|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 题号 | 题目 | 知识点 | 难度 | 通过率 |
|  |  |  |  |  |
| HJ1 | 字符串最后一个单词的长度 | 字符串 | 较难 | 27.03% |
| HJ2 | 计算字符个数 | 字符串哈希 | 较难 | 25.33% |
| HJ3 | 明明的随机数 | 数组 | 较难 | 18.04% |
| HJ4 | 字符串分隔 | 字符串 | 较难 | 24.14% |
| HJ5 | 进制转换 | 字符串 | 中等 | 27.50% |
| HJ6 | 质数因子 | 排序 | 中等 | 26.24% |
| HJ7 | 取近似值 | 数学 | 入门 | 43.65% |
| HJ8 | 合并表记录 |  | 中等 | 30.53% |
| HJ9 | 提取不重复的整数 | 数组哈希位运算 | 中等 | 36.40% |
| HJ10 | 字符个数统计 | 字符串哈希 | 中等 | 39.40% |
| HJ11 | 数字颠倒 | 字符串 | 简单 | 50.97% |
| HJ12 | 字符串反转 | 字符串 | 简单 | 51.22% |
| HJ13 | 句子逆序 | 数组 | 较难 | 32.89% |
| HJ14 | 字符串排序 | 字符串 | 中等 | 35.32% |
| HJ15 | 求int型数据在内存中存储时1的个数 | 位运算 | 入门 | 49.68% |
| HJ16 | 购物单 | 动态规划 | 中等 | 17.39% |
| HJ17 | 坐标移动 | 字符串 | 较难 | 18.94% |
| HJ18 | 识别有效的IP地址和掩码并进行分类统计 | 字符串 | 困难 | 19.10% |
| HJ19 | 简单错误记录 | 字符串 | 困难 | 17.24% |
| HJ20 | 密码验证合格程序 | 数组字符串 | 较难 | 24.28% |
| HJ21 | 简单密码破解 | 字符串 | 中等 | 35.16% |
| HJ22 | 汽水瓶 | 模拟 | 简单 | 24.28% |
| HJ23 | 删除字符串中出现次数最少的字符 | 字符串 | 较难 | 27.04% |
| HJ24 | 合唱队 | 动态规划队列 | 较难 | 22.77% |
| HJ25 | 数据分类处理 | 排序 | 较难 | 18.49% |
| HJ26 | 字符串排序 | 排序字符串 | 中等 | 30.65% |
| HJ27 | 查找兄弟单词 |  | 困难 | 16.42% |
| HJ28 | 素数伴侣 | 排序 | 困难 | 23.28% |
| HJ29 | 字符串加解密 | 字符串 | 较难 | 23.23% |
| HJ30 | 字符串合并处理 | 字符串排序 | 较难 | 23.39% |
| HJ31 | 【中级】单词倒排 | 字符串排序 | 困难 | 16.72% |
| HJ32 | 【中级】字符串运用-密码截取 | 字符串 | 较难 | 24.53% |
| HJ33 | 整数与IP地址间的转换 | 字符串 | 较难 | 26.06% |
| HJ34 | 图片整理 | 字符串 | 中等 | 38.91% |
| HJ35 | 蛇形矩阵 | 数组 | 中等 | 35.62% |
| HJ36 | 字符串加密 | 字符串 | 中等 | 34.71% |
| HJ37 | 统计每个月兔子的总数 | 排序 | 简单 | 35.73% |
| HJ38 | 求小球落地5次后所经历的路程和第5次反弹的高度 | 模拟 | 中等 | 29.53% |
| HJ39 | 判断两个IP是否属于同一子网 | 字符串模拟 | 较难 | 13.72% |
| HJ40 | 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 | 字符串 | 中等 | 36.27% |
| HJ41 | 称砝码 | 字符串 | 较难 | 34.48% |
| HJ42 | 学英语 | 字符串 | 较难 | 26.86% |
| HJ43 | 迷宫问题 | 排序 | 中等 | 31.06% |
| HJ44 | Sudoku-Java | 搜索数学 | 困难 | 24.70% |
| HJ45 | 名字的漂亮度 | 字符串 | 中等 | 30.33% |
| HJ46 | 按字节截取字符串 | 字符串 | 中等 | 38.15% |
| HJ47 | 线性插值 | 数组 | 较难 | 28.60% |
| HJ48 | 从单向链表中删除指定值的节点 | 链表 | 中等 | 29.04% |
| HJ49 | 多线程 | 字符串链表栈队列 | 中等 | 35.74% |
| HJ50 | 四则运算 | 字符串栈数学 | 简单 | 49.12% |
| HJ51 | 输出单向链表中倒数第k个结点 | 链表 | 较难 | 21.75% |
| HJ52 | 计算字符串的距离 | 字符串 | 中等 | 34.07% |
