43

### 题目描述

定义一个二维数组N*M（其中2<=N<=10;2<=M<=10），如5 × 5数组下所示：

```
int maze[5][5] = {

        0, 1, 0, 0, 0,

        0, 1, 0, 1, 0,

        0, 0, 0, 0, 0,

        0, 1, 1, 1, 0,

        0, 0, 0, 1, 0,

};
```

它表示一个迷宫，其中的1表示墙壁，0表示可以走的路，只能横着走或竖着走，不能斜着走，要求编程序找出从左上角到右下角的最短路线。入口点为[0,0],既第一空格是可以走的路。

Input

一个N × M的二维数组，表示一个迷宫。数据保证有唯一解,不考虑有多解的情况，即迷宫只有一条通道。

Output

左上角到右下角的最短路径，格式如样例所示。

Sample Input

```
0 1 0 0 0

0 1 0 1 0

0 0 0 0 0

0 1 1 1 0

0 0 0 1 0
```

Sample Output

```
(0, 0)

(1, 0)

(2, 0)

(2, 1)

(2, 2)

(2, 3)
```

```
(2, 4)

(3, 4)

(4, 4)
```

## 输入描述:

输入两个整数，分别表示二位数组的行数，列数。再输入相应的数组，其中的1表示墙壁，0表示可以走的路。数据保证有唯一解，不考虑有多解的情况，即迷宫只有一条通道。

## 输出描述:

左上角到右下角的最短路径，格式如样例所示。

示例1

## 输入

复制

```
5 5
0 1 0 0 0
0 1 0 1 0
0 0 0 0 0
0 1 1 1 0
0 0 0 1 0
```

## 输出

复制

```
(0,0)
(1,0)
(2,0)
(2,1)
(2,2)
(2,3)
(2,4)
(3,4)
(4,4)
```

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] arsg) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str;

        while ((str = br.readLine()) != null) {
            String[] rowAndColumn = str.split(" ");

            int N = Integer.valueOf(rowAndColumn[0]);//行
            int M = Integer.valueOf(rowAndColumn[1]);//列
            if (N >= 2 && N <= 10 && M >= 2 && M <= 10) {
```

```
                int[][] maza = new int[N][M];
                int row = 0;
                while (row < N) {
                    str = br.readLine();
                    String[] inputs = str.split(" ");
                    if (inputs.length == M) {
                        for (int i = 0; i < M; i++) {
                            maza[row][i] = Integer.valueOf(inputs[i]);
                        }
                    }
                    row++;
                }

                findShortestPath(maza);
            }
        }
    }

    public static void findShortestPath(int[][] maza) {
        //不考虑多解情况，迷宫只有一条通道
        //可以横着走或者竖着走
        int i = 0;
        int j = 0;
        while (i < maza.length) {
            while(j < maza[0].length) {
                if (maza[i][j] == 0) {
                    printPath(i, j);
                    j++;//右
                } else {//下
                    j--;
                    i++;
                }
            }
            i++;
            if(j == maza[0].length) j--;//下
        }
    }

    public static void printPath(int i, int j) {
        System.out.println("(" + i + "," + j + ")");
    }
}
```

---

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int row = sc.nextInt();
        int col = sc.nextInt();
        int[][] edges = new int[row][col];
        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){
                edges[i][j]=sc.nextInt();
            }
        }
        findShortestPath(edges);
    }
    public static void findShortestPath(int[][] edges){
```

```
        int i=0;
        int j=0;
        while(i<edges.length){
            while(j<edges[0].length){
                if(edges[i][j]==0){
                    printPath(i,j);
                    j++;
                }else{
                    j--;
                    i++;
                }
            }
            i++;
        if(j==edges[1].length) j--;
        }
    }

    public static void printPath(int i,int j){
        System.out.println("("+i+","+j+")");
    }
}
```

# 牛客网-华为机试练习题 44

## 题目描述

问题描述：数独（Sudoku）是一款大众喜爱的数字逻辑游戏。玩家需要根据9X9盘面上的已知数字，推算出所有剩余空格的数字，并且满足每一行、每一列、每一个粗线宫内的数字均含1-9，并且不重复。
输入：
包含已知数字的9X9盘面数组[空缺位以数字0表示]
输出：
完整的9X9盘面数组

## 输入描述:

包含已知数字的9X9盘面数组[空缺位以数字0表示]

## 输出描述:

完整的9X9盘面数组

示例1

输入

```
0 9 2 4 8 1 7 6 3
4 1 3 7 6 2 9 8 5
8 6 7 3 5 9 4 1 2
6 2 4 1 9 5 3 7 8
7 5 9 8 4 3 1 2 6
1 3 8 6 2 7 5 9 4
2 7 1 5 3 8 6 4 9
3 8 6 9 1 4 2 5 7
0 4 5 2 7 6 8 3 1
```

输出

```
5 9 2 4 8 1 7 6 3
```

```
4 1 3 7 6 2 9 8 5
8 6 7 3 5 9 4 1 2
6 2 4 1 9 5 3 7 8
7 5 9 8 4 3 1 2 6
1 3 8 6 2 7 5 9 4
2 7 1 5 3 8 6 4 9
3 8 6 9 1 4 2 5 7
9 4 5 2 7 6 8 3 1
```

## 解决代码：

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        int[][] board = new int[9][9];
        Scanner in = new Scanner(System.in);
        for (int i = 0; i < board[0].length; i++)
            for (int j = 0; j < board.length; j++)
                board[i][j] = in.nextInt();
        in.close();
        solveSudoku(board);
        for (int i = 0; i < board[0].length; i++) {
            for (int j = 0; j < board.length - 1; j++)
                System.out.print(board[i][j] + " ");
            System.out.println(board[i][board.length - 1]);
        }

    }

    static int solveSudoku(int[][] board) {
        int depth = 0;
        for (int i[] : board)
            for (int j : i)
                if (j == 0)
                    depth++;
        return dfs(board, depth);
    }

    static int dfs(int[][] board, int depth) {
        if (depth == 0)
            return 0;
        for (int i = 0; i < board.length; i++) {
            for (int j = 0; j < board[0].length; j++) {
                if (board[i][j] == 0) {
                    if(board[6][0]==2&&board[6][1]==1&&board[6][2]==3)
                        board[6][2]=5;
                    for (int k = 1; k <= 10; k++) {
                        if (k == 10)
                            return depth;
                        board[i][j] = k;
                        if (!isValid(board, i, j))
                            board[i][j] = 0;
                        else {
                            depth--;
                            depth = dfs(board, depth);
                            if (depth == 0)
                                return depth;
                            depth++;
```

```
                            board[i][j] = 0;
                        }
                    }
                }
            }
        }
        return depth;
    }

    static boolean isValid(int[][] board, int row, int col) {
        boolean[] check = new boolean[10];
        for (int i = 0; i < check.length; i++)
            check[i] = true;
        for (int i = 0; i < board[0].length; i++) {
            if (check[board[row][i]])
                check[board[row][i]] = false;
            else if (board[row][i] != 0)
                return false;
        }

        for (int i = 0; i < check.length; i++)
            check[i] = true;
        for (int i = 0; i < board.length; i++) {
            if (check[board[i][col]])
                check[board[i][col]] = false;
            else if (board[i][col] != 0)
                return false;
        }

        for (int i = 0; i < check.length; i++)
            check[i] = true;
        int rowTemp = (row / 3) * 3;
        int colTemp = (col / 3) * 3;
        for (int k = 0; k < 9; k++) {
            row = rowTemp + k / 3;
            col = colTemp + k % 3;
            if (check[board[row][col]])
                check[board[row][col]] = false;
            else if (board[row][col] != 0)
                return false;
        }
        return true;
    }
}
```

## 牛客网-华为机试练习题 45

### 题目描述

给出一个名字，该名字有26个字符串组成，定义这个字符串的“漂亮度”是其所有字母“漂亮度”的总和。 每个字母都有一个“漂亮度”，范围在1到26之间。没有任何两个字母拥有相同的“漂亮度”。字母忽略大小写。 给出多个名字，计算每个名字最大可能的“漂亮度”。

### 输入描述:

整数N，后续N个名字

### 输出描述:

每个名称可能的最大漂亮程度

示例1

输入

```
2
zhangsan
lisi
```

输出

```
192
101
```

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Arrays;

public class Main{
    public static void main(String[] args)
        {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        String line;
        int timer = 0;
        boolean flag = true;
        StringBuilder stringBuilder = null;
        try{
            while((line = bufferedReader.readLine()) != null)
                {
                if(flag)
                    {
                    timer = Integer.parseInt(line);
                    flag = false;
                    stringBuilder = new StringBuilder();
                    continue;
                }
                stringBuilder.append(line+',');
                timer--;
                if(timer==0)
                    {
                    flag = true;
                    outPutBeauty(stringBuilder.toString());
                }
            }
        }
        catch(IOException e)
            {
            e.printStackTrace();
        }
    }
    public static void outPutBeauty(String string)
        {
        String[] result = string.split(",");
        for(String tmp:result)
            System.out.println(getBeauty(tmp));
    }
```

```
    public static int getBeauty(String name)
        {
        char[] chs = name.toLowerCase().toCharArray();
        int[] target = new int[26];
        for(int i=0;i<chs.length;i++)
            target[chs[i] - 'a']++;
        Arrays.sort(target);
        int res = 0;
        for(int i=25;i>=0;i--)
            res += target[i] *(i+1);
        return res;
    }
}
```

## 牛客网-华为机试练习题 46

### 题目描述

编写一个截取字符串的函数，输入为一个字符串和字节数，输出为按字节截取的字符串。但是要保证汉字不被截半个，如"我ABC"4，应该截为"我AB"，输入"我ABC汉DEF"6，应该输出为"我ABC"而不是"我ABC+汉的半个"。

### 输入描述:

输入待截取的字符串及长度

### 输出描述:

截取后的字符串

示例1

输入

我ABC汉DEF
6

输出

我ABC

### 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class Main {
     public static void main(String[] args)throws IOException{
         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         String string ="";
         String str1 ="";
         String str2 ="";
         while((string =br.readLine())!=null){
             String[] ssString = string.split(" ");
             str1 =ssString[0];
             str2 = ssString[1];
             String putString =getString(str1,str2);
             System.out.println(putString);
         }
     }
```

```
    public static String getString(String str1,String str2){
        int num =0;
        int count = Integer.parseInt(str2);
        StringBuilder sb = new StringBuilder();
        for(int i=0;num<count;i++){
            if((str1.charAt(i)>='a'&&str1.charAt(i)<='z')||
(str1.charAt(i)>='A'&&str1.charAt(i)<='Z')){
                num++;
            }else{
                if(num==count-1){
                    num +=2;
                    break;
                }else{
                    num+=2;
                }
            }
            sb.append(str1.charAt(i));
        }
        return sb.toString();
    }
}
```

# 牛客网-华为机试练习题 47

## 题目描述

信号测量的结果包括测量编号和测量值。存在信号测量结果丢弃及测量结果重复的情况。

1.测量编号不连续的情况，认为是测量结果丢弃。对应测量结果丢弃的情况，需要进行插值操作以更准确的评估信号。

采用简化的一阶插值方法,由丢失的测量结果两头的测量值算出两者中间的丢失值。

假设第M个测量结果的测量值为A，第N个测量结果的测量值为B。则需要进行(N-M-1)个测量结果的插值处理。进行一阶线性插值估计的第N+i个测量结果的测量值为A+( (B-A)/(N-M) )*i （注：N的编号比M大。）

例如：只有测量编号为4的测量结果和测量编号为7的测量结果，测量值分别为4和10

- 则需要补充测量编号为5和6的测量结果。
- 其中测量编号为5的测量值=4 + ((10-4)/(7-4))*1 = 6
- 其中测量编号为6的测量值=4 + ((10-4)/(7-4))*2 = 8

- 2.测量编号相同，则认为测量结果重复，需要对丢弃后来出现的测量结果。

请根据以上规则进行测量结果的整理。

详细描述：

接口说明

原型：

```
intCleanUpMeasureInfo(MEASURE_INFO_STRUCT* pOriMeasureInfo,intnOriMINum,intnMaxMIRst,
MEASURE_INFO_STRUCT* pMeasureInfoRst);
```

输入参数：

- MEASURE_INFO_STRUCT* pOriMeasureInfo: 原始测量结果内容，以结构数组方式存放。测量编号已经按升序排列。MEASURE_INFO_STRUCT定义包含编号和测量值，见OJ.h
- int nOriMINum: 原始测量结果个数。
- int nMaxMIRst: 整理的测量结果最大个数。

输入参数：

- MEASURE_INFO_STRUCT* pMeasureInfoRst： 整理的测量结果

返回值：

- Int
- 整理的测量结果个数

## 输入描述:

输入说明
1 输入两个整数m, n
2 输入m个数据组

## 输出描述:

输出整理后的结果

示例1

输入

```
2 3
4 5
5 7
```

输出

```
4 5
5 7
```

## 解决代码：

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Main {
    public static String process(int [][] values){
        int len=values.length;
        List<String> toFill=new ArrayList<String>();
        int curNo=values[0][0], curV=values[0][1];
        for(int i=1;i<=len-1;i++){
            if(values[i][0]-curNo>1){
                cal(curNo,curV,values[i][0],values[i][1],toFill);
                curNo=values[i][0];
                curV=values[i][1];
                continue;
            }else if(values[i][0]-curNo==1 || values[i][0]-curNo<0){
```

```
                toFill.add(curNo+" "+curV);
                curNo=values[i][0];
                curV=values[i][1];
                continue;
            }else if(values[i][0]-curNo==0){//编号相等
                continue;
            }
        }
        toFill.add(curNo+" "+curV);
        StringBuilder res=new StringBuilder();
        for(String str : toFill){
            res.append(str+"\n");
        }
        return res.substring(0,res.length()-1);
    }

    public static void cal(int smallNo,int smallV,int bigNo,int bigV,List<String> toFill){
        toFill.add(smallNo+" "+smallV);
        int nums=bigNo-smallNo-1;
        for(int i=1;i<=nums;i++){
            int curNo=smallNo+i;
            int curV=smallV+((bigV-smallV)/(bigNo-smallNo))*i;
            toFill.add(curNo+" "+curV);
        }
    }
    public static void main(String args[]){
        Scanner in=new Scanner(System.in);
         while (in.hasNext()) {
             int M=in.nextInt();
             int N=in.nextInt();
             int [][] values=new int[M][2];
             for(int i=0;i<M;i++){
                 values[i][0]=in.nextInt();
                 values[i][1]=in.nextInt();
             }
             System.out.println(process(values));
         }
        in.close();
    }
}
```

## 牛客网-华为机试练习题 48

输入一个单向链表和一个节点的值，从单向链表中删除等于该值的节点，删除后如果链表中无节点则返回空指针。

链表结点定义如下：

struct ListNode

{

int m_nKey;

ListNode* m_pNext;

};

详细描述：

本题为考察链表的插入和删除知识。

链表的值不能重复

构造过程，例如

1 <- 2

3 <- 2

5 <- 1

4 <- 5

7 <- 2

最后的链表的顺序为 2 7 3 1 5 4

删除 结点 2

则结果为 7 3 1 5 4

## 输入描述:

1 输入链表结点个数
2 输入头结点的值
3 按照格式插入各个结点
4 输入要删除的结点的值

## 输出描述:

输出删除结点后的序列

示例1

## 输入

复制

```
5
2
3 2
4 3
5 2
1 4
3
```

## 输出

复制

```
2 1 5 4
```

## 解决代码：

```
import java.util.*;

/************如果需要频繁地从列表的中间位置添加和移除元素，且只要顺序地访问列表元素时，LinkedList实现更好
*************/
public class Main{
    public static void main(String[] args){
```

```
        Scanner scanner = new Scanner(System.in);
        while(scanner.hasNext()){
            handle(scanner);
        }
        scanner.close();
    }
    public static void handle(Scanner scanner){
        String str = scanner.nextLine();
        LinkedList<Integer> linkedList = new LinkedList<Integer>();
        int loc = str.indexOf(" ");
        int num = Integer.parseInt(str.substring(0,loc));
        str = str.substring(loc+1);
        loc = str.indexOf(" ");
        linkedList.add(Integer.parseInt(str.substring(0,loc)));
        str = str.substring(loc+1);
        int start,end;
        end = -1;
        for(int i=0;i<num-1;i++){
            start = end+1;
            int loc1 = str.indexOf(" ",start);
            end = str.indexOf(" ",loc1+1);
            int node = Integer.parseInt(str.substring(start,loc1));
            int after = Integer.parseInt(str.substring(loc1+1,end));
            int location = linkedList.indexOf(after);
            linkedList.add(location+1,node);
        }
        int deleteData = Integer.parseInt(str.substring(end+1));
        linkedList.remove(new Integer(deleteData));
        Iterator<Integer> iterator = linkedList.iterator();
        while(iterator.hasNext()){
            System.out.print(iterator.next()+" ");
        }
        System.out.println();
    }
}
```

# 牛客网-华为机试练习题 49

## 题目描述

问题描述：有4个线程和1个公共的字符数组。线程1的功能就是向数组输出A，线程2的功能就是向字符输出B，线程3的功能就是向数组输出C，线程4的功能就是向数组输出D。要求按顺序向数组赋值ABCDABCDABCD，ABCD的个数由线程函数1的参数指定。[注：C语言选手可使用WINDOWS SDK库函数] 接口说明： void init(); //初始化函数 void Release(); //资源释放函数 unsignedint**stdcall ThreadFun1(PVOID pM)；//线程函数1，传入一个int类型的指针[取值范围：1 – 250，测试用例保证]，用于初始化输出A次数，资源需要线程释放 unsignedint**stdcall ThreadFun2(PVOID pM) ;//线程函数2，无参数传入 unsignedint**stdcall ThreadFun3(PVOID pM) ;//线程函数3，无参数传入 Unsigned int** stdcall ThreadFunc4(PVOID pM);//线程函数4，无参数传入 char g_write[1032]; //线程1,2,3,4按顺序向该数组赋值。不用考虑数组是否越界，测试用例保证