| HJ53 | iNOC产品部-杨辉三角的变形 | 字符串 | 简单 | 43.04% |
| HJ54 | 表达式求值 | 字符串数学栈 | 简单 | 49.58% |
| HJ55 | （练习用）挑7 | 穷举数学 | 中等 | 36.86% |
| HJ56 | iNOC产品部--完全数计算 | 数学 | 简单 | 39.87% |
| HJ57 | 无线OSS－高精度整数加法 | 字符串 | 中等 | 35.65% |
| HJ58 | 输入n个整数，输出其中最小的k个 | 数组 | 较难 | 28.02% |
| HJ59 | 找出字符串中第一个只出现一次的字符 | 字符串 | 中等 | 26.39% |
| HJ60 | 查找组成一个偶数最接近的两个素数 | 穷举数学 | 中等 | 37.78% |
| HJ61 | 放苹果 | 递归动态规划 | 简单 | 40.52% |
| HJ62 | 查找输入整数二进制中1的个数 | 位运算 | 简单 | 43.04% |
| HJ63 | DNA序列 | 字符串 | 中等 | 35.22% |
| HJ64 | MP3光标位置 | 数组 | 中等 | 28.22% |
| HJ65 | 查找两个字符串a,b中的最长公共子串 | 字符串 | 中等 | 31.86% |
| HJ66 | 配置文件恢复 | 字符串 | 简单 | 38.08% |
| HJ67 | 24点游戏算法 | 搜索dfs | 中等 | 32.72% |
| HJ68 | 成绩排序 | 排序 | 较难 | 21.73% |
| HJ69 | 矩阵乘法 | 数组 | 中等 | 37.57% |
| HJ70 | 矩阵乘法计算量估算 | 字符串 | 中等 | 36.07% |
| HJ71 | 字符串通配符 | 字符串 | 中等 | 36.15% |
| HJ72 | 百钱买百鸡问题 |  | 简单 | 39.34% |
| HJ73 | 计算日期到天数转换 | 字符串 | 简单 | 36.96% |
| HJ74 | 参数解析 | 字符串 | 简单 | 38.83% |
| HJ75 | 公共字串计算 | 字符串 | 简单 | 36.15% |
| HJ76 | 尼科彻斯定理 | 数学 | 简单 | 39.10% |
| HJ77 | 火车进站 | 栈 | 较难 | 38.60% |
| HJ80 | 整形数组合并 | 数组排序 | 较难 | 27.81% |
| HJ81 | 字符串匹配 | 字符串 | 较难 | 25.83% |
| HJ82 | 将真分数分解为埃及分数 | 搜索数学 | 困难 | 36.02% |
| HJ83 | 二维数组操作 | 数组 | 简单 | 24.16% |
| HJ84 | 统计大写字母个数 | 字符串 | 简单 | 36.75% |
| HJ85 | 字符串运用-密码截取 | 字符串穷举 | 简单 | 35.36% |
| HJ86 | 求最大连续bit数 | 位运算 | 简单 | 36.82% |
| HJ87 | 密码强度等级 | 字符串 | 简单 | 29.35% |
| HJ88 | 扑克牌大小 | 字符串队列链表栈 | 中等 | 31.52% |
| HJ89 | 24点运算 | 字符串模拟穷举 | 困难 | 15.53% |
| HJ90 | 合法IP | 字符串链表栈队列 | 中等 | 31.33% |
| HJ91 | 201301 JAVA 题目2-3级 | 字符串 | 简单 | 37.40% |
| HJ92 | 在字符串中找出连续最长的数字串 | 字符串 | 较难 | 24.70% |
| HJ93 | 201301 JAVA题目0-1级 | 字符串递归 | 较难 | 30.02% |
| HJ94 | 记票统计 |  | 较难 | 23.20% |
| HJ95 | 人民币转换 | 字符串 | 较难 | 27.10% |
| HJ96 | 表示数字 | 字符串 | 中等 | 28.45% |
| HJ97 | 记负均正 | 数组 | 较难 | 17.06% |
| HJ98 | 自动售货系统 | 字符串 | 中等 | 23.17% |
| HJ99 | 自守数 |  | 中等 | 33.56% |
| HJ100 | 等差数列 | 数学 | 简单 | 34.53% |
| HJ101 | 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 | 排序 | 较难 | 22.97% |
| HJ102 | 字符统计 | 字符串排序 | 中等 | 25.49% |
| HJ103 | Redraiment的走法 | 排序 | 中等 | 23.79% |
| HJ105 | 记负均正II | 数组 | 较难 | 23.57% |
| HJ106 | 字符逆序 | 字符串 | 简单 | 39.98% |
| HJ107 | 求解立方根 | 数学二分 | 中等 | 27.75% |
| HJ108 | 求最小公倍数 | 递归数学 | 简单 | 32.77% |

目录

[HJ1 字符串最后一个单词的长度 4](#_Toc6804)

[HJ2 计算字符个数 4](#_Toc28746)

[HJ3 明明的随机数 4](#_Toc7668)

[HJ4 字符串分隔 5](#_Toc13124)

[HJ5 进制转换 6](#_Toc22615)

[HJ6 质数因子 6](#_Toc14920)

[HJ7 取近似值 7](#_Toc24681)

[HJ8 合并表记录 7](#_Toc3032)

[HJ9 提取不重复的整数 8](#_Toc27397)

[HJ10 字符个数统计 9](#_Toc14401)

[HJ11 数字颠倒 9](#_Toc8888)

[HJ12 字符串反转 10](#_Toc30083)

[HJ13 句子逆序 10](#_Toc27774)

[HJ14 字符串排序 11](#_Toc1957)

[HJ15 求int型数据在内存中存储时1的个数 12](#_Toc2908)

[HJ16 购物单 13](#_Toc16483)

[HJ17 坐标移动 14](#_Toc25019)

[HJ18 识别有效的IP地址和掩码并进行分类统计 15](#_Toc31131)

[HJ19 简单错误记录 17](#_Toc18100)

[HJ20 密码验证合格程序 18](#_Toc18410)

[HJ21 简单密码破解 19](#_Toc22194)

[HJ22 汽水瓶 19](#_Toc20304)

[HJ23 删除字符串中出现次数最少的字符 20](#_Toc8711)

[HJ24 合唱队 21](#_Toc12759)

[HJ25 数据分类处理 22](#_Toc29049)

[HJ26 字符串排序 23](#_Toc4239)

[HJ27 查找兄弟单词 24](#_Toc6501)

[HJ28 素数伴侣 24](#_Toc16559)

[HJ29 字符串加解密 26](#_Toc12435)

[HJ30 字符串合并处理 27](#_Toc28946)

[HJ31 【中级】单词倒排 28](#_Toc31426)

[HJ32 【中级】字符串运用-密码截取 29](#_Toc18712)

[HJ33 整数与IP地址间的转换 30](#_Toc29977)

[HJ34 图片整理 30](#_Toc24702)