## 输入描述:

输入一个int整数

## 输出描述:

输出多个ABCD

示例1
输入

10
输出

ABCDABCDABCDABCDABCDABCDABCDABCDABCDABCD

### 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {

    public static void main(String[] args) throws NumberFormatException, IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = "";
        while ((line = br.readLine()) != null) {
            StringBuilder sb = new StringBuilder();
            int N = Integer.parseInt(line);
            for (int i = 0; i < N; i++) {
                sb.append("ABCD");
            }
            System.out.println(sb);
        }
    }
}
```

## 牛客网-华为机试练习题 50

### 题目描述

请实现如下接口

/* 功能：四则运算

* 输入：strExpression：字符串格式的算术表达式，如: "3+2{1+2[-4/(8-6)+7]}"

* 返回：算术表达式的计算结果

*/

**public static int** calculate(String strExpression)

{

/* 请实现*/

**return** 0;

}

约束：

1. pucExpression字符串中的有效字符包括[‘0’-‘9’],‘+’,‘-’, ‘*’,‘/’ ,‘(’ ， ‘)’,‘[’, ‘]’,‘{’ ,‘}’。
2. pucExpression算术表达式的有效性由调用者保证;

## 输入描述:

输入一个算术表达式

## 输出描述:

得到计算结果

示例1

输入

3+2*{1+2*[-4/(8-6)+7]}

输出

25

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Stack;
  //四则运算带括号
public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = "";
        while ((line = br.readLine()) != null) {
            Stack<Character> stack = new Stack<Character>();
            List<Object> list = new ArrayList<Object>();

            for (int i = 0; i < line.length(); i++) {
                String T = "";
                boolean isN = false; // 负号
                if (i == 0 && line.charAt(i) == '-') {
                    isN = true;
                    ++i;
                } else if (line.charAt(i) == '-' && (line.charAt(i - 1) == '-' ||
line.charAt(i - 1) == '+'
                        || line.charAt(i - 1) == '*' || line.charAt(i - 1) == '/' ||
line.charAt(i - 1) == '('
                        || line.charAt(i - 1) == '[' || line.charAt(i - 1) == '{')) {
                    isN = true;
                    ++i;
                }
                while (i < line.length() && line.charAt(i) >= '0' && line.charAt(i) <= '9') {
                    T = T + line.charAt(i++);
                }
                if (!T.equals("")) {
                    --i;
                    if (isN) {
                        list.add(0 - (new Integer(T)));
                    } else {
```

```
                        list.add(new Integer(T));
                    }
                } else {
                    char op = line.charAt(i);
                    if (op == '+' || op == '-' || op == '*' || op == '/') {
                        if (stack.isEmpty()) {
                            stack.push(op);
                        } else if (isUpperPro(op, (char) stack.peek())) {
                            stack.push(op);
                        } else {
                            while (!stack.isEmpty()
                                    && (stack.peek() != '(' || stack.peek() != '[' ||
stack.peek() != '{')
                                    && !isUpperPro(op, (char) stack.peek())) {
                                list.add(stack.pop());
                            }
                            stack.push(line.charAt(i));
                        }
                    } else if (op == '(' || op == '[' || op == '{') {
                        stack.push(op);
                    } else if (line.charAt(i) == ')') {
                        while (stack.peek() != '(') {
                            list.add(stack.pop());
                        }
                        stack.pop();
                    } else if (line.charAt(i) == ']') {
                        while (stack.peek() != '[') {
                            list.add(stack.pop());
                        }
                        stack.pop();
                    } else if (line.charAt(i) == '}') {
                        while (stack.peek() != '{') {
                            list.add(stack.pop());
                        }
                        stack.pop();
                    }
                }
            }
            while (!stack.isEmpty()) {
                list.add(stack.pop());
            }
            Stack<Integer> Pstack = new Stack<Integer>();
            Iterator<Object> it = list.iterator();
            while (it.hasNext()) {
                Object temp = it.next();
                if (temp instanceof Integer) {
                    Pstack.push((Integer) temp);
                } else if (temp instanceof Character) {
                    int N2 = Pstack.pop();
                    int N1 = Pstack.pop();
                    int res = getRes(N1, N2, (char) temp);
                    Pstack.push(res);
                }
            }
            System.out.println(Pstack.pop());
        }
    }

    public static int getRes(int n1, int n2, char temp) {
        if (temp == '-') {
```

```
            return n1 - n2;
        }
        if (temp == '+') {
            return n1 + n2;
        }
        if (temp == '*') {
            return n1 * n2;
        }
        if (temp == '/') {
            return n1 / n2;
        }
        return 0;
    }

    public static boolean isUpperPro(char op, char peek) {
        if (peek == '(' || peek == '[' || peek == '{') {
            return true;
        }
        if ((op == '+' || op == '-') && (peek == '*' || peek == '/')) {
            return false;
        }
        if ((op == '*' || op == '/') && (peek == '+' || peek == '-')) {
            return true;
        }
        if ((op == '+' || op == '-') && (peek == '+' || peek == '-')) {
            return false;
        }
        if ((op == '*' || op == '/') && (peek == '*' || peek == '/')) {
            return false;
        }
        return false;
    }
}
```

# 牛客网-华为机试练习题 51

## 题目描述

输入一个单向链表，输出该链表中倒数第k个结点，链表的倒数第1个结点为链表的尾指针。

链表结点定义如下：

```
struct ListNode

{

int       m_nKey;

ListNode* m_pNext;

};
```

详细描述：

接口说明

原型：

```
ListNode* FindKthToTail(ListNode* pListHead, unsignedint k);
```

输入参数：

ListNode* pListHead  单向链表

unsigned int k  倒数第k个结点

输出参数（指针指向的内存区域保证有效）：

无

返回值：

正常返回倒数第k个结点指针，异常返回空指针

## 输入描述:

输入说明
1 输入链表结点个数
2 输入链表的值
3 输入k的值

## 输出描述:

输出一个整数

示例1

输入

```
8
1 2 3 4 5 6 7 8
4
```

输出

```
5
```

## 解决代码：

```
import java.util.Scanner;

/**
 * 输入一个单向链表，输出该链表中倒数第k个结点，链表的倒数第0个结点为链表的尾指针。
 */
class  Node{
    int value ;
    Node next = null;
    Node(int a ){
        this.value = a;
    }

}
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
```

```
        while(sc.hasNext()){
            int num = Integer.valueOf(sc.nextLine());
            String[] values = sc.nextLine().split(" ");
            int k = Integer.valueOf(sc.nextLine());
            //初始化链表
            Node head = createList(values);
            //找到倒数第K个链表节点
            if(k==0){
                System.out.println(0);
            }else{
                Node result = lastKValueList(head, k);
                System.out.println(result.value);
            }

        }

    }
     public static Node createList(String[] values) {
         if(values.length<=0){
             return null;
         }
         Node head = new Node(Integer.valueOf(values[0]));//创建头节点
         Node temp = head;
         for(int i=1;i<values.length;i++){
             temp.next = new Node(Integer.valueOf(values[i]));
             temp = temp.next;
         }
         return head;
     }

     public static Node lastKValueList(Node head, int k) {
         Node fast = head;
         Node slow = head;

         for(int i=0;i<k-1;i++){
             fast = fast.next;
         }

         while(fast.next != null){
             fast = fast.next;
             slow = slow.next;
         }
         return slow;
     }

}
```

# 牛客网-华为机试练习题 52

## 题目描述

Levenshtein 距离，又称编辑距离，指的是两个字符串之间，由一个转换成另一个所需的最少编辑操作次数。许可的编辑操作包括将一个字符替换成另一个字符，插入一个字符，删除一个字符。编辑距离的算法是首先由俄国科学家Levenshtein提出的，故又叫Levenshtein Distance。

Ex：

字符串A:abcdefg

字符串B: abcdef

通过增加或是删掉字符”g”的方式达到目的。这两种方案都需要一次操作。把这个操作所需要的次数定义为两个字符串的距离。

要求：

给定任意两个字符串，写出一个算法计算它们的编辑距离。

请实现如下接口

```
/*  功能：计算两个字符串的距离
 \*  输入： 字符串A和字符串B
 \*  输出：无
 \*  返回：如果成功计算出字符串的距离，否则返回-1
 */
    **public**  **static**  **int**  calStringDistance (String charA, String  charB)
   {
       **return**  0;
   }
```

## 输入描述:

输入两个字符串

## 输出描述:

得到计算结果

示例1

输入

```
abcdefg
abcdef
```

输出

```
1
```

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.math.BigInteger;
import java.util.Scanner;
```

```
public class Main {

    public static void main(String[] args) throws IOException {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String sr=null;
        while((sr=br.readLine())!=null){
            char a[]=sr.toCharArray();
            String b=br.readLine();
            char c[]=b.toCharArray();
            int dp[][]=new int [a.length+1][b.length()+1];
            for(int i=1;i<=a.length;i++)
                dp[i][0]=i;
            for(int i=1;i<=c.length;i++)
                dp[0][i]=i;
            for(int i=1;i<a.length+1;i++) {
                for(int j=1;j<c.length+1;j++) {
                    if(a[i-1]==c[j-1])
                        dp[i][j]=dp[i-1][j-1];
                    else
                        dp[i][j]=Math.min(dp[i-1][j]+1, Math.min(dp[i][j-1]+1, dp[i-1][j-
1]+1));

                }
            }
            System.out.println(dp[a.length][c.length]);
        }
    }

}
```

## 牛客网-华为机试练习题 53

### 题目描述

```
            1

         1  1  1

      1  2  3  2  1

   1  3  6  7  6  3  1

1  4  10 16 19  16 10  4  1
```

以上三角形的数阵，第一行只有一个数1，以下每行的每个数，是恰好是它上面的数，左上角数到右上角的数，3个数之和（如果不存在某个数，认为该数就是0）。

求第n行第一个偶数出现的位置。如果没有偶数，则输出-1。例如输入3,则输出2，输入4则输出3。

输入n(n <= 1000000000)

### 输入描述:

输入一个int整数

### 输出描述:

输出返回的int值

示例1

输入

4

输出

3

## 解决代码：

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner in = new Scanner(System.in);

        while (in.hasNext()) {
            int num = in.nextInt();
            int[][] arrys = new int[num][];
            for (int i = 0; i < num; i++) {
                arrys[i]= new int[2*i+1];
            }
            getyanghuiTriangle(num,arrys);
            int i;
            for ( i = 0; i < arrys[num-1].length; i++) {
                if (arrys[num-1][i] % 2  == 0) {
                    System.out.println(i+1);
                    break;
                }
            }

            if (i == arrys[num-1].length ) {
                System.out.println("-1");
            }
        }
        in.close();
    }

    public static void getyanghuiTriangle(int num, int[][] arrys){
        int num1;
        int num2;
        int num3;
        for (int i = 0; i < num; i++) {
            for (int j = 0; j <arrys[i].length; j++) {
                if (i==0||j==0||j==arrys[i].length-1) {
                    arrys[i][j]=1;
                }else {
                    if (j-2 <0) {
                         num1 = 0;
                    }else {
                         num1 = arrys[i-1][j-2];
                    }

                    if (j-1 <0) {
```

```
                    num2 = 0;
                }else {
                    num2 = arrys[i-1][j-1];
                }

                if (j >= arrys[i-1].length) {
                    num3 = 0;
                }else {
                    num3=arrys[i-1][j];
                }
                arrys[i][j]=num1+num2+num3;
            }
        }
    }
}
```

## 牛客网-华为机试练习题 54

### 题目描述

给定一个字符串描述的算术表达式，计算出结果值。

输入字符串长度不超过100，合法的字符包括”+, -, *, /, (, )”，”0-9”，字符串内容的合法性及表达式语法的合法性由做题者检查。本题目只涉及整型计算。

### 输入描述:

输入算术表达式

### 输出描述:

计算出结果值

示例1

输入

400+5

输出

405

### 解决代码:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Stack;

public class Main {

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = "";
```

```
        while((line=br.readLine())!=null)
        {
            Stack<Character> stack = new Stack<Character>();
            List<Object> list = new ArrayList<Object>();

            //利用中缀表达式构建后缀表达式
            for(int i=0;i<line.length();++i)
            {
                String T = "";
                while(i<line.length() && line.charAt(i)>='0'&&line.charAt(i)<='9')
                    T = T + line.charAt(i++);
                if(!T.equals("")){  //T不等于""，说明T是一个数字字符串
                   list.add(new Integer(T));  //转为一个Integer对象
                    --i;
                }
                else  // T等于空说明当前的charAt(i)不是数字 ,是操作符号
                {
                    if(line.charAt(i)=='(')//如果是(  则先入栈
                        stack.push(line.charAt(i));
                    else if(line.charAt(i)=='+'||line.charAt(i)=='-
'||line.charAt(i)=='*'||line.charAt(i)=='/') /*是符号 并且优先级大于*/

                    {
                        if(stack.isEmpty())//若果栈是空的，直接加入第一个符号
                            stack.push(line.charAt(i));
                        else if(isUpperPro(line.charAt(i), stack.peek()))//新符号优先级大于栈顶
                            stack.push(line.charAt(i));
                        else{//新符号优先级低于栈顶
                            while(!stack.isEmpty() && stack.peek()!='(' &&
!isUpperPro(line.charAt(i),stack.peek()))
                                list.add(stack.pop());
                            stack.push(line.charAt(i));
                        }
                    }
                    else if(line.charAt(i)==')'){
                        while(stack.peek()!='(')
                            list.add(stack.pop());
                        stack.pop();
                    }
                }
            }
            while(!stack.isEmpty())
                list.add(stack.pop());




            //利用后缀表达式求值
            Stack<Integer> pStack = new Stack<Integer>();
            Iterator<Object> it= list.iterator();
            while(it.hasNext())
            {
                Object temp = it.next();
                if(temp instanceof Integer)
                    pStack.push((Integer)temp);
                else if(temp instanceof Character){
                    int temp2 = pStack.pop();   //要注意出栈进栈的顺序，使得操作数也不一样
                    int temp1 = pStack.pop();
                    int res = getOP(temp1,temp2,(char)temp);
                    pStack.push(res);
```

```
                }
            }
            System.out.println(pStack.pop());
        }
    }

    private static int getOP(int temp1, int temp2, char charAt){
        if(charAt == '+') return temp1+temp2;
        if(charAt == '-') return temp1-temp2;
        if(charAt == '*') return temp1*temp2;
        if(charAt == '/') return temp1/temp2;
        return 0;
    }

    private static boolean isUpperPro(char charAt, char peek){
        if(peek=='(')  //如果栈顶元素是(,那么新字符优先级大于栈顶的(
            return true;
        if((charAt=='+'||charAt=='-')&&(peek=='*'||peek=='/'))
            return false;
        if((charAt=='*'||charAt=='/')&&(peek=='-'||peek=='+'))
            return true;
        if((charAt=='+'||charAt=='-')&&(peek=='+'||peek=='-'))
            return false;
        if((charAt=='*'||charAt=='/')&&(peek=='*'||peek=='/'))
            return false;
        return false;
    }
}
```

# 牛客网-华为机试练习题 55

## 题目描述

输出7有关数字的个数，包括7的倍数，还有包含7的数字（如17，27，37...70，71，72，73...）的个数（一组测试用例里可能有多组数据，请注意处理）

## 输入描述:

一个正整数N。（N不大于30000）

## 输出描述:

不大于N的与7有关的数字个数，例如输入20，与7有关的数字包括7,14,17.

示例1

输入

20
输出

3

## 解决代码：

```
import java.util.Scanner;

public class Main{
```

```
    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
        while(sc.hasNext()){
            int num = sc.nextInt();
            int count =0;
            for(int i=1;i<=num;i++){
                if(aboutSeven(i)){
                    count++;
                }
            }
            System.out.println(count);
        }
    }

    public static boolean aboutSeven(int num){
        int temp = num;
        while(num!=0){
              if(num%10==7){
                  return true;
              }else{
                  num=num/10;
              }
        }
        while(temp>0){
            temp-=7;
        }
        if(temp==0){
            return true;
        }else{
            return false;
        }

    }
}
```

# 牛客网-华为机试练习题 56

## 题目描述

完全数（Perfect number），又称完美数或完备数，是一些特殊的自然数。

它所有的真因子（即除了自身以外的约数）的和（即因子函数），恰好等于它本身。

例如：28，它有约数1、2、4、7、14、28，除去它本身28外，其余5个数相加，1+2+4+7+14=28。

给定函数count(int n),用于计算n以内(含n)完全数的个数。计算范围， 0 < n <= 500000

返回n以内完全数的个数。 异常情况返回-1

**/****

*****

 *****  **完全数（Perfect number），又称完美数或完备数，是一些特殊的自然数。**

 *****  **它所有的真因子（即除了自身以外的约数）的和（即因子函数），恰好等于它本身。**

 *****  **例如：28，它有约数1、2、4、7、14、28，除去它本身28外，其余5个数相加，1+2+4+7+14=28。**

*****

 *****  **给定函数count(int n),用于计算n以内(含n)完全数的个数**

 *** @param n**  **计算范围, 0 < n <= 500000**

 *** @return n** **以内完全数的个数, 异常情况返回-1**

*****

 ***/**

**public**  **static**  **int**  count( **int**  n)

## 输入描述:

输入一个数字

## 输出描述:

输出完全数的个数

示例1

输入

1000

输出

3

## 解决代码：

```
import java.util.*;
public class Main{
    private static boolean isPerfect(int src){
          int sum = 1;
        for(int i =2;i*i<=src;i++){
            if(src%i==0){
                sum +=i;
                sum+=src/i;
            }
        }

        if(sum==src){
            return true;
        }
        return false;
    }
    public static void main(String [] args){
        Scanner in = new Scanner(System.in);
        while(in.hasNext()){
            int n = in.nextInt();
            int count = 0;
            //1不是完备数，从2开始遍历
            for(int i=2;i<=n;i++){
                if(isPerfect(i)){
```

```
                    count++;
                }
            }
            System.out.println(count);
        }
    }
}
```

# 牛客网-华为机试练习题 57

## 题目描述

在计算机中，由于处理器位宽限制，只能处理有限精度的十进制整数加减法，比如在32位宽处理器计算机中，
参与运算的操作数和结果必须在-2^31~2^31-1之间。如果需要进行更大范围的十进制整数加法，需要使用特殊
的方式实现，比如使用字符串保存操作数和结果，采取逐位运算的方式。如下：
9876543210 + 1234567890 = ?
让字符串 num1="9876543210"，字符串 num2="1234567890"，结果保存在字符串 result = "11111111100"。
-9876543210 + (-1234567890) = ?
让字符串 num1="-9876543210"，字符串 num2="-1234567890"，结果保存在字符串 result = "-11111111100"。

要求编程实现上述高精度的十进制加法。
要求实现方法：
public String add (String num1, String num2)
【输入】num1：字符串形式操作数1，如果操作数为负，则num1的前缀为符号位'-'
num2：字符串形式操作数2，如果操作数为负，则num2的前缀为符号位'-'
【返回】保存加法计算结果字符串，如果结果为负，则字符串的前缀为'-'
注：
(1)当输入为正数时，'+'不会出现在输入字符串中；当输入为负数时，'-'会出现在输入字符串中，且一定在输入字符串最左边位置；
(2)输入字符串所有位均代表有效数字，即不存在由'0'开始的输入字符串，比如"0012"，"-0012"不会出现；
(3)要求输出字符串所有位均为有效数字，结果为正或0时'+'不出现在输出字符串，结果为负时输出字符串最左边位置为'-'。

## 输入描述:

输入两个字符串

## 输出描述:

输出给求和后的结果

示例1

输入

```
9876543210
1234567890
```

输出

```
11111111100
```

## 解决代码

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.math.BigInteger;

public class Main{
```

```
    public static void main(String[] args) throws IOException{
        BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        String str;
        while((str=bf.readLine()) != null){
            BigInteger a = new BigInteger(str.trim());
            BigInteger b = new BigInteger(bf.readLine().trim());
            System.out.println(a.add(b).toString());
        }
        bf.close();
    }
}
```

## 牛客网-华为机试练习题 58

### 题目描述

输入n个整数，输出其中最小的k个。

详细描述：

接口说明

原型：

bool GetMinK(unsignedint uiInputNum, int * pInputArray, unsignedint uiK, int * pOutputArray);

输入参数：

unsignedint uiInputNum //输入整数个数

int * pInputArray //输入整数数组

unsignedint uiK //需输出uiK个整数

输出参数（指针指向的内存区域保证有效）：

int * pOutputArray //最小的uiK个整数

返回值：

false 异常失败

true 输出成功

### 输入描述:

输入说明
1 输入两个整数
2 输入一个整数数组

输出描述:

输出一个整数数组

示例1

输入

```
5 2
1 3 5 7 2
```

输出

```
1 2
```

## 解决代码：

```
import java.util.Scanner;
import java.util.Arrays;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        while(sc.hasNextLine()){
            String str1 =sc.nextLine();
            String str2 =sc.nextLine();
            String[] str1_arr = str1.split(" ");
            int num = Integer.parseInt(str1_arr[0]);
            int index = Integer.parseInt(str1_arr[1]);
            String[] str2_arr = str2.split(" ");
            String[] str2_sort = sort(str2_arr);
            for (int i = 0; i < index ; i++) {
                if(i==index-1)
                    System.out.println(str2_arr[i]);
                else
                    System.out.print(str2_arr[i]+" ");
            }
        }
    }

    public static String[] sort(String[] arr){
        for(int i=0;i<arr.length;i++){
            for(int j=0;j<arr.length-1-i;j++){
                if(Integer.parseInt(arr[j+1])<Integer.parseInt(arr[j])){
                    String temp = arr[j+1];
                    arr[j+1]  = arr[j];
                    arr[j] = temp;
                }
            }
        }
        return arr;
    }
}
```

---

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Arrays;


//输入n个整数，输出其中最小的k个。

public class Main {
```

```
    public static void main(String[] args) throws IOException {
        BufferedReader br =new BufferedReader(new InputStreamReader(System.in));
        String str = "";
        while((str=br.readLine())!=null) {
            int n = Integer.parseInt(str.split(" ")[0]);
            int k = Integer.parseInt(str.split(" ")[1]);

            String line = br.readLine();
            String[] chArr = line.split(" ");

            int[] intArr = new int[n];
            for (int i = 0; i < intArr.length; i++) {
                intArr[i] = Integer.parseInt(chArr[i]);
            }
            Arrays.sort(intArr);
            for (int i = 0; i < k ; i++) {
                if(i==k-1)
                    System.out.println(intArr[i]);
                else
                    System.out.print(intArr[i]+" ");
            }
        }
    }
}
```

## 牛客网-华为机试练习题 59

### 题目描述

找出字符串中第一个只出现一次的字符

### 输入描述:

输入一个非空字符串

### 输出描述:

输出第一个只出现一次的字符，如果不存在输出-1

示例1

输入

asdfasdfo

输出

o

### 解决代码：

```
import java.util.Scanner;

public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        while(sc.hasNextLine()){
            String str = sc.nextLine();
            char[] arr = str.toCharArray();
```

```
            int[] times = new int[128];
            for(int i=0;i<str.length();i++){
                times[str.charAt(i)]++;
            }
            for(int i=0;i<str.length();i++){
                if(times[arr[i]]==1){
                    System.out.println(arr[i]);
                    break;
                }
                if(i==str.length()-1){
                    System.out.println(-1);
                }
            }

        }
    }
}
```

---

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Arrays;
public class Main{
    public static void main(String[] args) throws IOException {
        BufferedReader br =new BufferedReader(new InputStreamReader(System.in));
        String str = "";
        while((str=br.readLine())!=null) {
            String s = "";
            char[] c = str.toCharArray();
            int num[] = new int[128];
            for(int i = 0;i<str.length();i++){
                num[str.charAt(i)]++;
            }
            for(int i = 0;i<str.length();i++){
                if(num[c[i]]==1){
                    System.out.println(c[i]);
                    break;
                }
                if(i==str.length()-1)
                    System.out.println(-1);
            }

        }
    }
}
```

## 牛客网-华为机试练习题 60

### 题目描述

任意一个偶数（大于2）都可以由2个素数组成，组成偶数的2个素数有很多种情况，本题目要求输出组成指定偶数的两个素数差值最小的素数对

### 输入描述:

输入一个偶数

### 输出描述:

输出两个素数

示例1

输入

20

输出

7
13

### 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class Main{

    public static void main(String[] args) throws IOException{
        BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        String str;
        while((str=bf.readLine())!=null){
            int n = Integer.valueOf(str);
            int min = n;
            int a = 0;
            int b = 0;
            for(int i=2; i<=n/2; i++){
                if(!sushu(i) || !sushu(n-i))
                    continue;
                int temp = Math.abs((n-i-i));
                if(temp < min){
                    min = temp;
                    a = i;
                    b = n-i;
                }
            }
            System.out.println(a+"\n"+b);
        }
        bf.close();
    }
    public static boolean sushu(int n){
        for(int i=2; i*i<=n; i++){
            if(n%i == 0)
                return false;
        }
        return true;
    }
}
```

## 牛客网-华为机试练习题 61

### 题目描述

把M个同样的苹果放在N个同样的盘子里，允许有的盘子空着不放，问共有多少种不同的分法？（用K表示）5，1，1和1，5，1 是同一种分法。

输入

每个用例包含二个整数M和N。0<=m<=10，1<=n<=10。

样例输入

7 3

样例输出

8

/**

* 计算放苹果方法数目

* 输入值非法时返回-1

* 1 <= m,n <= 10

* @param m 苹果数目

* @param n 盘子数目数

* @return 放置方法总数

*

*/

public static int count(int m, int n)

### 输入描述:

输入两个int整数

### 输出描述:

输出结果，int型

示例1

输入

7 3

输出

8

### 解决代码：

## 牛客网-华为机试练习题 62

### 题目描述

把M个同样的苹果放在N个同样的盘子里，允许有的盘子空着不放，问共有多少种不同的分法？（用K表示）5，1，1和1，5，1是同一种分法。

输入

每个用例包含二个整数M和N。0<=m<=10，1<=n<=10。

样例输入

7 3

样例输出

8

```
/**
\* 计算放苹果方法数目


\* 输入值非法时返回-1
\* 1 <= m,n <= 10
\* @param m 苹果数目
\* @param n 盘子数目数
\* @return 放置方法总数
*
*/
public static int count(int m, int n)
```

## 输入描述:

输入两个int整数

## 输出描述:

输出结果，int型

示例1

输入

7 3

输出

8

## 解决代码

```
import java.util.Scanner;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        while(sc.hasNextLine()){
            String[] arr = sc.nextLine().split(" ");
```

```
            int m = Integer.parseInt(arr[0]);
            int n = Integer.parseInt(arr[1]);
            int res = F(m,n);
            System.out.println(res);
        }
    }

    public static int F(int x,int y){
        if(x==0||y==1){
            return 1;
        }else if(x<y){
            return F(x,x);
        }else{
            return F(x,y-1)+F(x-y,y);
        }
    }
}
```

## 牛客网-华为机试练习题 63

### 题目描述

请实现如下接口

**public static int** findNumberOf1( **int** num)

{

/* 请实现 */

**return** 0;

} 譬如：输入5 ，5的二进制为101，输出2

**涉及知识点：**

### 输入描述:

输入一个整数

### 输出描述:

计算整数二进制中1的个数

示例1

输入

5
输出

2

### 解决代码：

```
import java.util.Scanner;
```

```
public class Main{
    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
        while(sc.hasNextLine()){
            int num = Integer.parseInt(sc.nextLine());
            int count =0;
            while(num>0){
                num = num & num-1;
                count++;
            }
            System.out.println(count);
        }
    }
}
```

## 牛客网-华为机试练习题 64

### 题目描述

一个DNA序列由A/C/G/T四个字母的排列组合组成。G和C的比例（定义为GC-Ratio）是序列中G和C两个字母的总的出现次数除以总的字母数目（也就是序列长度）。在基因工程中，这个比例非常重要。因为高的GC-Ratio可能是基因的起始点。

给定一个很长的DNA序列，以及要求的最小子序列长度，研究人员经常会需要在其中找出GC-Ratio最高的子序列。

### 输入描述:

输入一个string型基因序列，和int型子串的长度

### 输出描述:

找出GC比例最高的子串,如果有多个输出第一个的子串

示例1

输入

```
AACTGTGCACGACCTGA
5
```

输出

```
GCACG
```

### 解决代码：

```
import java.io.*;

public class Main{
    public static void main(String[] args) throws NumberFormatException, IOException{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str;
        while((str = br.readLine()) != null){
            int maxRadio = 0;
            int index = 0;
            int n = Integer.parseInt(br.readLine());
            for(int i = 0;i <= str.length() - n;i++){
                int temp = getMaxRadio(str.substring(i,i+n));
                if(temp > maxRadio){
```

```
                    maxRadio = temp;
                    index = i;
                }
            }
            System.out.println(str.substring(index,index + n));
        }
    }

    public static int getMaxRadio(String str){
        int count = 0;
        for(int i = 0;i < str.length();i++){
            if('G' == str.charAt(i) || 'C' == str.charAt(i)){
                count++;
            }
        }
        return count;
    }

}
```

## 牛客网-华为机试练习题 65

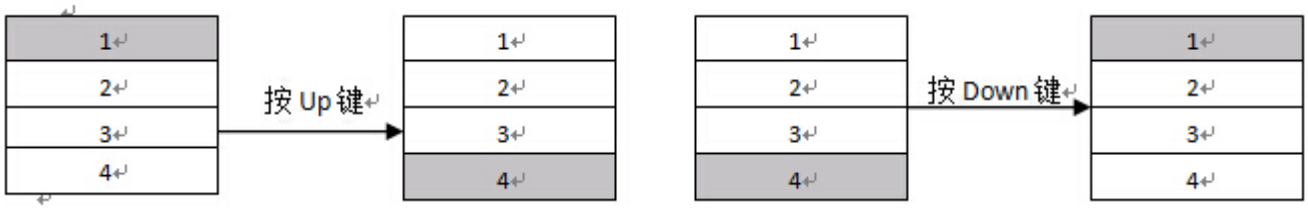
### 题目描述

MP3 Player因为屏幕较小，显示歌曲列表的时候每屏只能显示几首歌曲，用户要通过上下键才能浏览所有的歌曲。为了简化处理，假设每屏只能显示4首歌曲，光标初始的位置为第1首歌。

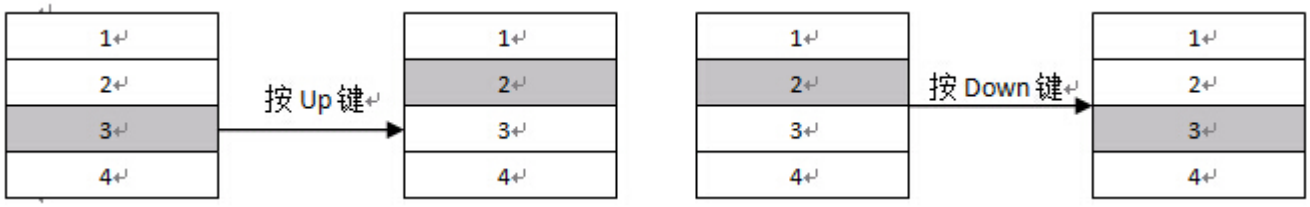
现在要实现通过上下键控制光标移动来浏览歌曲列表，控制逻辑如下：

1. 歌曲总数<=4的时候，不需要翻页，只是挪动光标位置。

光标在第一首歌曲上时，按Up键光标挪到最后一首歌曲；光标在最后一首歌曲时，按Down键光标挪到第一首歌曲。

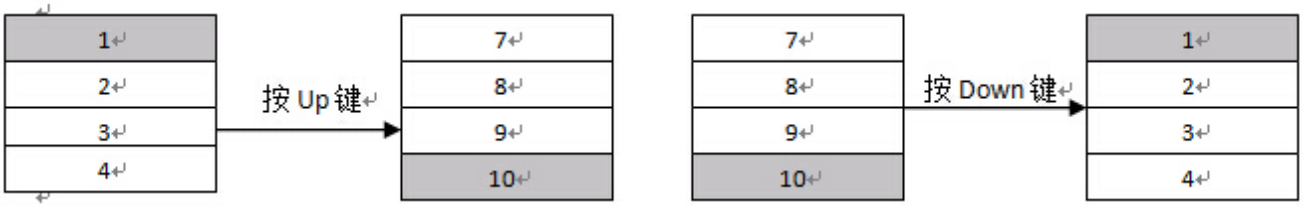



其他情况下用户按Up键，光标挪到上一首歌曲；用户按Down键，光标挪到下一首歌曲。

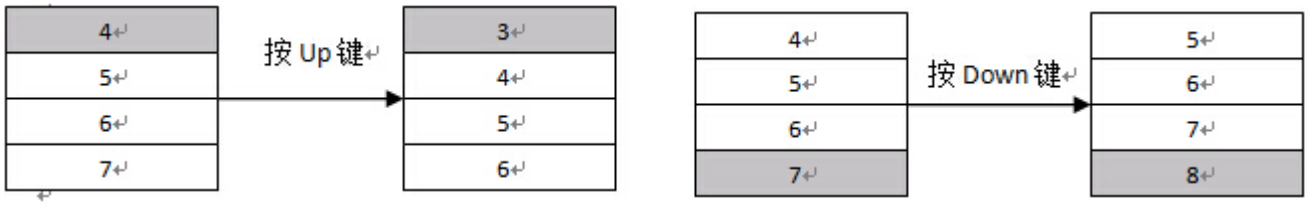



\2. 歌曲总数大于4的时候（以一共有10首歌为例）：

特殊翻页：屏幕显示的是第一页（即显示第1 – 4首）时，光标在第一首歌曲上，用户按Up键后，屏幕要显示最后一页（即显示第7-10首歌），同时光标放到最后一首歌上。同样的，屏幕显示最后一页时，光标在最后一首歌曲上，用户按Down键，屏幕要显示第一页，光标挪到第一首歌上。




一般翻页：屏幕显示的不是第一页时，光标在当前屏幕显示的第一首歌曲时，用户按Up键后，屏幕从当前歌曲的上一首开始显示，光标也挪到上一首歌曲。光标当前屏幕的最后一首歌时的Down键处理也类似。




其他情况，不用翻页，只是挪动光标就行。

## 输入描述:

输入说明：
1 输入歌曲数量
2 输入命令 U或者D

## 输出描述:

输出说明
1 输出当前列表
2 输出当前选中歌曲

示例1

输入

```
10
UUUU
```

输出

```
7 8 9 10
7
```

## 解决代码:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class Main{
    public static String[] music(String str,int n){
        int point=1;//光标
        int head=1;//屏幕的第一首歌
        for(int i=0;i<str.length();i++){
            char c=str.charAt(i);
            if(n<=4){
               if(c=='U'){
                  if(point==head) point=n;
                  else  point-=1;
```

```
            }
            if(c=='D'){
                if(point==head+n-1) point=1;
                else  point+=1;
            }
            continue;
        }
        if(c=='U'){//向上
            if(point==head){//需要向上翻页
                if(point==1){//特殊翻页
                    point=n;
                    head=n-3;
                }
                else{//普通翻页
                    point=point-1;
                    head=head-1;
                }
            }
            else{//不需要翻页
                point-=1;
            }
        }
        if(c=='D'){//向下
            if(point==head+3){//需要向下翻页
                if(point==n){//特殊翻页
                    point=1;
                    head=1;
                }
                else{//普通翻页
                    point+=1;
                    head+=1;
                }
            }
            else{//无需翻页
                point+=1;
            }
        }
    }
    String[] strary=new String[2];
    strary[0]=head+" "+(head+1)+" "+(head+2)+" "+(head+3);
    if(n<=4){
        strary[0]=head+"";
         for(int i=0;i<n-1;i++){
             strary[0]=strary[0]+" "+(head+i+1);
         }

    }
    strary[1]=point+"";
    return strary;
}
public static void main(String[] args)throws IOException{
    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
    String line="";
    while((line=br.readLine())!=null){
        int n=Integer.valueOf(line);
        String str=br.readLine();
        System.out.println(music(str,n)[0]);
        System.out.println(music(str,n)[1]);
    }
```

```
    }
}
```

## 牛客网-华为机试练习题 66

### 题目描述

查找两个字符串a,b中的最长公共子串。若有多个，输出在较短串中最先出现的那个。

### 输入描述:

输入两个字符串

### 输出描述:

返回重复出现的字符

示例1

输入

```
abcdefghijklmnop
abcsafjklmnopqrstuvw
```

输出

```
jklmnop
```

### 解决代码：

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class Main{
    public static String lcs(String str1,String str2){
        String strshort="";
        String strlong="";
        if(str1.length()<=str2.length()){
            strshort=str1;strlong=str2;
        }
        else{
            strshort=str2;strlong=str1;
        }
        int s1len=strshort.length();
        int s2len=strlong.length();
        int[][] dp=new int[s1len+1][s2len+1];
        for(int i=0;i<s1len+1;i++){
            for(int j=0;j<s2len+1;j++){
                if(i==0||j==0)  dp[i][j]=0;
                else{
                    if(strlong.charAt(j-1)==strshort.charAt(i-1))
                        dp[i][j]=dp[i-1][j-1]+1;
                    else
                        dp[i][j]=0;
                }
```

```
            }
        }
        int max=(int)(-Math.pow(2,31));
        int maxi=0;
        int maxj=0;
        for(int i=0;i<s1len+1;i++){
            for(int j=0;j<s2len+1;j++){
                if(dp[i][j]>max){
                    max=dp[i][j];
                    maxi=i-dp[i][j]+1;
                    maxj=i;
                }
            }
        }
        return strshort.substring(maxi-1,maxj);
    }
    public static void main(String[] args)throws IOException{
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String line="";
        while((line=br.readLine())!=null){
            String str1=line;
            String str2=br.readLine();
            System.out.println(lcs(str1,str2));
        }
    }
}
```

## 牛客网-华为机试练习题 67

### 题目描述

有6条配置命令，它们执行的结果分别是：

| 命 令 | 执 行 |
| --- | --- |
| **reset** | reset what |
| **reset board** | board fault |
| **board add** | where to add |
| **board delet** | no board at all |
| **reboot backplane** | impossible |
| **backplane abort** | install first |
| ***he he*** | unkown command |

**注意：he he不是命令。**

为了简化输入，方便用户，以“最短唯一匹配原则”匹配： 1、若只输入一字串，则只匹配一个关键字的命令行。例如输入：r，根据该规则，匹配命令reset，执行结果为：reset what；输入：res，根据该规则，匹配命令reset，执行结果为：reset what； 2、若只输入一字串，但本条命令有两个关键字，则匹配失败。例如输入：reb，可以找到命令reboot backpalne，但是该命令有两个关键词，所有匹配失败，执行结果为：unkown command 3、若输入两字串，则先匹配第一关键字，如果有匹配但不唯一，继续匹配第二关键字，如果仍不唯一，匹配失败。例如输入：r b，找到匹配命令reset board，执行结果为：board fault。

4、若输入两字串，则先匹配第一关键字，如果有匹配但不唯一，继续匹配第二关键字，如果唯一，匹配成功。例如输入：b a，无法确定是命令**board add**还是**backplane abort**，匹配失败。 5、若输入两字串，第一关键字匹配成功，则匹配第二关键字，若无匹配，失败。例如输入：bo a，确定是命令**board add**，匹配成功。 6、若匹配失败，打印“unkown command”

### 输入描述:

多行字符串，每行字符串一条命令

### 输出描述:

执行结果，每条命令输出一行

示例1

### 输入

```
reset
reset board
board add
board delet
reboot backplane
backplane abort
```

### 输出

```
reset what
board fault
where to add
no board at all
impossible
install first
```

### 解决代码：