[HJ35 蛇形矩阵 31](#_Toc29378)

[HJ36 字符串加密 32](#_Toc5767)

[HJ37 统计每个月兔子的总数 33](#_Toc21433)

[HJ38 求小球落地5次后所经历的路程和第5次反弹的高度 33](#_Toc14870)

[HJ39 判断两个IP是否属于同一子网 34](#_Toc9941)

[HJ40 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 35](#_Toc21505)

[HJ41 称砝码 36](#_Toc14709)

[HJ42 学英语 37](#_Toc11424)

[HJ43 迷宫问题 38](#_Toc26935)

[HJ44 Sudoku-Java 39](#_Toc12063)

[HJ45 名字的漂亮度 40](#_Toc28819)

[HJ46 按字节截取字符串 41](#_Toc8226)

[HJ47 线性插值 41](#_Toc28006)

[HJ48 从单向链表中删除指定值的节点 43](#_Toc2740)

[HJ49 多线程 44](#_Toc6312)

[HJ50 四则运算 45](#_Toc7588)

[HJ51 输出单向链表中倒数第k个结点 46](#_Toc8409)

[HJ52 计算字符串的距离 47](#_Toc17584)

[HJ53 iNOC产品部-杨辉三角的变形 48](#_Toc25846)

[HJ54 表达式求值 49](#_Toc25654)

[HJ55 （练习用）挑7 50](#_Toc31727)

[HJ56 iNOC产品部--完全数计算 50](#_Toc4932)

[HJ57 无线OSS－高精度整数加法 51](#_Toc5367)

[HJ58 输入n个整数，输出其中最小的k个 53](#_Toc20277)

[HJ59 找出字符串中第一个只出现一次的字符 54](#_Toc24358)

[HJ60 查找组成一个偶数最接近的两个素数 54](#_Toc15687)

[HJ61 放苹果 55](#_Toc23354)

[HJ62 查找输入整数二进制中1的个数 55](#_Toc29131)

[HJ63 DNA序列 56](#_Toc20621)

[HJ64 MP3光标位置 57](#_Toc12761)

[HJ65 查找两个字符串a,b中的最长公共子串 58](#_Toc9838)

[HJ66 配置文件恢复 59](#_Toc2997)

[HJ67 24点游戏算法 60](#_Toc11644)

[HJ68 成绩排序 61](#_Toc17222)

[HJ69 矩阵乘法 62](#_Toc189)

[HJ70 矩阵乘法计算量估算 63](#_Toc15177)

[HJ71 字符串通配符 64](#_Toc9346)

[HJ72 百钱买百鸡问题 66](#_Toc2349)

[HJ73 计算日期到天数转换 66](#_Toc31067)

[HJ74 参数解析 67](#_Toc4279)

[HJ75 公共字串计算 67](#_Toc2975)

[HJ76 尼科彻斯定理 68](#_Toc11745)

[HJ77 火车进站 69](#_Toc19550)

[HJ80 整形数组合并 70](#_Toc26255)

[HJ81 字符串匹配 71](#_Toc26291)

[HJ82 将真分数分解为埃及分数 72](#_Toc11023)

[HJ83 二维数组操作 74](#_Toc15568)

[HJ84 统计大写字母个数 75](#_Toc9091)

[HJ85 字符串运用-密码截取 76](#_Toc12855)

[HJ86 求最大连续bit数 77](#_Toc14753)

[HJ87 密码强度等级 77](#_Toc21762)

[HJ88 扑克牌大小 79](#_Toc18799)

[HJ89 24点运算 81](#_Toc17963)

[HJ90 合法IP 83](#_Toc30947)

[HJ91 201301 JAVA 题目2-3级 83](#_Toc292)

[HJ92 在字符串中找出连续最长的数字串 84](#_Toc30102)

[HJ93 201301 JAVA题目0-1级 85](#_Toc25935)

[HJ94 记票统计 87](#_Toc4324)