```
import java.util.*;
import java.io.*;
public class Main{
    public static void main(String[] args)throws Exception{
        BufferedReader br =new BufferedReader(new InputStreamReader(System.in));
        HashMap<String, String> hMap = new HashMap<>();
         //HashMap<String, String> hMap = new HashMap<>();
            hMap.put("reset", "reset what");
            hMap.put("reset board", "board fault");
            hMap.put("board add", "where to add");
            hMap.put("board delet", "no board at all");
            hMap.put("reboot backplane", "impossible");
            hMap.put("backplane abort", "install first");
        String str =br.readLine();
        while(str!=null){

            if(hMap.containsKey(str)){
                System.out.println(hMap.get(str));
            }else{
                System.out.println("unkown command");
            }
            str = br.readLine();
        }
    }
}
```

## 牛客网-华为机试练习题 68

### 题目描述

问题描述：给出4个1-10的数字，通过加减乘除，得到数字为24就算胜利 输入： 4个1-10的数字。[数字允许重复，但每个数字仅允许使用一次，测试用例保证无异常数字] 输出： true or false

## 输入描述:

输入4个int整数

## 输出描述:

返回能否得到24点，能输出true，不能输出false

示例1

输入

```
7 2 1 10
```

输出

```
true
```

## 解决代码：

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = "";
        while ((line = br.readLine()) != null) {
            String[] strs = line.split(" ");
            List<Integer> list = new ArrayList<Integer>();
            for (int i = 0; i < 4; i++) {
                list.add(Integer.parseInt(strs[i]));
            }
            boolean flag = fun(list);
            System.out.println(flag);
        }
    }

    public static boolean fun(List<Integer> list) {
        for (int i = 0; i < list.size(); i++) {
            int temp = list.get(i);
            list.remove(i);
            if (getResult(list, temp)) {
                return true;
            }
            list.add(i, temp);
        }
        return false;
    }

    public static boolean getResult(List<Integer> list, int temp) {
        if (list.size() > 0) {
            for (int i = 0; i < list.size(); i++) {
                int n = list.get(i);
                list.remove(i);
```

```
                if (getResult(list, temp * n) || getResult(list, temp + n) || getResult(list,
temp - n)) {
                    return true;
                } else if (temp % n == 0) {
                    if (getResult(list, temp / n)) {
                        return true;
                    }
                }
                list.add(i, n);
            }
            return false;
        } else {
            if (temp == 24) {
                return true;
            } else {
                return false;
            }
        }
    }
}
```

## 牛客网-华为机试练习题 69

### 题目描述

查找和排序

题目：输入任意（用户，成绩）序列，可以获得成绩从高到低或从低到高的排列,相同成绩 都按先录入排列在前的规则处理。

例示： jack 70 peter 96 Tom 70 smith 67

从高到低 成绩
peter 96
jack 70
Tom 70
smith 67

从低到高

smith 67

Tom 70
jack 70
peter 96

### 输入描述:

输入多行，先输入要排序的人的个数，然后分别输入他们的名字和成绩，以一个空格隔开

### 输出描述:

按照指定方式输出名字和成绩，名字和成绩之间以一个空格隔开

输入

3
0

```
fang 90
yang 50
ning 70
```

输出

```
fang 90
ning 70
yang 50
```

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main{

    public static void main(String[] args){
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        try {
            String str;
            while((str=reader.readLine())!=null){
                //获取要排序的人的个数
                int count = Integer.parseInt(str);
                //获取输入的排序方法(升序还是降序)
                int sortType = Integer.parseInt(reader.readLine());

                String[] users = new String[count];
                int[] scores = new int[count];
                for(int i=0;i<count;i++){
                    String line = reader.readLine();
                    String[] parts = line.split(" ");
                    String user = parts[0];
                    int score = Integer.parseInt(parts[1]);
                    if(sortType==0){
                        int j = 0;
                        for(j=i-1;j>=0;j--){
                            if(scores[j]<score){
                                scores[j+1] = scores[j];
                                users[j+1] = users[j];
                            }
                            else{
                                break;
                            }
                        }
                        scores[j+1] = score;
                        users[j+1] = user;
                    }
                    else{
                        int j = 0;
                        for(j=i-1;j>=0;j--){
                            if(scores[j]>score){
                                scores[j+1] = scores[j];
                                users[j+1] = users[j];
                            }
                            else{
                                break;
                            }
```

```
                        }
                        scores[j+1] = score;
                        users[j+1] = user;
                    }
                }
                for(int i=0;i<count;i++){
                    System.out.println(users[i]+" "+scores[i]);
                }
            }
        } catch(IOException e){

        }
    }

}
```

# 牛客网-华为机试练习题 70

## 题目描述

- 如果A是个x行y列的矩阵，B是个y行z列的矩阵，把A和B相乘，其结果将是另一个x行z列的矩阵C。这个矩阵的每个元素是由下面的公式决定的：

原型：

voidmatrix_multiply(int *m1,int *m2,int *r, int x, int y, int z);

输入参数：

int *m1：x行y列的矩阵(array1[x][y])

int *m2：y行z列的矩阵(array2[y][z])

int x：矩阵m1的行数

int y：矩阵m1的列数/矩阵m2的行数

int z：矩阵m2的列数

输出参数：

int *r：矩阵m1, m2相乘的结果(array3[x][z])

返回值：

void

## 输入描述:

输入说明：
1、第一个矩阵的行数
2、第一个矩阵的列数和第二个矩阵的行数
3、第二个矩阵的列数
4、第一个矩阵的值
5、第二个矩阵的值

## 输出描述:

输出两个矩阵相乘的结果

示例1

输入

```
2
2
2
3 8
8 0
9 0
18 9
```

输出

```
171 72
72 0
```

## 解决代码:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNext()) {
            int r1, c1, r2, c2;
            r1 = Integer.parseInt(sc.nextLine());
            c1 = Integer.parseInt(sc.nextLine());
            r2 = c1;
            c2 = Integer.parseInt(sc.nextLine());

            int[][] x = new int[r1][c1];
            int[][] y = new int[r2][c2];

            for (int i = 0; i < r1; i++) {
                String[] str = sc.nextLine().split(" ");
                for (int j = 0; j < str.length; j++) {
                    x[i][j] = Integer.parseInt(str[j]);
                }
            }

            for (int i = 0; i < r2; i++) {
                String[] str = sc.nextLine().split(" ");
                for (int j = 0; j < str.length; j++) {
                    y[i][j] = Integer.parseInt(str[j]);
                }
            }
            print2DArray(matrixMultiplication(x, y));
        }
        sc.close();
    }

    private static int[][] matrixMultiplication(int[][] x, int[][] y) {
        int r1 = x.length, c1 = x[0].length, r2 = y.length, c2 = y[0].length;
        int[][] result = new int[r1][c2];
        for (int i = 0; i < r1; i++) {
```

```
            for (int j = 0; j < c2; j++) {
                result[i][j] = getResult(x, i, y, j);
            }
        }
        return result;
    }

    private static int getResult(int[][] x, int r, int[][] y, int c) {
        int r1 = x.length, c1 = x[0].length, r2 = y.length, c2 = y[0].length;
        int[] a = new int[c1];
        int[] b = new int[r2];

        for (int i = 0; i < c1; i++) {
            a[i] = x[r][i];
            b[i] = y[i][c];
        }
        int sum = 0;
        for (int i = 0; i < c1; i++) {
            sum = sum + a[i] * b[i];
        }
        return sum;
    }

       public static void print2DArray(int[][] a) {
            StringBuffer sb = new StringBuffer();
            for (int i = 0; i < a.length; i++) {
                for (int j = 0; j < a[i].length - 1; j++) {
                    sb.append(a[i][j]).append(" ");
                }
                sb.append(a[i][a[i].length - 1]).append("\n");
            }
            System.out.println(sb.toString().substring(0, sb.length() - 1));
        }

    }
```

# 牛客网-华为机试练习题 71

## 题目描述

矩阵乘法的运算量与矩阵乘法的顺序强相关。

例如：

A是一个50×10的矩阵，B是10×20的矩阵，C是20×5的矩阵

计算A*B*C有两种顺序：（（AB）C）或者（A（BC））），前者需要计算15000次乘法，后者只需要3500次。

编写程序计算不同的计算顺序需要进行的乘法次数

## 输入描述:

输入多行，先输入要计算乘法的矩阵个数n，每个矩阵的行数，列数，总共2n的数，最后输入要计算的法则

## 输出描述:

输出需要进行的乘法次数

示例1
输入

```
3
50 10
10 20
20 5
(A(BC))
```

输出

```
3500
```

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class Main {
    public static int getNo(String str,int[][] data) {
        int total=0;
        //读右括号
        for(int i=0;i<str.length();i++) {
            char c=str.charAt(i);
            if(c==')') {
                for(int j=i-1;j>=0;j--) {
                    char c1=str.charAt(j);
                    if(c1=='(') {
                        while(str.charAt(j+2)!=')') {
                            char c2=str.charAt(j+1);
                            char c3=str.charAt(j+2);
                            total+=data[c2-65][0]*data[c2-65][1]*data[c3-65][1];
                            //需要改变数组，改变字符串
                            data[c2-65][1]=data[c3-65][1];
                            str=str.substring(0,j+2)+str.substring(j+3);
                        }
                        //i位置的）与j为止的（要删除
                        //str已经变短，j+1不见了，i--
                        i--;
                        str=(j==0?"":str.substring(0,j))+str.substring(j+1,i)+
(i==str.length()-1?"":str.substring(i+1));
                        //i指向原来左括号的位置
                        i=j;
                        break;
                    }
                }
            }
        }
        return total;
    }
    public static void main(String[] args)throws IOException{
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String line="";
        while((line=br.readLine())!=null) {
            int n=Integer.valueOf(line);
            int[][] data=new int[n][2];
            for(int i=0;i<n;i++) {
                line=br.readLine();
```

```
                int x=Integer.valueOf(line.substring(0, line.lastIndexOf(" ")));
                int y=Integer.valueOf(line.substring(line.lastIndexOf(" ")+1));
                data[i][0]=x;
                data[i][1]=y;
            }
            line=br.readLine();
            System.out.println(getNo(line, data));
        }
    }
}
```

## 牛客网-华为机试练习题 72

### 题目描述

问题描述：在计算机中，通配符一种特殊语法，广泛应用于文件搜索、数据库、正则表达式等领域。现要求各位实现字符串通配符的算法。 要求： 实现如下2个通配符： *：匹配0个或以上的字符（字符由英文字母和数字0-9组成，不区分大小写。下同） ？：匹配1个字符

输入： 通配符表达式； 一组字符串。

输出： 返回匹配的结果，正确输出true，错误输出false

### 输入描述:

先输入一个带有通配符的字符串，再输入一个需要匹配的字符串

### 输出描述:

返回匹配的结果，正确输出true，错误输出false

示例1

输入

```
te?t*.*
txt12.xls
```

输出

```
false
```

### 解决代码：

```
/*问题描述：在计算机中，通配符一种特殊语法，广泛应用于文件搜索、数据库、正则表达式等领域。现要求各位实现字符串通配符的算法。
要求：
实现如下2个通配符：
*：匹配0个或以上的字符（字符由英文字母和数字0-9组成，不区分大小写。下同）
？：匹配1个字符
输入：
通配符表达式；
一组字符串。
输出：
返回匹配的结果，正确输出true，错误输出false

te?t*.*
txt12.xls
```

```
false
*/
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
public class Main {
    public static boolean isMatch(String str1,String str2) {
        str1=str1.replaceAll("\\?", "[0-9a-zA-Z]");//把str1变成正则表达式
        str1=str1.replaceAll("\\*","[0-9a-zA-Z]*");
        str1=str1.replaceAll("\\.", "\\.");
    //  str1=str1.replaceAll("\\","\\");      //  \在java里表示转义，
    //  str1=str1.replaceAll("\\+","\\+");
        if(str2.matches(str1))  return true;
        return false;
    }
    public static void main(String[] args)throws IOException{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line="";
        while((line=br.readLine())!=null) {
            String str1=line;
            String str2=br.readLine();
            System.out.println(isMatch(str1, str2));
        }
    }
}
```

# 牛客网-华为机试练习题 73

## 题目描述

公元前五世纪，我国古代数学家张丘建在《算经》一书中提出了“百鸡问题”：鸡翁一值钱五，鸡母一值钱三，鸡雏三值钱一。百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？

详细描述：

接口说明

原型：

int GetResult(vector &list)

输入参数：

无

输出参数（指针指向的内存区域保证有效）：

list 鸡翁、鸡母、鸡雏组合的列表

返回值：

-1 失败

0 成功

## 输入描述:

输入任何一个整数，即可运行程序。

### 输出描述:

示例1

输入

```
1
```

输出

```
0 25 75
4 18 78
8 11 81
12 4 84
```

### 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line ="";
        while((line=br.readLine())!=null){
            int i;
            int j;
            for(i=0;i<=20;++i)
            {
                for(j=0;j<=100-i;++j)
                {
                    int k = 100-i-j;
                    if((k%3 ==0) && (5*i+3*j+k/3==100))
                        System.out.println(i+" "+j+" "+k);
                }
            }


        }
    }

}
```

## 牛客网-华为机试练习题 74

### 题目描述

根据输入的日期，计算是这一年的第几天。。

详细描述：

输入某年某月某日，判断这一天是这一年的第几天？。

接口设计及说明：

```
 /*****************************************************************************
 Description    : 数据转换
```

```
Input Param   : year 输入年份
                Month 输入月份
                Day 输入天

Output Param  :
Return Value  : 成功返回0，失败返回-1（如：数据错误）
*****************************************************************************/
public static int iConverDateToDay(int year, int month, int day)
{
    /* 在这里实现功能，将结果填入输入数组中*/
    return 0;
}

/*****************************************************************************
Description   :
Input Param   :

Output Param  :
Return Value  : 成功:返回outDay输出计算后的第几天;
                                                失败:返回-1
*****************************************************************************/
public static int getOutDay()
{
 return 0;
}
```

## 输入描述:

输入三行，分别是年，月，日

## 输出描述:

成功:返回outDay输出计算后的第几天;
失败:返回-1

示例1

输入

2012

12

31

输出

366

## 解决代码：

```
//
//输入年月日，返回天数
//1-31 2-28 3-31 4-30 5-31 6-30 7-31 8-31 9-30 10-31 11-30 12-31
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
```

```
public class Main {
    public static int getAllDay(int year,int month,int day) {
        int[] dayary= {31,28,31,30,31,30,31,31,30,31,30,31};
        //判断闰年
        if(year%4==0) dayary[1]=29;
        int total=0;
        for(int i=0;i<month-1;i++) {
            total+=dayary[i];
        }
        total+=day;
        return total;
    }
    public static void main(String[] args)throws IOException{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line="";
        while((line=br.readLine())!=null) {
            String[] lineary=line.split(" ");
            int year=Integer.valueOf(lineary[0]);
            int month=Integer.valueOf(lineary[1]);
            int day=Integer.valueOf(lineary[2]);
            System.out.println(getAllDay(year, month, day));
        }

    }
}
```

# 牛客网-华为机试练习题 75

## 题目描述

在命令行输入如下命令：

xcopy /s c:\ d:\,

各个参数如下：

参数1：命令字xcopy

参数2：字符串/s

参数3：字符串c:\

参数4: 字符串d:\

请编写一个参数解析程序，实现将命令行各个参数解析出来。

解析规则：

1.参数分隔符为空格
2.对于用“”包含起来的参数，如果中间有空格，不能解析为多个参数。比如在命令行输入xcopy /s “C:\program files” “d:\”时，参数仍然是4个，第3个参数应该是字符串C:\program files，而不是C:\program，注意输出参数时，需要将“”去掉，引号不存在嵌套情况。
3.参数不定长
4.输入由用例保证，不会出现不符合要求的输入

## 输入描述:

输入一行字符串，可以有空格

## 输出描述:

输出参数个数，分解后的参数，每个参数都独占一行

示例1

输入

```
xcopy /s c:\\ d:\\
```

输出

```
4
xcopy
/s
c:\\
d:\\
```

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String s = br.readLine();

        String[] strings = s.split(" ");
        List<String> list = new ArrayList<String>();

        for(int i=0;i<strings.length;) {
            if(strings[i].charAt(0)!='"') {
                list.add(strings[i]);
                ++i;
            }

            else {
                if(strings[i].charAt(strings[i].length()-1)=='"') {
                    list.add(strings[i].substring(1, strings[i].length()-1));
                    ++i;
                }

                else {
                    StringBuilder sb = new StringBuilder();
                    sb.append(strings[i].substring(1)+" ");
                    ++i;
                    while(strings[i].charAt(strings[i].length()-1)!='"') {
                        sb.append(strings[i]+" ");
                        ++i;
                    }

           //           if(i != strings.length){
                        sb.append(strings[i].substring(0, strings[i].length()-1));
                        ++i;
```

```
                //         }
                     list.add(sb.toString());
                 }
             }
         }

         System.out.println(list.size());
         Iterator it = list.iterator();
         while(it.hasNext()) {
             System.out.println(it.next());
         }


    }
}
```

## 牛客网-华为机试练习题 76

### 题目描述

题目标题：

计算两个字符串的最大公共字串的长度，字符不区分大小写

详细描述：

接口说明

原型：

```
int getCommonStrLength(char * pFirstStr, char * pSecondStr);
```

输入参数：

```
     char * pFirstStr //第一个字符串

     char * pSecondStr//第二个字符串
```

### 输入描述:

输入两个字符串

### 输出描述:

输出一个整数

示例1

输入

```
asdfas werasdfaswer
```

输出

```
6
```

### 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
```

```
import java.io.InputStreamReader;


public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        String string1 = "";
        while ((string1 = bufferedReader.readLine()) != null) {
            String string2 = bufferedReader.readLine();
            char[] str1 = string1.toCharArray();
            char[] str2 = string2.toCharArray();
            int[][] dp = new int[str1.length][str2.length];
            for(int i = 0;i < str1.length;i++){
                if(str1[i] == str2[0]){
                    dp[i][0] = 1;
                }
            }
            for(int j = 1;j < str2.length;j++){
                if(str2[j] == str1[0]){
                    dp[0][j] = 1;
                }
            }
            int max = 0;
            for(int i = 1;i < str1.length;i++){
                for(int j = 1;j < str2.length;j++){
                    if(str1[i] == str2[j]){
                        dp[i][j] = dp[i-1][j-1] + 1;
                    }
                    max= Math.max(max,dp[i][j]);
                }
            }
            System.out.println(max);
        }
    }

}
```

# 牛客网-华为机试练习题 77

## 题目描述

验证尼科彻斯定理，即：任何一个整数m的立方都可以写成m个连续奇数之和。

例如：

1^3=1

2^3=3+5

3^3=7+9+11

4^3=13+15+17+19

接口说明

原型：

```
 /*
 功能: 验证尼科彻斯定理，即：任何一个整数m的立方都可以写成m个连续奇数之和。
```

```
原型：
    int GetSequeOddNum(int m,char * pcSequeOddNum);
输入参数：
    int m：整数(取值范围：1～100)

返回值：
    m个连续奇数(格式："7+9+11");
*/

public String GetSequeOddNum(int m)
{
    /*在这里实现功能*/

    return null;
}
```

## 输入描述:

输入一个int整数

## 输出描述:

输出分解后的string

示例1
输入

6
输出

31+33+35+37+39+41

## 解决代码:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = "";
        while ((line = br.readLine()) != null) {
            int n = Integer.parseInt(line);
            getRes(n);
        }
    }

    public static void getRes(int n) {
        String str = "";
        if (n % 2 == 0) { // n为偶数
            int mid = (int) (Math.pow(n, 3) / n);
            str = (mid - 1) + "+" + (mid + 1);
            int T = n / 2;
            for (int i = 1; i < T; i++) {
                str = (mid - (2 * i) - 1) + "+" + str + "+" + (mid + (2 * i) + 1);
            }
        } else { // n为奇数
            int mid = (int) (Math.pow(n, 3) / n);
```