[HJ95 人民币转换 88](#_Toc14355)

[HJ96 表示数字 89](#_Toc13870)

[HJ97 记负均正 90](#_Toc19606)

[HJ98 自动售货系统 91](#_Toc21536)

[HJ99 自守数 95](#_Toc9842)

[HJ100 等差数列 96](#_Toc22816)

[HJ101 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 96](#_Toc22232)

[HJ102 字符统计 97](#_Toc9307)

[HJ103 Redraiment的走法 98](#_Toc4092)

[HJ105 记负均正II 99](#_Toc6153)

[HJ106 字符逆序 99](#_Toc2247)

[HJ107 求解立方根 100](#_Toc7888)

[HJ108 求最小公倍数 101](#_Toc23320)

# HJ1 字符串最后一个单词的长度

**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:**

输入一行，代表要计算的字符串，非空，长度小于5000。

**输出描述:**

输出一个整数，表示输入字符串最后一个单词的长度。

示例1

**输入**

hello nowcoder

**输出**

8

import java.io.InputStream;

public class Main {

public static void main(String[] args) throws Exception{

int length = 0;

InputStream inputStream = System.in;

char c = (char)inputStream.read();

while(c!='\n'){

if(c==' '){

length =0;

}else{

length +=1;

}

c= (char)inputStream.read();

}

System.out.print(length);

}

}

}

# HJ2 计算字符个数

**题目描述**

写出一个程序，接受一个由字母、数字和空格组成的字符串，和一个字母，然后输出输入字符串中该字母的出现次数。不区分大小写。

**输入描述:**

第一行输入一个由字母和数字以及空格组成的字符串，第二行输入一个字母。

**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**

ABCabc

A

**输出**

2

import java.io.\*;

public class Main {

public static void main(String[] args) throws IOException {

int count = 0;

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

char[] chars1 = bufferedReader.readLine().toLowerCase().toCharArray();

char[] chars2 = bufferedReader.readLine().toLowerCase().toCharArray();

char c1 = chars2[0];

for(char c : chars1){

if(c == c1){

count++;

}

}

System.out.println(count);

}

}

# HJ3 明明的随机数

**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(用于不同的调查)，希望大家能正确处理)。

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。

当没有新的输入时，说明输入结束。

**输入描述:**

注意：输入可能有多组数据(用于不同的调查)。每组数据都包括多行，第一行先输入随机整数的个数N，接下来的N行再输入相应个数的整数。具体格式请看下面的"示例"。

**输出描述:**

返回多行，处理后的结果

示例1

**输入**

3

2

2

1

11

10

20

40

32

67

40

20

89

300

400

15

**输出**

1

2

10

15

20

32

40

67

89

300

400

**说明**

样例输入解释：

样例有两组测试

第一组是3个数字，分别是：2，2，1。

第二组是11个数字，分别是：10，20，40，32，67，40，20，89，300，400，15。

import java.io.\*;

public class Main {

public static void main(String[] args) throws Exception{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str;

while ((str = br.readLine()) != null) {

boolean[] stu = new boolean[1001];

StringBuilder sb = new StringBuilder();

int n = Integer.valueOf(str);

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

stu[Integer.valueOf(br.readLine())] = true;

for (int i = 1; i < 1001; i++) {

if (stu[i]) {

sb.append(i).append("\n");

}

}

sb.deleteCharAt(sb.length() - 1);

System.out.println(sb.toString());

}

}

}

# HJ4 字符串分隔

**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(输入多次,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

import java.io.\*;

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 len = str.length();

int st = 0;

while(len>=8){

System.out.println(str.substring(st,st+8));

st+=8;

len-=8;

}

if(len>0){

char[] tmp = new char[8];

for(int i = 0;i<8;i++){

tmp[i]='0';

}

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

tmp[i]=str.charAt(st++);

}

System.out.println(String.valueOf(tmp));

}

}

}

}

# HJ5 进制转换

**题目描述**

写出一个程序，接受一个十六进制的数，输出该数值的十进制表示。

**输入描述:**

输入一个十六进制的数值字符串。注意：一个用例会同时有多组输入数据，请参考帖子https://www.nowcoder.com/discuss/276处理多组输入的问题。

**输出描述:**

输出该数值的十进制字符串。不同组的测试用例用\n隔开。

示例1

**输入**

0xA

0xAA

**输出**

10

170

import java.io.\*;

public class Main {

public static void main(String[] args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String line = null;

while((line = br.readLine()) != null){

System.out.println(Long.parseLong(line.substring(2) , 16));

}

}

}

# HJ6 质数因子

**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因子（重复的也要列举）（如180的质因子为2 2 3 3 5 ）

最后一个数后面也要有空格

**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

import java.io.\*;

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 num = Integer.parseInt(str);

StringBuffer sb = new StringBuffer();

//Math.sqrt开方

for (int i = 2; i <= Math.sqrt(num); i++) {

if (num % i == 0) {

sb.append(i).append(" ");

num = num / i;

i--;

}

}

sb.append(num + " ");

System.out.println(sb.toString());

}

}

}

# HJ7 取近似值

**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

import java.io.\*;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String s = br.readLine();

int index = s.indexOf(".");

int a =Integer.parseInt(s.substring(0,index));

int b=Integer.parseInt(s.substring(index+1,index+2));

if (b >=5) {

System.out.println(a+1);

}else {

System.out.println(a);

}

}

}

# HJ8 合并表记录

**题目描述**

数据表记录包含表索引和数值（int范围的正整数），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

解法一：

import java.io.\*;

public class Main{

public static void main(String[] args) throws IOException {

StreamTokenizer st = new StreamTokenizer(new BufferedReader(new InputStreamReader(System.in)));

st.nextToken();

int n = (int)st.nval;

int[] arr = new int[n];

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

st.nextToken();

int key = (int)st.nval;

st.nextToken();

int value = (int)st.nval;

arr[key] = arr[key]+value;

}

StringBuilder sb = new StringBuilder();

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

if(arr[i]!=0){

sb.append(i).append(" ").append(arr[i]).append("\n");

}

}

System.out.println(sb.toString());

}

}

解法二：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.HashMap;

import java.util.Map;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String line = null;

Map<Integer, Integer> map = new HashMap<>();

line = br.readLine();

int[] ints = new int[Integer.valueOf(line)];

while ((line =br.readLine()) !=null){

int a = line.indexOf(" ");

int b = line.indexOf(" ", a);

Integer c = Integer.valueOf(line.substring(0, a));

Integer d = Integer.valueOf(line.substring(a + 1));

ints[c] = ints[c]+d;

}

StringBuilder sb = new StringBuilder();

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

if (ints[i]!=0){

sb.append(i).append(" ").append(ints[i]).append("\n");

}

}

sb.deleteCharAt(sb.length()-1);

System.out.println(sb.toString());

}

}

解法三：

public static class Main{

public static void main(String[] args) throws IOException{

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String in;

while((in=reader.readLine())!=null){

int num = Integer.parseInt(in);

int[] arr = new int[1000];

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

String input = reader.readLine();

int index = input.indexOf(' ');

int key =Integer.parseInt(input.substring(0,index));

int value = Integer.parseInt(input.substring(index+1));

arr[key]+=value;

}

StringBuilder out=new StringBuilder();

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

if(arr[i]>0){

out.append(i).append(' ').append(arr[i]).append('\n');

}

}

System.out.println(out);

}

}

}

# HJ9 提取不重复的整数

**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。

保证输入的整数最后一位不是0。

**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

import java.io.\*;

import java.util.\*;

public class Main{

public static void main(String args[]) throws Exception{

InputStream in = System.in;

byte[] b = new byte[1024];

int len = 0;

while((len = in.read(b))>0){

String str = new String(b,0,len-1);

char[] chars = str.toCharArray();

String result = "";