```
            str = mid + "";
            int T = n / 2;
            for (int i = 0; i < T; i++) {
                str = (mid - 2 * (i + 1)) + "+" + str + "+" + (mid + 2 * (i + 1));
            }
        }
        System.out.println(str);
    }
}
```

## 牛客网-华为机试练习题 78

### 题目描述

给定一个正整数N代表火车数量，0<N<10，接下来输入火车入站的序列，一共N辆火车，每辆火车以数字1-9编号。要求以字典序排序输出火车出站的序列号。

### 输入描述:

有多组测试用例，每一组第一行输入一个正整数N（0<N<10），第二行包括N个正整数，范围为1到9。

### 输出描述:

输出以字典序从小到大排序的火车出站序列号，每个编号以空格隔开，每个输出序列换行，具体见sample。

示例1

输入

```
3
1 2 3
```

输出

```
1 2 3
1 3 2
2 1 3
2 3 1
3 2 1
```

### 解决代码：

```
import java.util.*;

public class Main{
    private static Stack<String> stack1=new Stack<String>();
    private static Stack<String> stack2=new Stack<String>();
    private static List<String> list=new ArrayList<String>();

    public static void ff(String str){
        if(stack1.isEmpty()&&stack2.isEmpty()){
            list.add(str.trim());
            return;
        }
        if(!stack2.isEmpty()){
            String str1=stack2.pop();
            ff(str+" "+str1);
            stack2.push(str1);
        }
        if(!stack1.isEmpty()){
            String str2=stack1.pop();
```

```
            stack2.push(str2);
            ff(str);
            stack2.pop();
            stack1.push(str2);
        }
    }

    public static void main(String[] args){
        Scanner scanner=new Scanner(System.in);
        while(scanner.hasNext()){
            int n=scanner.nextInt();
            scanner.nextLine();
            String str=scanner.nextLine();
            String[] ss=str.split(" ");
            for(int i=ss.length-1;i>=0;i--)
                stack1.push(ss[i]);
            ff("");
            Collections.sort(list);
            for(String s:list)
                System.out.println(s);
        }

    }
}
```

# 牛客网-华为机试练习题 79

## 题目描述

请设计一个算法完成两个超长正整数的加法。

接口说明

```
/*
请设计一个算法完成两个超长正整数的加法。
输入参数：
String addend：加数
String augend：被加数
返回值：加法结果
*/

public String AddLongInteger(String addend, String augend)
{
    /*在这里实现功能*/


 return null;
}
```

## 输入描述:

输入两个字符串数字

## 输出描述:

输出相加后的结果，string型

示例1

输入

```
99999999999999999999999999999999999999999999999999
1
```

输出

```
100000000000000000000000000000000000000000000000000
```

### 解决代码：

```
import java.util.*;
import java.io.*;

public class Main{
    public static void main(String[] args) throws IOException{
        BufferedReader bReader = new BufferedReader(new InputStreamReader(System.in));
        String str1;
        while((str1=bReader.readLine())!=null) {
            String str2=bReader.readLine();
            StringBuilder sb1=new StringBuilder(str1);
            StringBuilder sb2=new StringBuilder(str2);
            int len1=sb1.length();
            int len2=sb2.length();
            if (len1>len2){
                for (int i=0;i<len1-len2;i++)
                    sb2.insert(0,"0");
            } else {
                for (int i=0;i<len2-len1;i++)
                    sb1.insert(0,"0");
            }
            StringBuilder sb=new StringBuilder();
            int temp=0;
            for (int i=sb1.length()-1;i>=0;i--){
                int m=sb1.charAt(i)-'0';
                int n=sb2.charAt(i)-'0';
                int sum=m+n+temp;
                sb.insert(0,sum%10);
                temp=sum/10;
            }
            if (temp!=0)
                sb.insert(0,temp);
            System.out.println(sb.toString());

        }
    }

}
```

## 牛客网-华为机试练习题 80

### 题目描述

对于不同的字符串，我们希望能有办法判断相似程度，我们定义了一套操作方法来把两个不相同的字符串变得相同，具体的操作方法如下：

1 修改一个字符，如把“a”替换为“b”。

2 增加一个字符，如把“abdd”变为“aebdd”。

3 删除一个字符，如把“travelling”变为“traveling”。

比如，对于“abcdefg”和“abcdef”两个字符串来说，我们认为可以通过增加和减少一个“g”的方式来达到目的。上面的两种方案，都只需要一次操作。把这个操作所需要的次数定义为两个字符串的距离，而相似度等于“距离 + 1”的倒数。也就是说，“abcdefg”和“abcdef”的距离为1，相似度为1/2=0.5.

给定任意两个字符串，你是否能写出一个算法来计算出它们的相似度呢？

请实现如下接口

```
 /* 功能：计算字符串的相似度
  \* 输入：pucAExpression/ pucBExpression：字符串格式，如: "abcdef"
  \* 返回：字符串的相似度,相似度等于“距离 + 1”的倒数,结果请用1/字符串的形式,如1/2
  */
 public static  String  calculateStringDistance(String expressionA, String expressionB)
 {
     /* 请实现*/
     return null;
 }
```

约束：

1、PucAExpression/ PucBExpression字符串中的有效字符包括26个小写字母。

2、PucAExpression/ PucBExpression算术表达式的有效性由调用者保证；
3、超过result范围导致信息无法正确表达的，返回null。

## 输入描述:

输入两个字符串

## 输出描述:

输出相似度，string类型

示例1

输入

```
abcdef
abcdefg
```

输出

```
1/2
```

## 解决代码：

```
import java.util.*;
import java.io.*;

public class Main {
    public static void main(String[] args) throws IOException {
                BufferedReader in = new BufferedReader(
                    new InputStreamReader(System.in)
```

```
                );
                String s1 = "";
                while ( null != (s1 = in.readLine())  ){
                //将两个输入字符串转为数组
                String s2=in.readLine();
                char[] cs1=s1.toCharArray();
                char[] cs2=s2.toCharArray();
                int[][] dp=new int[s1.length()+1][s2.length()+1];
                //用动态规划的方式获取一个数组变为另一个数组的步骤次数
                //初始化二维数组
                for(int row=1;row<=s1.length();row++){
                    dp[row][0]=row;
                }
                for(int col=1;col<=s2.length();col++){
                    dp[0][col]=col;
                }
                //动态规划
                for(int row=1;row<=s1.length();row++){
                    for(int col=1;col<=s2.length();col++){
                        if(cs1[row-1]==cs2[col-1]){
                            dp[row][col]=dp[row-1][col-1];
                        }
                        else{
                            int min1=Math.min(dp[row-1][col],dp[row][col-1])+1;
                            dp[row][col]=Math.min(min1,dp[row-1][col-1]+1);

                        }
                    }
                }
                System.out.println("1/"+ (dp[s1.length()][s2.length()]+1) );

            }
        }
    }
```

## 牛客网-华为机试练习题 81

### 题目描述

题目标题：

将两个整型数组按照升序合并，并且过滤掉重复数组元素[注：题目更新了。输出之后有换行]

详细描述：

接口说明

原型：

```
voidCombineBySort(int* pArray1,intiArray1Num,int* pArray2,intiArray2Num,int*
pOutputArray,int* iOutputNum);
```

输入参数：

int* pArray1 ：整型数组1

intiArray1Num：数组1元素个数

int* pArray2 ：整型数组2

intiArray2Num：数组2元素个数

输出参数（指针指向的内存区域保证有效）：

int* pOutputArray：合并后的数组

int* iOutputNum：合并后数组元素个数

返回值：

void

## 输入描述:

输入说明，按下列顺序输入：
1 输入第一个数组的个数
2 输入第一个数组的数值
3 输入第二个数组的个数
4 输入第二个数组的数值

## 输出描述:

输出合并之后的数组

示例1

输入

```
3
1 2 5
4
-1 0 3 2
```

输出

```
-101235
```

## 解决代码：

```
import java.util.Arrays;
import java.util.Iterator;
import java.util.Scanner;
import java.util.TreeSet;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner in = new Scanner(System.in);
        while (in.hasNext()) {
            int numa = in.nextInt();
            int[] a = new int[numa];
            for (int i = 0; i < numa; i++) {
                a[i] = in.nextInt();
            }
            int numb = in.nextInt();
            int[] b = new int[numb];
            for (int i = 0; i < numb; i++) {
```

```
                b[i] = in.nextInt();
            }
            System.out.print(combineBySort(a, b));
        }
        in.close();
    }
    public static String combineBySort(int[] a, int[] b) {
        int[] c = new int[a.length + b.length];
        Arrays.sort(a);
        Arrays.sort(b);

        int aindex = 0;
        int bindex = 0;
        int cindex = 0;

        while (aindex < a.length && bindex < b.length) {
            if (a[aindex] < b[bindex]) {
                c[cindex] = a[aindex];
                aindex++;
            }
            else if (a[aindex] > b[bindex]) {
                c[cindex] = b[bindex];
                bindex++;
            }
            else {
                c[cindex] = a[aindex];
                aindex++;
                bindex++;
            }
            cindex++;
        }

        for (int i = aindex; i < a.length; i++) {
            if (a[i] != c[cindex - 1]) {
                c[cindex] = a[i];
                cindex++;
            }
        }

        for (int i = bindex; i < b.length; i++) {
            if (b[i] != c[cindex - 1]) {
                c[cindex] = b[i];
                cindex++;
            }
        }

        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < cindex; i++) {
            builder.append(c[i]);
        }
        return builder.toString();
    }

}
```

## 牛客网-华为机试练习题 82

### 题目描述

题目标题：

判断短字符串中的所有字符是否在长字符串中全部出现

详细描述：

接口说明

原型：

`boolIsAllCharExist(char* pShortString,char* pLongString);`

输入参数：

- `char* pShortString`：短字符串
- `char* pLongString`：长字符串

## 输入描述:

输入两个字符串。第一个为短字符，第二个为长字符。

## 输出描述:

返回值：

示例1

输入

bc
abc
输出

true

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str = "";
        while ((str = br.readLine()) != null) {
            String str2 = br.readLine();
            int i;
            for (i = 0; i < str.length(); i++) {
                if (str2.indexOf(str.charAt(i)) == -1) {
                    System.out.println("false");
                    i--;
                    break;
                }
            }
            if (i == str.length()) {
                System.out.println(true);
```

```
            }
        }
        br.close();
    }
}
```

# 牛客网-华为机试练习题 83

## 题目描述

分子为1的分数称为埃及分数。现输入一个真分数(分子比分母小的分数，叫做真分数)，请将该分数分解为埃及分数。如：8/11 = 1/2+1/5+1/55+1/110。

接口说明

```
/*
功能: 将分数分解为埃及分数序列
输入参数：
    String pcRealFraction:真分数(格式“8/11”)
返回值：
    String pcEgpytFraction:分解后的埃及分数序列(格式“1/2+1/5+1/55+1/100”)
*/

public static String  ConvertRealFractToEgpytFract(String pcRealFraction)
{
 return null;
}
```

## 输入描述:

输入一个真分数，String型

## 输出描述:

输出分解后的string

示例1

输入

```
8/11
```

输出

```
1/2+1/5+1/55+1/110
```

## 解决代码：

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
```

```
        // TODO Auto-generated method stub
        Scanner in=new Scanner(System.in);
        while(in.hasNext()){
        String str=in.next();
        Egyptsocre(str);
        }

        }
        public static void Egyptsocre(String str){
        String[] parts=str.split("/");
        int a=Integer.parseInt(parts[0]);//分子
        int b=Integer.parseInt(parts[1]);//分母
        StringBuilder sb=new StringBuilder();
        int c;
                while (a != 1) {
                    if (b % (a - 1) == 0) {
                        sb.append("1/").append(b / (a - 1)).append('+');
                        a = 1;
                    } else {
                        c = b / a + 1; //重要点
                        sb.append("1/").append(c).append('+');
                        a = a * c - b;
                        b = c * b;           //计算分式子
                        if (b % a == 0) {
                            b = b / a;
                            a = 1;
                        }
                    }
                }
                sb.append("1/").append(b);
                System.out.println(sb.toString());
            }

}
```

# 牛客网-华为机试练习题 84

## 题目描述

有一个数据表格为二维数组（数组元素为int类型），行长度为ROW_LENGTH,列长度为COLUMN_LENGTH。对该表格中数据的操作可以在单个单元内，也可以对一个整行或整列进行操作，操作包括交换两个单元中的数据；插入某些行或列。

请编写程序，实现对表格的各种操作，并跟踪表格中数据在进行各种操作时，初始数据在表格中位置的变化轨迹。

详细要求:

1.数据表规格的表示方式为“行*列”, 数据表元素的位置表示方式为[行,列]，行列均从0开始编号

2.数据表的最大规格为9行*9列，对表格进行操作时遇到超出规格应该返回错误

3.插入操作时，对m*n表格，插入行号只允许0~m，插入列号只允许0~n。超出范围应该返回错误

4.只需记录初始表格中数据的变化轨迹，查询超出初始表格的数据应返回错误

例如: 初始表格为4*4，*可查询的元素范围为[0,0]~[3,3]，假设插入了第2行，数组变为5*4，查询元素[4,0]时应该返回错误

5.查询数据要求返回一个链表，链表中节点的顺序即为该查询的数据在表格中的位置变化顺序（需包含初始位置）

## 输入描述:

输入数据按下列顺序输入：
1 表格的行列值
2 要交换的两个单元格的行列值
3 输入要插入的行的数值
4 输入要插入的列的数值
5 输入要获取运动轨迹的单元格的值

## 输出描述:

输出按下列顺序输出：
1 初始化表格是否成功，若成功则返回0， 否则返回-1
2 输出交换单元格是否成功
3 输出插入行是否成功
4 输出插入列是否成功
5 输出要查询的运动轨迹的单元查询是否成功

示例1

输入

```
3 4
1 1
0 1
2
1
2 2
```

输出

```
0
0
0
0
0
```

## 解决代码：

```
import java.util.Scanner;

/**
 * Created by dy on 2016/7/6.
 */
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext()) {
            int[] arr = new int[10];
            for (int i = 0; i < 10; i++) {
                arr[i] = scanner.nextInt();
            }
            System.out.print(solve(arr));
        }
        scanner.close();
    }

    private static String solve(int[] arr) {
        int[] result = new int[5];
```

```
        // 检查行列值
        if (arr[0] < 0 || arr[0] > 9 || arr[1] < 0 || arr[1] > 9) {
            result[0] = -1;
        } else {
            result[0] = 0;
        }
// 检查交换单元格是否合法
        if (result[0] == 0 && (arr[2] >= 0 && arr[2] < arr[0] && arr[3] >= 0 && arr[3] <
arr[1])
                && (arr[4] >= 0 && arr[4] < arr[0] && arr[5] >= 0 && arr[5] < arr[1])) {
            result[1] = 0;
        } else {
            result[1] = -1;
        }
// 检查插入行是否成功
        if (result[0] == 0 && (arr[6] >= 0 && arr[6] < arr[0])) {
            result[2] = 0;
        } else {
            result[2] = -1;
        }
// 检查插入列是否成功
        if (result[0] == 0 && (arr[7] >= 0 && arr[7] < arr[1])) {
            result[3] = 0;
        } else {
            result[3] = -1;
        }
// 检查访问是否成功
        if (result[0] == 0 && (arr[8] >= 0 && arr[8] < arr[0] && arr[9] >= 0 && arr[9] <
arr[1])) {
            result[4] = 0;
        } else {
            result[4] = -1;
        }
        StringBuilder b = new StringBuilder();
        for (int i : result) {
            b.append(i).append('\n');
        }
        return b.toString();
    }
}
```

## 牛客网-华为机试练习题 85

### 题目描述

找出给定字符串中大写字符(即'A'-'Z')的个数

接口说明

原型：int CalcCapital(String str);

返回值：int

### 输入描述:

输入一个String数据

## 输出描述:

输出string中大写字母的个数

示例1

输入

add123#$%#%#O
输出

1

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

/**
 * 题目描述
找出给定字符串中大写字符(即'A'-'Z')的个数
接口说明
    原型：int CalcCapital(String str);
    返回值：int


输入描述:
输入一个String数据
输出描述:
输出string中大写字母的个数
示例1
输入

add123#$%#%#O
输出

1
 * @author Administrator
 *
 */
public class Main {

    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = "";
        while((line = br.readLine())!=null)
        {
            System.out.println(CalcCapital(line));

        }

    }

    private static int CalcCapital(String line) {

        int sum = 0;
        for(int i=0;i<line.length();++i)
```

```
            if(line.charAt(i)>='A' && line.charAt(i)<='Z')
                ++sum;

        return sum;
    }

}
```

# 牛客网-华为机试练习题 86

## 题目描述

Catcher 是MCA国的情报员，他工作时发现敌国会用一些对称的密码进行通信，比如像这些ABBA，ABA，A，123321，但是他们有时会在开始或结束时加入一些无关的字符以防止别国破解。比如进行下列变化 ABBA->12ABBA,ABA->ABAKK,123321->51233214 。因为截获的串太长了，而且存在多种可能的情况（abaaab可看作是aba,或baaab的加密形式），Cathcer的工作量实在是太大了，他只能向电脑高手求助，你能帮Catcher找出最长的有效密码串吗？

（注意：记得加上while处理多个测试用例）

## 输入描述:

输入一个字符串

## 输出描述:

返回有效密码串的最大长度

示例1

输入

ABBA

输出

4

## 解决代码：

```
import java.io.*;

public class Main
{
  public static int getLongestStr(String str)
  {
    int maxLen = -1;
    //奇数
    for(int i=0;i<str.length();++i)
    {
      int j=i-1;
      int k=i+1;
      while(j>=0&&k<str.length()&&str.charAt(j)==str.charAt(k))
      {
        if((k-j+1)>maxLen)  maxLen = k-j+1;
        j--;
        k++;
      }
    }
```

```
     for(int i=0;i<str.length();++i)
     {
        int j=i;
        int k=i+1;
        while(j>=0&&k<str.length()&&str.charAt(j)==str.charAt(k))
        {
          if((k-j+1)>maxLen)  maxLen = k-j+1;
          j--;
          k++;
        }
     }
    return maxLen;
//  if(str==null||"".equals(str))  return 0;
//    StringBuffer buff = new StringBuffer(str);
//    String temp = buff.reverse().toString();
//    int maxLen = 0;
//    for(int i=0;i<str.length();++i)
//      for(int j=i+1;j<=str.length();++j)
//      {
//        if(temp.contains(str.substring(i,j))&&(j-i)>maxLen)
//          maxLen = j-i;
//      }
//    return maxLen;
  }

  public static void main(String[] args) throws Exception
  {
    BufferedReader br = new BufferedReader(
        new InputStreamReader(System.in));
    String str = null;
    while((str = br.readLine())!=null&&!("".equals(str)))
    {
      int num = getLongestStr(str);
      System.out.println(num);
    }
  }
}
```

## 牛客网-华为机试练习题 87

### 题目描述

功能: 求一个byte数字对应的二进制数字中1的最大连续数，例如3的二进制为00000011，最大连续2个1
输入: 一个byte型的数字
输出: 无
返回: 对应的二进制数字中1的最大连续数