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

if(result.indexOf(chars[i])<0){

result+=(chars[i]);

}

}

System.out.println(result);

}

}

}

# HJ10 字符个数统计

[字符个数统计](javascript:void(0);)

**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)，换行表示结束符，不算在字符里。不在范围内的不作统计。多个相同的字符只计算一次。

例如，对于字符串abaca而言，有a、b、c三种不同的字符，因此输出3。

**输入描述:**

输入一行没有空格的字符串。

**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

import java.util.\*;

import java.io.\*;

public class Main{

public static void main(String[]args)throws Exception{

BufferedReader shuru = new BufferedReader(new InputStreamReader(System.in));

String s = shuru.readLine();

int[] a = new int[128];

int count=0;

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

char b = s.charAt(i);

if(a[b]==0){

count++;

a[b]=1;

}

}

System.out.println(count);

}

}

# HJ11 数字颠倒

[数字颠倒](javascript:void(0);)

**题目描述**

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

import java.io.\*;

public class Main {

public static void main(String[] args) throws IOException{

InputStream stream = System.in;

byte[] bytes = new byte[1024];

int l = 0;

StringBuffer strBuffer = new StringBuffer();

while((l = stream.read(bytes)) > 0){

String str = new String(bytes,0,l-1);

strBuffer = new StringBuffer(str);

}

System.out.print(strBuffer.reverse());

}

}

# HJ12 字符串反转

字符串反转

**题目描述**

接受一个只包含小写字母的字符串，然后输出该字符串反转后的字符串。（字符串长度不超过1000）

**输入描述:**

输入一行，为一个只包含小写字母的字符串。

**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

import java.io.\*;

public class Main{

public static void main(String[] args) throws IOException{

InputStream in = System.in;

int len;

byte[] b = new byte[1024];

while((len = in.read(b)) > 0){

String str = new String(b,0,len-1);

char [] chars = str.toCharArray();

char [] charsFb = new char[chars.length];

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

charsFb[i] = chars[chars.length -1 -i];

}

System.out.println(new String(charsFb));

}

}

}

# HJ13 句子逆序

[句子逆序](javascript:void(0);)

**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

**输入描述:**

输入一个英文语句，每个单词用空格隔开。保证输入只包含空格和字母。

**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

解法一：

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class Main{

public static void main(String[] args) throws Exception{

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

String str = in.readLine();

char[] chars = (str+" ").toCharArray();

int length = str.length();

char[] newchars = new char[length];

int start = 0;

int end = length;

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

if(chars[i]==' '){

int \_length = i-start;

int poit = end - \_length;

System.arraycopy(chars, start, newchars, poit, \_length);

start=i+1;

length = length - \_length -1;

end = length;

if(end>0){

newchars[end]=' ';

}

}

}

System.out.println(new String(newchars));

}

}

解法二：

import sun.security.util.Length;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.LinkedHashMap;

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){

char[] strArr = str.toCharArray();

recerseString(strArr,0,strArr.length - 1);

reverseWord(strArr);

System.out.println(new String(strArr));

}

}

private static void recerseString (char[] str, int start, int end){

int i = start;

int j = end;

while(i < j){

char t = str[i];

str[i] = str[j];

str[j] = t;

i++;

j--;

}

}

private static void reverseWord(char[] str){

int i = 0;

int j = 0;

while(j < str.length){

while( j < str.length && str[j] != ' '){

j++;

}

if(j != 0){

recerseString(str, i, j - 1);

}

while(j < str.length && str[j] == ' '){

j++;

}

i = j;

}

}

}

解法三：

import java.util.\*;

public class Main{

public static void main(String[] agrs){

Scanner in = new Scanner(System.in);

String input = in.nextLine();

System.out.println(getResult(input));

}

public static String getResult(String input) {

StringBuilder sb = new StringBuilder();

Stack<String> stack = new Stack<>();

String[] arr = input.split("\\s");

for (String str : arr) {

stack.push(str);