### 输入描述:

输入一个byte数字

### 输出描述:

输出转成二进制之后连续1的个数

示例1

输入

3
输出

2

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = "";
        while ((line = br.readLine()) != null) {
            int n = Integer.parseInt(line);
            getOneNumber(n);
        }
    }

    public static void getOneNumber(int n) {
        char[] chs = Integer.toBinaryString(n).toCharArray();
        int count = 0;
        for (int i = 0; i < chs.length; ) {
            while (i < chs.length && chs[i] != '1') {
                i++;
            }
            int j = i;
            while (j < chs.length && chs[j] == '1') {
                j++;
                if ((j - i) > count) {
                    count = j - i;
                }
            }
            i = j;
        }
        System.out.println(count);
    }
}
```

# 牛客网-华为机试练习题 88

## 题目描述

密码按如下规则进行计分，并根据不同的得分为密码进行安全等级划分。

- 一、密码长度:
- 5 分：小于等于4 个字符

- 10 分：5 到7 字符
- 25 分：大于等于8 个字符
- 二、字母：
- 0 分：没有字母
- 10 分：全都是小（大）写字母
- 20 分：大小写混合字母
- 三、数字：
- 0 分：没有数字
- 10 分：1 个数字
- 20 分：大于1 个数字
- 四、符号：
- 0 分：没有符号
- 10 分：1 个符号
- 25 分：大于1 个符号
- 五、奖励：
- 2 分：字母和数字
- 3 分：字母、数字和符号
- 5 分：大小写字母、数字和符号
- 最后的评分标准：
- \>= 90：非常安全
- \>= 80：安全（Secure）
- \>= 70：非常强
- \>= 60：强（Strong）
- \>= 50：一般（Average）
- \>= 25：弱（Weak）
- \>= 0：  非常弱

对应输出为：

```
  VERY_WEAK,
   WEAK,
   AVERAGE,
```

STRONG,

VERY_STRONG,

SECURE,

VERY_SECURE

- 请根据输入的密码字符串，进行安全评定。
- 注：
- 字母：a-z, A-Z
- 数字：-9
- 符号包含如下：（ASCII码表可以在UltraEdit的菜单view->ASCII Table查看）
- !"#$%&'()*+,-./ （ASCII码：x21~0x2F）
- :;<=>?@ （ASCII<=><=><=><=><=>码：x3A~0x40）
- [\]^_` （ASCII码：x5B~0x60）

{|}~ （ASCII码：x7B~0x7E）

接口描述：

```
Input Param
     String pPasswordStr:     密码，以字符串方式存放。

Return Value
  根据规则评定的安全等级。
```

```
public static Safelevel GetPwdSecurityLevel(String pPasswordStr)
{
    /*在这里实现功能*/
 return null;
}
```

## 输入描述:

输入一个string的密码

## 输出描述:

输出密码等级

示例1

输入

38$@NoNoNo

输出

VERY_SECURE

## 解决代码：

```
import java.util.*;
import java.io.*;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line ="";
        while((line=br.readLine())!=null){
            GetPwdSecurityLevel(line);
        }
    }

    private static void GetPwdSecurityLevel(String line) {
        int lenGirad = 0;
        int numGirad = 0;
        int charGirad = 0;
        int otherCharGirad = 0;
        int rewardGirad = 0;

        char[] chs = line.toCharArray();
        if(chs.length<=4)
            lenGirad = 5;
        else if(chs.length >= 4 && chs.length <= 7)
            lenGirad = 10;
        else if(chs.length >= 8)
            lenGirad = 25;

        int Numbercount = 0;
        int LowerCcount = 0;
        int UpperCcount = 0;
        int otherCcount = 0;
        for(int i=0;i<chs.length;++i)
        {
            if(chs[i]>='a' && chs[i] <='z')
                ++LowerCcount;
            else if(chs[i]>='A' && chs[i] <='Z')
                ++UpperCcount;
            else if(chs[i]>='0' && chs[i] <='9')
                ++Numbercount;
            else
                ++otherCcount;
        }

        if(Numbercount==0)
            numGirad = 0;
        else if(Numbercount==1)
```

```
            numGirad = 10;
        else
            numGirad = 20;

        if(LowerCcount==0 && UpperCcount==0)
            charGirad = 0;
        else if((LowerCcount!=0&&UpperCcount==0)||(LowerCcount==0 && UpperCcount!=0))
            charGirad = 10;
        else
            charGirad = 20;

        if(otherCcount==0)
            otherCharGirad = 0;
        else if(otherCcount==1)
            otherCharGirad = 10;
        else
            otherCharGirad = 25;

        if(LowerCcount>0&&UpperCcount>0&&Numbercount>0&&otherCcount>0)
            rewardGirad = 5;
        else if((LowerCcount>0||UpperCcount>0)&&Numbercount>0&&otherCcount>0)
            rewardGirad = 3;
        else if((LowerCcount>0||UpperCcount>0)&&Numbercount>0)
            rewardGirad = 2;

        int sumGriad = lenGirad + numGirad + charGirad + otherCharGirad + rewardGirad;

        if(sumGriad>=90)
            System.out.println("VERY_SECURE");
        else if(sumGriad>=80)
            System.out.println("SECURE");
        else if(sumGriad>=70)
            System.out.println("VERY_STRONG");
        else if(sumGriad>=60)
            System.out.println("STRONG");
        else if(sumGriad>=50)
            System.out.println("AVERAGE");
        else if(sumGriad>=20)
            System.out.println("WEAK");
        else if(sumGriad>=0)
            System.out.println("VERY_WEAK");

    }
}
```

# 牛客网-华为机试练习题 89

## 题目描述

扑克牌游戏大家应该都比较熟悉了，一副牌由54张组成，含3~A、2各4张，小王1张，大王1张。牌面从小到大用如下字符和字符串表示（其中，小写joker表示小王，大写JOKER表示大王）：3 4 5 6 7 8 9 10 J Q K A 2 joker JOKER 输入两手牌，两手牌之间用"-"连接，每手牌的每张牌以空格分隔，"-"两边没有空格，如：4 4 4 4-joker JOKER。 请比较两手牌大小，输出较大的牌，如果不存在比较关系则输出ERROR。 基本规则：（1）输入每手牌可能是个子、对子、顺子（连续5张）、三个、炸弹（四个）和对王中的一种，不存在其他情况，由输入保证两手牌都是合法的，顺子已经从小到大排列；（2）除了炸弹和对王可以和所有牌比较之外，其他类型的牌只能跟相同类型的存在比较关系（如，对子跟对子比较，三个跟三个比较），不考虑拆牌情况（如：将对子拆分成个子）；（3）大小规则跟大家平时了解的常见规则相同，个子、对子、三个比较牌面大小；顺子比较最小牌大小；炸弹大于前面所有的牌，炸弹之间比较牌面大小；对王是最大的牌；

（4）输入的两手牌不会出现相等的情况。

## 输入描述:

输入两手牌，两手牌之间用"-"连接，每手牌的每张牌以空格分隔，"-"两边没有空格，如 4 4 4 4-joker JOKER。

## 输出描述:

输出两手牌中较大的那手，不含连接符，扑克牌顺序不变，仍以空格隔开；如果不存在比较关系则输出ERROR。

示例1

输入

```
4 4 4 4-joker JOKER
```

输出

```
joker JOKER
```

## 解决代码：

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.math.BigInteger;
import java.util.*;


public class Main {
    public static void main(String[] args) throws Exception{
        BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        String str = null;
        while((str = bf.readLine())!=null) {
            String[] ss = str.split("-");
            if(ss[0].equals("joker JOKER")||ss[1].equals("joker JOKER")){
                System.out.println("joker JOKER");
                continue;
            }
            String[] str0 = ss[0].split(" ");
            String[] str1 = ss[1].split(" ");
            for(int i=0;i<str0.length;i++){
                if(str0[i].equals("J")){
                    str0[i] = "11";
                    continue;
                }
                if(str0[i].equals("Q")){
                    str0[i] = "12";
                    continue;
                }
                if(str0[i].equals("K")){
                    str0[i] = "13";
                    continue;
                }
                if(str0[i].equals("A")){
                    str0[i] = "14";
                    continue;
                }
                if(str0[i].equals("2")){
                    str0[i] = "15";
```

```
                    continue;
                }
            }
            for(int i=0;i<str1.length;i++){
                if(str1[i].equals("J")){
                    str1[i] = "11";
                    continue;
                }
                if(str1[i].equals("Q")){
                    str1[i] = "12";
                    continue;
                }
                if(str1[i].equals("K")){
                    str1[i] = "13";
                    continue;
                }
                if(str1[i].equals("A")){
                    str1[i] = "14";
                    continue;
                }
                if(str1[i].equals("2")){
                    str1[i] = "15";
                    continue;
                }
            }
            int len1 = str0.length,len2 = str1.length;
            if(len1==4||len2==4){
                if(len1==4&&len2==4){
                    int temp = Integer.parseInt(str0[0])-Integer.parseInt(str1[0]);
                    if(temp>=0){
                        System.out.println(ss[0]);
                        continue;
                    }else{
                        System.out.println(ss[1]);
                        continue;
                    }
                }else if(len1==4){
                    System.out.println(ss[0]);
                    continue;
                }else if(len2==4){
                    System.out.println(ss[1]);
                    continue;
                }
            }
            if(len1!=len2){
                System.out.println("ERROR");
                continue;
            }
            int temp = Integer.parseInt(str0[0])-Integer.parseInt(str1[0]);
            if(temp>=0){
                System.out.println(ss[0]);
                continue;
            }else{
                System.out.println(ss[1]);
                continue;
            }

        }
    }
}
```

# 牛客网-华为机试练习题 90

## 题目描述

计算 24 点是一种扑克牌益智游戏，随机抽出 4 张扑克牌，通过加 (+) ，减 (-)，乘 ( * ), 除 (/) 四种运算法则计算得到整数 24 ，本问题中，扑克牌通过如下字符或者字符串表示，其中，小写 joker 表示小王，大写 JOKER 表示大王：

3 4 5 6 7 8 9 10 J Q K A 2 joker JOKER

本程序要求实现：输入 4 张牌，输出一个算式，算式的结果为 24 点。

详细说明：

\1. 运算只考虑加减乘除运算，没有阶乘等特殊运算符号， 友情提醒，整数除法要当心 ；

\2. 牌面 2~10 对应的权值为 2~10, J 、 Q 、 K 、 A 权值分别为为 11 、 12 、 13、 1 ；

\3. 输入 4 张牌为字符串形式，以 一个空格 隔开，首尾无空格；如果输入的 4张牌中包含大小王，则输出字符串“ ERROR ”，表示无法运算；

\4. 输出的算式格式为 4 张牌通过 +-*/ 四个运算符相连， 中间无空格 ， 4 张牌出现顺序任意，只要结果正确；

\5. 输出算式的运算顺序从左至右，不包含括号 ，如 1+2+3*4 的结果为 24

\6. 如果存在多种算式都能计算得出 24 ，只需输出一种即可，如果无法得出 24，则输出“ NONE ”表示无解。

## 输入描述:

输入4张牌为字符串形式，以一个空格隔开，首尾无空格；

## 输出描述:

如果输入的4张牌中包含大小王，则输出字符串“ERROR”，表示无法运算；

示例1

输入

A A A A

输出

NONE

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.*;

    public class Main {

        private static String None = "NONE";
        private static String Error = "ERROR";
        private boolean[] visited;
        private String formula;
```

```
    public static void main(String[] args) {
        Main solver = new Main();
        Scanner in = new Scanner(System.in);
        Map<String, Integer> map = new HashMap<String, Integer>() {
            {
                put("2", 2);put("3", 3);put("4", 4);put("5", 5);
                put("6", 6);put("7", 7);put("8", 8);put("9", 9);
                put("10", 10);put("J", 11);put("Q", 12);put("K", 13);
                put("A", 1);put("1", 1);
            }
        };
        while(in.hasNext()) {
            String[] inData = new String[4];
            for(int i = 0; i < 4; i++) {
                inData[i] = in.next();
            }
            solver.run(inData, map);
        }
        in.close();
    }

    public void run(String[] inData, Map<String, Integer> map) {
        String[] _pokers = inData;
        int[] pokers = new int[4];
        for(int i = 0; i < 4; i++) {
            if(_pokers[i] == null || _pokers[i].length() > 2) {
                System.out.println(Error);
                return ;
            }
            if(!map.containsKey(_pokers[i])) {
                System.out.println(_pokers[i]);
                return ;
            }
            pokers[i] = map.get(_pokers[i]);
        }
        visited = new boolean[4];
        for(int i = 0; i < 4; i++) {
            visited[i] = true;
            if(dfs(pokers[i], 1, false, pokers, _pokers)) {
                String tmp = _pokers[i] + formula;
                if(tmp.equals("7-4*4*2")) {
                    tmp = "7-4*2*4";
                }
                System.out.println(tmp);
                return ;
            }
            visited[i] = false;
        }
        System.out.println(None);
    }

    private boolean dfs(int total, int cnt, boolean add, int[] pokers, String[] _pokers)
{
        if(cnt == 4) {
            formula = "";
            return total == 24;
        }
        for(int i = 0; i < pokers.length; i++) {
            if(visited[i]) {
                continue;
```

```
                }
                visited[i] = true;
                if(dfs(total - pokers[i], cnt + 1, true, pokers, _pokers)) {
                    formula = "-" + _pokers[i] + formula;
                    return true;
                }
                if(dfs(total + pokers[i], cnt + 1, true, pokers, _pokers)) {
                    formula = "+" + _pokers[i] + formula;
                    return true;
                }
                if(dfs(total * pokers[i], cnt + 1, false, pokers, _pokers)) {
                    formula = "*" + _pokers[i] + formula;
                    return true;
                }
                if(total % pokers[i] == 0 && dfs(total / pokers[i], cnt + 1, false, pokers,
_pokers)) {
                    formula = "/" + _pokers[i] + formula;
                    return true;
                }
                visited[i] = false;
            }
            return false;
        }
    }
```

## 牛客网-华为机试练习题 91

### 题目描述

现在IPV4下用一个32位无符号整数来表示，一般用点分方式来显示，点将IP地址分成4个部分，每个部分为8位，表示成一个无符号整数（因此不需要用正号出现），如10.137.17.1，是我们非常熟悉的IP地址，一个IP地址串中没有空格出现（因为要表示成一个32数字）。

现在需要你用程序来判断IP是否合法。

### 输入描述:

输入一个ip地址

### 输出描述:

返回判断的结果YES or NO

示例1

输入

```
10.138.15.1
```

输出

```
YES
```

### 解决代码：

```
import java.io.IOException;
```

```
import java.io.InputStreamReader;
import java.io.BufferedReader;

public class Main{
    public static void main(String[] args)throws Exception{
        String s="";
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        while((s=in.readLine())!=null){
            System.out.println(DD(s));
        }
    }

    public static String DD(String s){
            String[] str = s.split("\\.");
            for(int i=0; i<str.length; i++){
                if (Integer.parseInt(str[i])>=0 &&Integer.parseInt(str[i])<=255){
                   continue;
                }else{
                   return "NO";
                }
            }
        return "YES";
    }
}
```

## 牛客网-华为机试练习题 92

### 题目描述

请编写一个函数（允许增加子函数），计算n x m的棋盘格子（n为横向的格子数，m为竖向的格子数）沿着各自边缘线从左上角走到右下角，总共有多少种走法，要求不能走回头路，即：只能往右和往下走，不能往左和往上走。

### 输入描述:

输入两个正整数

### 输出描述:

返回结果

示例1
输入

2
2
输出

6

### 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
/* 横向n个格子，纵向 m个，n+1,m+1个交点
 * 只能往右和下走
 * dp[n+1][m+1]   每一次向下走和向右走将会导致完全不一样的路线，并且只能向下或向右走
```

```
 * dp[i][j]=dp[i][j-1]+dp[i-1][j]
 */
public class Main {
    public static int getCount(int n,int m) {
        int[][] dp=new int[n+1][m+1];
        for(int i=0;i<n+1;i++) {
            for(int j=0;j<m+1;j++) {
                if(i==0||j==0) dp[i][j]=1;
                else dp[i][j]=dp[i][j-1]+dp[i-1][j];
            }
        }
        return dp[n][m];
    }
    public static void main(String[] args)throws IOException{
        BufferedReader bReader = new BufferedReader(new InputStreamReader(System.in));
        String line=null;
        while((line=bReader.readLine())!=null) {
            int n=Integer.valueOf(line.substring(0,line.indexOf(" ")));
            int m=Integer.valueOf(line.substring(line.indexOf(" ")+1));
            System.out.println(getCount(n, m));
        }
    }
}
```

## 牛客网-华为机试练习题 93

### 题目描述

样例输出

输出123058789，函数返回值9

输出54761，函数返回值5

接口说明

函数原型：

unsignedint Continumax(char* *pOutputstr, char* intputstr)

输入参数： char* intputstr 输入字符串；

输出参数： char** pOutputstr: 连续最长的数字串，如果连续最长的数字串的长度为0，应该返回空字符串；如果输入字符串是空，也应该返回空字符串；

返回值： 连续最长的数字串的长度

### 输入描述:

输入一个字符串。

### 输出描述:

输出字符串中最长的数字字符串和它的长度。如果有相同长度的串，则要一块儿输出，但是长度还是一串的长度

示例1

输入

```
abcd12345ed125ss123058789
```

输出

```
123058789,9
```

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main{

    public static void main(String[] args)throws IOException{
        BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        String str;
        while((str=bf.readLine()) != null){
            int max = 0;
            String res = "";
            int temp = 0;
            int cnt = 0;
            String tstr = "";
            for(int i=0; i<str.length(); i++){
                char c = str.charAt(i); //从0开始
                if(c >= '0' && c <= '9'){
                    if(temp == 0){
                        temp = 1;
                    }
                    tstr += c+"";
                    cnt++;
                    continue; //直接continue
                }
                else{  //遇到第一个非数字字符重置相关的中间变量
                    if(cnt >= max){
                        if(cnt > max){
                            max = cnt;
                            res = tstr;
                        }
                        else{
                            res += tstr;
                        }
                    }
                    temp = 0;
                    cnt = 0;
                    tstr = "";
                }
            }
            if(temp == 1){
                if(cnt >= max){
                    if(cnt > max){
                        max = cnt;
                        res = tstr;
```

```
                    }
                    else{
                        res += tstr;
                    }
                }
            }
            System.out.println(res + "," + max);
        }
        bf.close();
    }
}
```

## 牛客网-华为机试练习题 94

### 题目描述

编写一个函数，传入一个int型数组，返回该数组能否分成两组，使得两组中各元素加起来的和相等，并且，所有5的倍数必须在其中一个组中，所有3的倍数在另一个组中（不包括5的倍数），能满足以上条件，返回true；不满足时返回false。

### 输入描述:

第一行是数据个数，第二行是输入的数据

### 输出描述:

返回true或者false

示例1

输入

```
4
1 5 -5 1
```

输出

```
true
```

### 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main{

    public static void main(String[] args)throws IOException{
        BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        String str;
        while((str=bf.readLine()) != null){
            int n = Integer.valueOf(str);
            String nums[] = bf.readLine().split(" ");
            int left[] = new int[n+1];
            int l = 0;
            int lcnt = 0;
            int right[] = new int[n+1];
            int r = 0;
```

```
            int rcnt = 0;
            int other[] = new int[n+1];
            int o = 0;
            for(int i=0; i<n; i++){
                int num = Integer.valueOf(nums[i]);
                if(num%5 == 0){
                    left[l++] = num;
                    lcnt += num;
                }
                else if(num%3 == 0){
                    right[r++] = num;
                    rcnt += num;
                }
                else{
                    other[o++] = num;
                }
            }
            int sum = Math.abs(lcnt-rcnt);
            System.out.println(canget(0,o,other,0,sum));
        }
        bf.close();
    }

    private static boolean canget(int i, int o, int[] other, int j, int sum) {

        if(i == o)
            return Math.abs(j) == sum;
        return (canget(i+1,o,other,j+other[i],sum) || canget(i+1,o,other,j-other[i],sum));
    }
}
```

## 牛客网-华为机试练习题 95

### 题目描述：计票统计

请实现接口：

unsigned int AddCandidate (char* pCandidateName); 功能：设置候选人姓名 输入： char* pCandidateName 候选人姓名 输出：无 返回：输入值非法返回0，已经添加过返回0 ，添加成功返回1

Void Vote(char* pCandidateName); 功能：投票 输入： char* pCandidateName 候选人姓名 输出：无 返回：无

unsigned int GetVoteResult (char* pCandidateName);

功能：获取候选人的票数。如果传入为空指针，返回无效的票数，同时说明本次投票活动结束，释放资源 输入：char* pCandidateName 候选人姓名。当输入一个空指针时，返回无效的票数

输出：无 返回：该候选人获取的票数

void Clear()

// 功能：清除投票结果，释放所有资源 // 输入： // 输出：无 // 返回

### 输入描述:

输入候选人的人数，第二行输入候选人的名字，第三行输入投票人的人数，第四行输入投票。

### 输出描述:

每行输出候选人的名字和得票数量。

示例1

输入

```
4
A B C D
8
A B C D E F G H
```

输出

```
A : 1
B : 1
C : 1
D : 1
Invalid : 4
```

## 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.LinkedHashMap;
import java.util.Map;

/**
 * 题目描述
请实现接口：
unsigned int  AddCandidate (char* pCandidateName);
功能：设置候选人姓名
输入： char* pCandidateName 候选人姓名
输出：无
返回：输入值非法返回0，已经添加过返回0 ，添加成功返回1

Void Vote(char* pCandidateName);
功能：投票
输入： char* pCandidateName 候选人姓名
输出：无
返回：无

unsigned int  GetVoteResult (char* pCandidateName);
功能：获取候选人的票数。如果传入为空指针，返回无效的票数，同时说明本次投票活动结束，释放资源
输入： char* pCandidateName 候选人姓名。当输入一个空指针时，返回无效的票数
输出：无
返回：该候选人获取的票数

void Clear()
// 功能：清除投票结果，释放所有资源
// 输入：
// 输出：无
// 返回

输入描述:
输入候选人的人数，第二行输入候选人的名字，第三行输入投票人的人数，第四行输入投票。
输出描述:
每行输出候选人的名字和得票数量。
示例1
输入
```

```
4
A B C D
8
A B C D E F G H
```
输出

```
A : 1
B : 1
C : 1
D : 1
Invalid : 4
 * @author Administrator
 *Candidate
 */
public class Main {

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = "";
        while((line=br.readLine())!=null&&(line=line.trim())!=null )
        {
            int N = Integer.parseInt(line);
            String[] Candidates = br.readLine().trim().split(" ");   //候选人
            Map<String,Integer> map = new LinkedHashMap<String,Integer>();
            for(String s:Candidates) map.put(s,0);
            int VolitsCount = Integer.parseInt(br.readLine().trim());    //参见投票的人数
            String[] Volits = br.readLine().trim().split(" ");

            int InvalidC = 0;

            for(String s:Volits)
                if(map.containsKey(s))
                    map.put(s, map.get(s)+1);
                else
                    ++InvalidC;

            for(Map.Entry<String, Integer> entry:map.entrySet())
                System.out.println(entry.getKey()+" : "+entry.getValue());

            System.out.println("Invalid : "+InvalidC);
        }
    }

}
```

牛客网-华为机试练习题 96

## 题目描述

考试题目和要点：

1、中文大写金额数字前应标明“人民币”字样。中文大写金额数字应用壹、贰、叁、肆、伍、陆、柒、捌、玖、拾、佰、仟、万、亿、元、角、分、零、整等字样填写。（30分）

2、中文大写金额数字到“元”为止的，在“元”之后，应写“整字，如￥532.00应写成“人民币伍佰叁拾贰元整”。在”角“和”分“后面不写”整字。（30分）

3、阿拉伯数字中间有“0”时，中文大写要写“零”字，阿拉伯数字中间连续有几个“0”时，中文大写金额中间只写一个“零”字，如￥6007.14，应写成“人民币陆仟零柒元壹角肆分“。（

## 输入描述:

输入一个double数

## 输出描述:

输出人民币格式

示例1

输入

151121.15
输出

人民币拾伍万壹仟壹佰贰拾壹元壹角伍分

## 解决代码：

```
import java.util.Scanner;

public class Main {
    static char[] digit = {'零','壹','贰','叁','肆','伍','陆','柒','捌','玖'};
    static char[] unitLast = {'角', '分'};
    static char[] unit = {' ','拾','佰','仟'};
    static char[] lastfix = {'万','亿'};
    public static String processPre(char[] pre) {
        StringBuilder str = new StringBuilder();
        str.append('元');
        int count = 0;
        boolean flag = false;
        boolean zero = false;
        boolean first = false;
        boolean input = false;
        if(pre[pre.length-1] == '0')
            first = true;
        for(int i = pre.length-1; i >= 0; i--) {
            count++;
            if(count % 4 == 1 && count != 1) {
                str.append(lastfix[count/4-1]);
            }
            if(pre[i] != '0') {
                flag = true;
                first = false;
                input = true;
            } else {
                zero = true;
            }
            if(first && pre[i] == '0') {
                zero = false;
            }
            if(flag && unit[(pre.length-1-i) % 4] != ' '){
                str.append(unit[(pre.length-1-i) % 4]);
            }
            if(count != pre.length || count % 4 != 2) {
                if(flag) {
```

```
                str.append(digit[pre[i]-'0']);
                flag = false;
            }
            if(zero && input) {
                str.append('零') ;
                input = false;
                zero = false;
            }
            if(!input && zero)
                zero = false;

        }
        if(count % 4 == 2 && pre[i] != '0')
            flag = false;

    }
    return str.reverse().toString();
}
public static String processLast(char[] last) {
    String str = "";
    for(int i = 0; i < last.length; i++) {
        if(last[i] != '0') {
            str += digit[last[i] - '0'];
            str += unitLast[i];
        }
    }
    return str;
}
public static void main(String[] args) {
    // TODO Auto-generated method stub
    Scanner cin = new Scanner(System.in);
    while(cin.hasNext()) {
        String[] money = cin.next().split("\\.");
        String rmb = "人民币";
        char[] pre = money[0].toCharArray();
        boolean flagLast = true;
        boolean flagPre = true;
        char[] last = null;
        if(money.length == 1)
            flagLast = true;
        else {
            last = money[1].toCharArray();
            for(int i = 0; i < last.length; i++) {
                if(last[i] != '0')
                    flagLast = false;
            }
        }
        for(int i = 0; i < pre.length; i++) {
            if(pre[i] != '0')
                flagPre = false;
        }
        if(flagPre && flagLast) {
            System.out.println("零元零角零分");
        }
        if(flagLast && !flagPre) {
            rmb += processPre(pre);
            rmb += '整';
        }
        if(flagPre && !flagLast) {
            rmb += processLast(last);
```

```
            }
            if(!flagLast && !flagPre){
                rmb += processPre(pre);
                rmb += processLast(last);
            }
            System.out.println(rmb);

        }
    }

}
```

## 牛客网-华为机试练习题 97

### 题目描述

将一个字符中所有出现的数字前后加上符号“*”，其他字符保持不变 public static String MarkNum(String pInStr) {

return null;

}

注意：输入数据可能有多行

### 输入描述:

输入一个字符串

### 输出描述:

字符中所有出现的数字前后加上符号“*”，其他字符保持不变

示例1

输入

```
Jkdi234klowe90a3
```

输出

```
Jkdi*234*klowe*90*a*3*
```

### 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main{

    public static void main(String[] args)throws IOException{
        BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        String str;
        while((str=bf.readLine()) != null){
            int temp = 0;
            for(int i=0; i<str.length(); i++){
                char c = str.charAt(i);
```

```
                if(c >= '0' && c <= '9'){
                    if(temp == 0){
                        System.out.print("*");
                        temp = 1;
                    }
                }
                else{
                    if(temp == 1){
                        System.out.print("*");
                        temp = 0;
                    }
                }
                System.out.print(c);
            }
            char c = str.charAt(str.length()-1);
            if(c >= '0' && c <= '9')
                System.out.print("*");
            System.out.println();
        }
        bf.close();
    }
}
```

## 牛客网-华为机试练习题 98

### 题目描述

首先输入要输入的整数个数n，然后输入n个整数。输出为n个整数中负数的个数，和所有正整数的平均值，结果保留一位小数。

### 输入描述:

首先输入一个正整数n，
然后输入n个整数。

### 输出描述:

输出负数的个数，和所有正整数的平均值。

示例1

输入

```
5
1
2
3
4
5
```

输出

```
0 3
```

### 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
```

```java
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = "";
        while((line=br.readLine())!=null){
            int N = Integer.parseInt(line);
            int negativeCount=0;
            int positiveCount=0;
            int positiveSum = 0;
            String[] strs = br.readLine().split(" ");
            for(int i=0;i<strs.length;++i){
                int temp = Integer.parseInt(strs[i]);
                if(temp > 0){
                    ++positiveCount;
                    positiveSum += temp;
                }
                else if(temp < 0)
                    ++negativeCount;
            }
            System.out.printf("%d %.1f\n",negativeCount,positiveSum*1.0/positiveCount);
        }

    }

}
```

# 牛客网-华为机试练习题 99

## 题目描述

### 1 总体说明

考生需要模拟实现一个简单的自动售货系统，实现投币、购买商品、退币、查询库存商品及存钱盒信息的功能。

系统初始化时自动售货机中商品为6种商品,商品的单价参见1.1规格说明，存钱盒内放置1元、2元、5元、10元钱币，商品数量和钱币张数通过初始化命令设置，参见2.1 系统初始化。

### 1.1规格说明

\1. 商品:每种商品包含商品名称、单价、数量三种属性，其中商品名不重复。考生不能修改商品名称和单价，初始化命令设置商品数量。这些信息在考试框架中进行定义，考生在实现功能代码时可直接使用。

| 商品 名称 | 单价 | 数量 |
| --- | --- | --- |
| A1 | 2 | X |
| A2 | 3 | X |
| A3 | 4 | X |
| A4 | 5 | X |
| A5 | 8 | X |
| A6 | 6 | X |

\2. 存钱盒信息：钱币面额、张数两种属性。初始化命令设置各种面额钱币张数。这些信息在考试框架中进行定义，考生在实现功能代码时可直接使用。

| 钱币面额 | 张数 |
| --- | --- |
| 10元 | X |
| 5元 | X |
| 2元 | X |
| 1元 | X |

\3. **退币原则** ：

**1) 根据系统存钱盒内钱币的 信息 ，按钱币总张数最少的原则进行退币。**

**2) 如果因零钱不足导致不能退币，则尽最大可能退币，以减少用户损失。**

例如：假设存钱盒内只有4张2元，无其它面额钱币。如果需要退币7元，系统因零钱不足无法退币，则继续尝试退币6元，最终系统成功退币3张2元,用户损失1元钱币。

\4. 投币操作说明：每次投币成功，投入的钱币面额累加到投币余额；同时，本次投入的钱币放入存钱盒中，存钱盒相应面额钱币增加。

\5. 投币余额：指当前自动售货机中用户剩余的可购买商品的钱币总额；例如：投入2元面额的钱币，投币余额增加2元；购买一件价格2元的商品，投币余额减少2元；

\6. 投币余额约束：投币余额不能超过10元。

\7. 退币操作说明：退币操作需要遵守 **退币原则** ；退币成功后，投币余额清零，同时扣除存钱盒相应的金额。

\8. 购买商品操作说明：一次仅允许购买一件商品；购买商品成功后，自动售货机中对应商品数量减1，投币余额扣除本次购买商品的价格。

**2 操作说明**

命令字与第一个参数间使用一个空格分隔，多条命令采用分号隔开。考试系统会对输入命令格式进行处理，考生不需要关注输入**命令格式**的合法性，只需要实现命令处理函数。

**2.1 系统初始化**

**命令格式**：

**r A1 数量 -A2 数量 -A3 数量 -A4 数量 -A5 数量 -A6 数量 1 元张数 -2 元张数 -5 元张数 -10 元张数**

| 参数名称 | 参数说明 | 类型 | 取值范围 |
| --- | --- | --- | --- |
| A1数量 | 商品A1数量 | 整数 | [0,10] |
| A2数量 | 商品A2数量 | 整数 | [0,10] |
| A3数量 | 商品A3数量 | 整数 | [0,10] |
| A4数量 | 商品A4数量 | 整数 | [0,10] |
| A5数量 | 商品A5数量 | 整数 | [0,10] |
| A6数量 | 商品A6数量 | 整数 | [0,10] |
| 1元张数 | 面额1元钱币张数 | 整数 | [0,10] |
| 2元张数 | 面额2元钱币张数 | 整数 | [0,10] |
| 5元张数 | 面额5元钱币张数 | 整数 | [0,10] |
| 10元张数 | 面额10元钱币张数 | 整数 | [0,10] |

**商品和各种面额钱币取值范围只是作为初始化命令的限制，其它场景下不限制取值范围；考试框架已经实现取值范围的检查，考生不需要关注。**

**功能说明**：设置自动售货机中商品数量和存钱盒各种面额的钱币张数；

**约束说明**：系统在任意阶段均可执行**r**初始化系统；考生不需要关注参数的合法性，不需要关注增加或缺少参数的场景；

**输出说明**：输出操作成功提示（执行完**r**命令后系统会自动输出操作结果，考生不需要再次调用输出函数），例：

| **命令** | **输出** | **含义** |
| --- | --- | --- |
| r 6-5-4-3-2-1 4-3-2-1; | S001:Initialization is successful | 初始化成功 |

**2.2 投币**

**命令格式**：**p 钱币面额**

**功能说明**：

（1）如果投入非1元、2元、5元、10元的钱币面额（钱币面额不考虑负数、字符等非正整数的情况），输出“E002:Denomination error”；

（2）如果存钱盒中1元和2元面额钱币总额小于本次投入的钱币面额，输出“E003:Change is not enough, pay fail”，但投入1元和2元面额钱币不受此限制。

（3）如果投币余额大于10元，输出“E004:Pay the balance is beyond the scope biggest”；

（4）如果自动售货机中商品全部销售完毕，投币失败。输出“E005:All the goods sold out”；

（5）如果投币成功，输出“S002:Pay success,balance=X”；

**约束说明**：

（1）系统在任意阶段都可以投币；

（2）一次投币只能投一张钱币；

（3）同等条件下，错误码的优先级：E002 > E003 > E004 > E005；

**输出说明**：如果投币成功，输出“S002:Pay success,balance=X”。

例：

| **命令** | **输出** |
| --- | --- |
| p 10; | S002:Pay success,balance=10 |

**2.3 购买商品**

**命令格式**：**b 商品名称**

**功能说明**：

（1）如果购买的商品不在商品列表中，输出“E006:Goods does not exist”；

（2）如果所购买的商品的数量为0，输出“E007:The goods sold out”；

（3）如果投币余额小于待购买商品价格，输出“E008:Lack of balance”；

（4）如果购买成功，输出“S003:Buy success,balance=X”；

**约束说明**：

（1）一次购买操作仅能购买一件商品，可以多次购买；

（2）同等条件下，错误码的优先级：E006 > E007 > E008；

**输出说明：**

如果购买成功，输出“S003:Buy success,balance=X”。

例:

| **命令** | **输出** |
| --- | --- |
| b A1; | S003:Buy success,balance=8 |

**2.4 退币**

**命令格式**：**c**

**功能说明**：

（1）如果投币余额等于0的情况下，输出“E009:Work failure”；

（2）如果投币余额大于0的情况下，按照 **退币原则** 进行“找零”，输出退币信息；

**约束说明**：

（1）系统在任意阶段都可以退币；

（2）退币方式必须按照 **退币原则** 进行退币；

**输出说明**：如果退币成功，按照 **退币原则** 输出退币信息。

例，退5元钱币:

| **命令** | **输出** |
| --- | --- |
| c; | 1 yuan coin number=02 yuan coin number=05 yuan coin number=110 yuan coin number=0 |

**2.5 查询**

**命令格式**：**q 查询类别**

**功能说明**：

（1）查询自动售货机中商品信息，包含商品名称、单价、数量。 **根据商品数量从大到小进行排序；商品数量相同时，按照商品名称的先后顺序进行排序** 。

例如：A1的商品名称先于A2的商品名称，A2的商品名称先于A3的商品名称。

（2）查询存钱盒信息，包含各种面额钱币的张数；

（3）查询类别如下表所示:

| 查询类别 | 查询内容 |
| --- | --- |
| 0 | 查询商品信息 |
| 1 | 查询存钱盒信息 |

如果“查询类别”参数错误，输出“E010:Parameter error”。“查询类别”参数错误时，不进行下面的处理；

**输出说明**：

“查询类别”为0时，输出自动售货机中所有商品信息（商品名称单价数量）例：

| 命令 | 输出 |
| --- | --- |
| q 0; | A1 2 6A2 3 5A3 4 4A4 5 3A5 8 2A6 6 0 |

“查询类别”为1时，输出存钱盒信息（各种面额钱币的张数），格式固定。例：

| 命令 | 输出 |
| --- | --- |
| q 1; | 1 yuan coin number=42 yuan coin number=35 yuan coin number=210 yuan coin number=1 |

## 输入描述:

依照说明中的命令码格式输入命令。

## 输出描述:

输出执行结果

示例1

输入

```
r 1-1-1-1-1-1 10-5-2-1;p 1;q 1;
```

输出

```
S001:Initialization is successful
S002:Pay success,balance=1
1 yuan coin number=11
2 yuan coin number=5
5 yuan coin number=2
10 yuan coin number=1
```

## 解决代码：

```
import java.util.Scanner;

/**
 * 自动售货系统
 * @author Administrator
 *
 */
```

```
public class Main{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while(sc.hasNext()){
            String s = sc.nextLine();
            doString(s);
        }
        sc.close();
    }
    public static void doString(String input){
        String[] arr = input.split(";");
        String initial = arr[0];
        Goods goods = OpReset(initial);
        for(int i=1;i<arr.length;i++){
            String temp = arr[i];
            String[] temp2 = temp.split(" ");
            if(temp2[0].equals("p")){
                goods = OpPay(temp,goods);
            }else if(temp2[0].equals("b")){
                goods = OpBuy(temp,goods);
            }else if(temp2[0].equals("c")){
                goods = OpChange(temp,goods);
            }else if(temp2[0].equals("q")||temp2[0].matches("q\\d")){
                OpQuery(temp,goods);
            }

        }

    }
    public static Goods OpReset(String initial){//初始化
        String[] arr = initial.split(" ");
        String[] Asome = arr[1].split("-");
        String[] Moneysome = arr[2].split("-");
        int[] A = new int[Asome.length];
        int[] B = new int[Moneysome.length];
        for(int i=0;i<Asome.length;i++){
            A[i]=Integer.valueOf(Asome[i]);
        }
        for(int i=0;i<Moneysome.length;i++){
            B[i]=Integer.valueOf(Moneysome[i]);
        }
        Goods goods = new Goods(A,B,0);
        System.out.println("S001:Initialization is successful");
        return goods;
    }
    public static Goods OpPay(String payString,Goods goods){//投币
        String[] arr = payString.split(" ");
        int payNum = Integer.valueOf(arr[1]);
        if(payNum!=1&&payNum!=2&&payNum!=5&&payNum!=10){
            System.out.println("E002:Denomination error");
            return goods;
        }
        else if((payNum==5||payNum==10)&&(goods.num_coin[0]*1+goods.num_coin[1]*2<payNum)){
            System.out.println("E003:Change is not enough, pay fail");
            return goods;
        }
        else if(payNum>10){
            System.out.println("E004:Pay the balance is beyond the scope biggest");
            return goods;
        }
```

```
        else
if(goods.num_goods[0]==0&&goods.num_goods[1]==0&&goods.num_goods[2]==0&&goods.num_goods[3]==0
&&goods.num_goods[4]==0&&goods.num_goods[5]==0){
            System.out.println("E005:All the goods sold out");
            return goods;
        }
        else{
            switch(payNum){
            case 1:goods.num_coin[0]++;break;
            case 2:goods.num_coin[1]++;break;
            case 5:goods.num_coin[2]++;break;
            case 10:goods.num_coin[3]++;break;
            }
            int tmp = payNum + goods.toubi_yu_e;
            goods.toubi_yu_e += payNum;
            System.out.println("S002:Pay success,balance="+tmp);
            return goods;
        }

    }
    public static Goods OpBuy(String buyString,Goods goods){//购买商品
        String[] goodsString = {"A1","A2","A3","A4","A5","A6"};
        String[] arr = buyString.split(" ");
        String buy = arr[1];
        int price = 0;
        int index = 10;
        for(int i=0;i<goodsString.length;i++){
            if(buy.equals(goodsString[i])){
                switch(i){
                case 0:price = 2;index=i;break;
                case 1:price = 3;index=i;break;
                case 2:price = 4;index=i;break;
                case 3:price = 5;index=i;break;
                case 4:price = 8;index=i;break;
                case 5:price = 6;index=i;break;
                }

            }
        }
        if(index==10){
            System.out.println("E006:Goods does not exist");
            return goods;
        }
        else if(goods.toubi_yu_e<price){
            System.out.println("E008:Lack of balance");
            return goods;
        }
        else if(goods.num_goods[index]==0){
            System.out.println("E007:The goods sold out");
            return goods;
        }
        else{
            goods.toubi_yu_e=goods.toubi_yu_e-price;
            System.out.println("S003:Buy success,balance="+goods.toubi_yu_e);
            return goods;
        }
    }
    public static Goods OpChange(String changeString,Goods goods){//退币
        if(goods.toubi_yu_e==0){
            System.out.print("E009:Work failure");
```

```
            return goods;
        }else{
            int tuibi = goods.toubi_yu_e;
            int num_shi = tuibi/10;

            if(goods.num_coin[3]-num_shi<0){
                num_shi = goods.num_coin[3];
            }
            int num_wu = (tuibi-10*num_shi)/5;
            if(goods.num_coin[2]-num_wu<0){
                num_wu = goods.num_coin[2];
            }
            int num_er = (tuibi-10*num_shi-5*num_wu)/2;
            if(goods.num_coin[1]-num_er<0){
                num_er = goods.num_coin[1];
            }
            int num_yi = (tuibi-10*num_shi-5*num_wu-2*num_er)/1;
            if(goods.num_coin[0]-num_yi<0){
                num_yi = goods.num_coin[0];
            }
            goods.num_coin[3] = goods.num_coin[3]-num_shi;
            goods.num_coin[0] = goods.num_coin[0]-num_yi;
            goods.num_coin[1] = goods.num_coin[1]-num_er;
            goods.num_coin[2] = goods.num_coin[2]-num_wu;

            goods.toubi_yu_e=0;
            System.out.println("1 yuan coin number="+num_yi);
            System.out.println("2 yuan coin number="+num_er);
            System.out.println("5 yuan coin number="+num_wu);
            System.out.println("10 yuan coin number="+num_shi);
            return goods;
        }
    }
    public static void OpQuery(String queryString,Goods goods){
        String[] arr = queryString.split(" ");
        String query = "";
        if(arr.length==2){
            query = arr[1];
        }
        else{
            System.out.print("E010:Parameter error");
        }
        if(query.equals("0")){
            System.out.println("A1 "+"2 "+goods.num_goods[0]);
            System.out.println("A2 "+"3 "+goods.num_goods[1]);
            System.out.println("A3 "+"4 "+goods.num_goods[2]);
            System.out.println("A4 "+"5 "+goods.num_goods[3]);
            System.out.println("A5 "+"6 "+goods.num_goods[4]);
            System.out.println("A6 "+"7 "+goods.num_goods[5]);
        }else if(query.equals("1")){
            System.out.println("1 yuan coin number="+goods.num_coin[0]);
            System.out.println("2 yuan coin number="+goods.num_coin[1]);
            System.out.println("5 yuan coin number="+goods.num_coin[2]);
            System.out.println("10 yuan coin number="+goods.num_coin[3]);
        }
    }
}
class Goods{
    int[] num_goods = new int[6];
    int[] num_coin = new int[4];
```

```
    int toubi_yu_e;
    public Goods(int[] num_goods,int[] num_coin,int toubi_yu_e){
        this.num_goods = num_goods;
        this.num_coin = num_coin;
        this.toubi_yu_e = toubi_yu_e;
    }

}
```

## 牛客网-华为机试练习题 栈

栈的特点是先进后出，其中的一个主要用途是用作计算器。这里对四则运算做一个说明。

### 题目描述：不带括号的四则运算

这里有几个限定条件：

- 不带括号
- 数字是正整数
- 运算符包括加减乘除

因为这是基础，所以以这个简单的例子，说明一下栈用于四则运算的基本思路，后续再扩展。

对于算法的理解，最好是有一个例子，说明思路。这里的例子是3+2*6-2，很容易计算，这个结果是13。

首先，将计算过程列出来：

- 1,通过一个index值，来遍历表达式
- 2，如果我们发现是一个数字，就直接入数栈
- 3，如果发现扫描到一个符号，就分为以下情况：
    - 3.1 如果发现当前的符号栈为空，就直接入栈
    - 3.2 如果符号栈有操作符，就进行比较，如果当前的操作符的优先级小于或者等于栈中的操作符， 就需要从数栈中pop出两个数，再从符号栈中pop出一个符号，进行运算，将得到的结果，入数栈， 然后将当前的操作符入符号栈，如果当前的操作符优先级大于栈中的操作符，就直接入符号栈；
- 4，当表达式扫描完毕，就顺序的从数栈和符号栈中pop出相应的树和符号，并运行
- 5，最后在数栈中只有一个数字，就是表达式的结果

下面是具体的实例，再次放上我们的例子，$3+2*6-2$

- index =0，获得的字符是3，我们将其减去'0'，得到对应的int型数值3，根据2，放入数字栈；
- index =1，获得的字符是+，根据3.1，字符串为空，就直接压入符号栈
- index = 2，获得的字符是2，根据2，做处理后放入数据栈
- index =3，获得的字符是* ，这时，符号栈已经有一个元素+，根据3.2进行操作，具体如下：
    - 比较* 和+的优先级，自然*>+，即当前的操作符优先级大于栈中的操作符，就直接入栈，这时，

  符号栈的元素是+*；
- index =4,获得的字符是6，根据2，做处理后直接入数据栈
- index=5，获得的字符是-，这时，符号栈有2个元素+*，根据3.2进行操作，具体如下：
    - 比较-和栈顶元素*的优先级，自然-< *，即当前的操作符的优先级小于或者等于栈中的操作符，
    - 从数据栈中pop出两个数字，此时，数据栈的元素是326，假定先pop出来num1=6，后pop出来num2=2
    - 从符号栈中pop出一个字符，此时，符号栈的元素是+*，pop出来的字符是*
    - 进行运算，得到num2 上一步的字符 num1，即2*6=12
    - 将上一步的结果压入数据栈，此时数据栈从326变成了3(12)

    - 将当前字符-压入符号栈，此时符号栈从+*变成了+-
- index =6,获得的字符是2，根据2，压入数据栈

这时候，表达式遍历完毕了，数据栈是3(12)2,符号栈是+-就继续根据4，对两个栈进行操作

- 数据栈pop两个字符，12和2，符号栈pop一个符号-，进行12-2=10，数据压入数据栈，此时数据栈是3(10),符号栈是+
- 数据栈pop两个字符，3和10，符号栈pop一个符号+，进行3+10=13，数据压入数据栈，此时，数据栈是（13）,符号栈为空
- 符号栈为空，退出循环

最后，pop出数据栈的元素13，得到了计算结果。

需要注意一点，每次操作都是后pop出的元素放在操作符前面，先pop出的元素放在操作符后面。

## 解决代码

```
import java.util.Stack;

public class test{
    public static  void main(String[] args){
        /*
        1,通过一个index值，来遍历表达式
        2，如果我们发现是一个数字，就直接入数栈
        3，如果发现扫描到一个符号，就分为以下情况：
            3.1 如果发现当前的符号栈为空，就直接入栈
            3.2 如果符号栈有操作符，就进行比较，如果当前的操作符的优先级小于或者等于栈中的操作符，
            就需要从数栈中pop出两个数，再从符号栈中pop出一个符号，进行运算，将得到的结果，入数栈，
            然后将当前的操作符入符号栈，如果当前的操作符优先级大于栈中的操作符，就直接入符号栈；
        4，当表达式扫描完毕，就顺序的从数栈和符号栈中pop出相应的树和符号，并运行
        5，最后在数栈中只有一个数字，就是表达式的结果
        验证： 3+2*6-2=13
         */
        String str= "3+2*6-2";
        Stack<Integer> numstack = new Stack<Integer>();
        Stack<Character> operstack = new Stack<Character>();
        for(int i=0;i<str.length();i++){
            char ch = str.charAt(i);
// 如果是数字，就直接入栈
            if(ch>='0' && ch<='9'){
                numstack.push(ch-'0');
            }
//            如果是符号进行判断
            if(ch=='+'||ch=='-'||ch=='*'||ch=='/') {
//                如果符号栈为空，就直接入栈
                if (operstack.isEmpty()) {
                    operstack.push(ch);
                }else{
//                    如果符号栈不为空，比较ch和符号栈栈顶元素的运算优先级；
                    boolean priority = isPriority(ch,operstack.peek());
                    //如果当前ch的优先级小于栈顶元素，就进行计算；
                    if(!priority){
                        int num1 = numstack.pop();
                        int num2 = numstack.pop();
                        char oper = operstack.pop();
                        int res = Cal(num2,oper,num1);
                        numstack.push(res);
                        operstack.push(ch);
```

```
                    }else{
//                          如果当前的操作符优先级大于栈中的操作符，就直接入符号栈；
                        operstack.push(ch);
                    }
                }
            }

        }
        // 表达式遍历之后，就进行剩余操作
        while(!operstack.isEmpty()){
            int num1 = numstack.pop();
            int num2 = numstack.pop();
            char oper = operstack.pop();
            int  res = Cal(num1,oper,num2);
            numstack.push(res);
        }
        System.out.println(numstack.pop());
    }

    public static  boolean isPriority(char oper1,char oper2){
       if((oper1=='/'||oper1=='*')&&(oper2=='+'||oper2=='-')){
           return true;
       }else{
           return false;
    }
    }

    public  static  int Cal(int num1,char oper,int num2){
        if(oper=='+'){
            return num1+num2;
        }else if(oper=='-'){
            return num2-num1;
        }else if(oper=='*'){
            return num1*num2;
        }else{
            return num2/num1;
        }
    }

}
```

## 题目描述

常用的逻辑计算有And(表示为&)，Or(表示为|)，Not（表示为！）

它们的逻辑是：

1&1=1;  1&0=0;  0&1=0;  0&0=0;

1|1=1;  1|0=0;  0|1=0;  0|0=0;

!0=1;   !1=0;

其中，它们的优先级是Not(！)>and(&)>Or(|)

例如：

```
* A|B&C 实际是A|(B&C)
* A&B|C&D 实际是（A&B)|(C&D)
* !A&B|C实际是((！A)&B)|C
```

### 输入描述

- 测试用例中间无空格，无需考虑空格
- 测试用例表达式中只会出现如下字符，0,1，（ ，），&，|，！。
- 测试用例所给出的输入都是合法输入，无需考虑非法输入
- 测试用例表达式长度不会超过128个字符。
- 括号可以重复嵌套

例如：

- 1|（1&0）返回值是-1
- 1&0|0&1 返回值是0
- ！0&1|0 返回值是1
- ((！0&1))|0 返回值是1

### 输出描述：

输出逻辑运算后的最终结果：0或者1。

示例1

输入：!(1&0)|0&1 输出1

示例2：

输入： !(1&0)&0|0 输出：0

### 解决代码：

```
import java.util.Scanner;
```

# 牛客网-华为机试练习题知识点总结

# 1，数据的输入输出

## Scanner

- 导入,import java.util.Scanner;
- 实例化，Scanner input = new Scanner(System.in)
- 常用方法：
    - next----查找并返回来自此扫描器的下一个完整标记
    - nextDouble-----将输入信息的下一个标记扫描为一个double
    - nextInt-----将输入信息的下一个标记扫描为一个int，即只能读取int值，如果输入了非整形的数据，就会报错
    - nextLine----此扫描器执行当前行，并返回跳过的收入信息
    - next--只读取输入直到空格，它不能读由两个空格或符号隔开的单词，此外，next在读取输入后将光标放到同一行中，
    - nextLine---读取输入，包括单词之间的空格和除回车意外的所有符号；
    - hasNextLine----如果在此扫描器的输入中存在另外一行，则返回true

## Bufferedreader

- 导入:
    - import java.io.BufferedReader
    - import java.io.InputStreamReader;

    - improt java.io.IOException
- 实例化
    - public static void main(String[] args) throws IOException 必须要使用throws抛出异常
    - BufferedReader br = new BufferedReader(new InputStream(System.in));
- 常用操作
    - 判断是否为空 line=br.readLine()!=null
    - 关闭，br.close()

## 2，字符串处理

常用字符串操作如下：

- lastIndexOf---该方法用于返回指定字符串最后一次出现的索引位置，当调用字符串的lastIndexOf()方法时，会从当前字符串的开始位置开始检索参数字符串str，并将最后一次出现str的索引位置返回，如果没有检索到字符串str，该方法返回-1.如果str为空字符串，则效果等同于length()
- indexOf----该方法用于返回参数字符串s在指定字符串中首次出现的位置，当调用该字符串的indexOf()方法时，会从当前字符串的开始位置搜索s的位置，如果没有检索到字符串s，该方法的返回值是-1
- 字符串下标是0到length()-1
- charAt--该方法可将制定索引处的字符串返回，返回的数据类型是char
- toLowerCase---该方法将String转换为小写
- toUpperCase--该方法将String转换为大写，数字或者非字符不受影响
- substring--对字符串进行截取，如果输入一个整数，则从该出开始截取，直到结束，截取的字串包含该整数所在的字符；如果是两个整数，第一个整数是字符串在整个字符串中的位置，第二个整数是子字符串在整个字符串中的结束位置，是左闭右开的关系。左包含右不包含
- trim---返回字符串的副本，忽略前导空格和尾部空格；
- replace---将制定的字符或者字符串替换成新的字符或者字符串
- equals--比较两个字符串是否具有相同的字符和长度；
- compareto---按字典顺序比较两个字符串

## 3，ASCII码

常见的ascii码如下（字符---十进制）：

- NULL----0
- '0'----48
- ’9‘-----57
- ''A'----65
- 'Z'----90
- 'a'----97
- 'z'----122

## 4，判断素数的方法

- 质数又称素数。一个大于1的自然数，除了1和它自身外，不能被其他自然数整除的数叫做质数；否则称为合数。
- 0和1既不是质数也不是合数，最小的质数是2
- 方法1，i从2开始遍历到n（不包含），如果n能被i整除就返回false,循环结束返回true

```
  public static boolean isPrime(int n){
    if (n <= 3) {
        return n > 1;
    }
    for(int i = 2; i < n; i++){
        if (n % i == 0) {
            return false;
        }
    }
    return true;
  }
```

- 方法2，i从2开始遍历，到sqrt(n)(包含)，如果n能够被i整除就返回false，循环结束返回true

```
public static boolean isPrime(int n) {
    if (n <= 3) {
        return n > 1;
    }
    int sqrt = (int)Math.sqrt(n);
    for (int i = 2; i <= sqrt; i++) {
        if(n % i == 0) {
            return false;
        }
    }
    return true;
```

- 方法3，i从5开始遍历，到sqrt(n)(包含)，如果n能够被i整除或者被i+2整除就返回false，否则返回true，理由是素数总是等于6x+1或者6x-1;

```
public static boolean isPrime(int num) {
    if (num <= 3) {
        return num > 1;
    }
    // 不在6的倍数两侧的一定不是质数
    if (num % 6 != 1 && num % 6 != 5) {
        return false;
    }
    int sqrt = (int) Math.sqrt(num);
    for (int i = 5; i <= sqrt; i += 6) {
        if (num % i == 0 || num % (i + 2) == 0) {
            return false;
        }
    }
    return true;
}
```

## 5，Interger的方法

integer中有两个主要方法。

- parseInt---用于将字符串参数作为有符号的十进制整数进行解析，如果方法有两个参数，使用第二个参数指定的基数，将字符串参数解析为有符号的整数。简单说，就是将字符串处理成整数，第二个数字表示基，如果是2，就是2进制数字，如果是10，就是十进制数据
- valueOf---用于返回给定参数的原生numbr对象值，参数可以是原生数据类型，string等。该方法可以接受两个参数，一个是字符串，一个是基数
- parseInt返回的是基本类型int，valueOf返回的是包装类Integer。
- toBinaryString--获取数字的二进制表示，返回的数据类型是String
- toHexString--获取数字的十六进制表示，返回的数据类型是String

- toOctalString--获取数字的八进制表示，返回的数据类型是String

## 6，Stringbuilder方法

- append--用于向字符串生成器中追加内容
- insert---用于向字符串生产器中的指定位置插入数据内容，第一个位置是索引，第二个位置是要添加的字符串
- delete--移除此序列的子字符串中的字符，第一个参数start索引，第二个参数是end索引，左闭右开
- toString--返回字符串

## 7，set集合

- hashset类实现set接口，由哈希表支持，它不保证set的迭代顺序，允许使用null元素
- treeset类实线额set集合在遍历集合时按照自然顺序递增排序，也可以按照指定比较器递增排序
- add--将制定的对象添加到该集合中
- remove--将指定的对象从该集合中
- isEmpty---返回boolean值，用于判断当前集合是否为空
- iterator--返回迭代器，用于遍历集合中的对象
- size---返回int值，获取该集合中元素的个数
- hashset导入---import java.util.LinkedHashSet;
- hashset实例化--HashSet<Character> set = new LinkedHashSet<Character>();
- 迭代器导入---import java.util.Iterator
- 迭代器实例化--Iterator iter = set.Iterator()
- 迭代器判断---iter.hasNext()
- 迭代器迭代--iter.next()