}

while (!stack.isEmpty()) {

sb.append(stack.pop() + " ");

}

return sb.toString();

}

}

# HJ14 字符串排序

[字符串排序](javascript:void(0);)

**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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));

int count = Integer.parseInt(bf.readLine());

String[] arr = new String[count];

for (int i = 0; i < count; i++) arr[i] = bf.readLine();

StringBuilder sb = new StringBuilder();

Arrays.sort(arr);

for (String w : arr) sb.append(w).append('\n');

System.out.println(sb.toString());

}

}

# HJ15 求int型数据在内存中存储时1的个数

[求int型正整数在内存中存储时1的个数](javascript:void(0);)

**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

import java.io.InputStream;

public class Main {

public static void main(String[] args) throws Exception {

InputStream stream = System.in;

int l ;

byte[] bytes = new byte[1024];

while ((l = stream.read(bytes)) > 0) {

String numStr = new String(bytes, 0, l - 1);

int num = Integer.parseInt(numStr);

char [] numChars = Integer.toBinaryString(num).toCharArray();

int countNum = 0;

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

if (numChars[i] == '1'){

countNum = countNum + 1;

}

}

System.out.println(countNum);

}

}

}

# HJ16 购物单

[购物单](javascript:void(0);)

**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

import java.io.\*;

public class Main{

public static int dw = 100;

public static void main(String[] args)throws IOException{

boolean flag = true;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

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

int N = Integer.parseInt(firstArr[0]);

int m = Integer.parseInt(firstArr[1]);

Good[] A = new Good[m+1];

for(int i = 1;i<=m;i++){

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

int v = Integer.parseInt(perArr[0]);

int p = Integer.parseInt(perArr[1]);

int q = Integer.parseInt(perArr[2]);

if(flag)

if(v%dw!=0){

flag = false;

dw = 10;

for(int j = 1;j<i;j++){

A[j].v = A[j].v\*dw;

}

}

v=v/dw;

A[i] = new Good(v,p,q);

if(q>0){

if(A[q].a1 == 0){

A[q].a1=i;

}else{

A[q].a2=i;

}

}

}

N=N/dw;

check(N,A);

}

public static void check(int N,Good[] A ){

int[][] dp = new int[A.length][N+1];

for(int i=1;i<A.length;i++){

int v=-1,v1=-1,v2=-1,v3=-1,tmpDp=-1,tmpDp1=-1,tmpDp2=-1,tmpDp3=-1;

v = A[i].v;//主件

tmpDp = v\*A[i].p;

if(A[i].a2!=0&&A[i].a1!=0){//主件+附件1+附件2

v3 = v+A[A[i].a2].v+A[A[i].a1].v;

tmpDp3 = tmpDp + A[A[i].a2].v\*A[A[i].a2].p+ A[A[i].a1].v\*A[A[i].a1].p;

}

if(A[i].a1!=0){//主件+附件1

v1 = v+A[A[i].a1].v;

tmpDp1 = tmpDp + A[A[i].a1].v\*A[A[i].a1].p;

}

if(A[i].a2!=0){//主件+附件2

v2 = v+A[A[i].a2].v;

tmpDp2 = tmpDp + A[A[i].a2].v\*A[A[i].a2].p;

}

for(int j = i;j<=N;j++){

dp[i][j] = dp[i-1][j];

if(A[i].q==0){//是主件

if(j>=v&&v!=-1)dp[i][j] = Math.max(dp[i][j],dp[i-1][j-v]+tmpDp);

if(j>=v1&&v1!=-1)dp[i][j] = Math.max(dp[i][j],dp[i-1][j-v1]+tmpDp1);

if(j>=v2&&v2!=-1)dp[i][j] = Math.max(dp[i][j],dp[i-1][j-v2]+tmpDp2);

if(j>=v3&&v3!=-1)dp[i][j] = Math.max(dp[i][j],dp[i-1][j-v3]+tmpDp3);

}

}

}

System.out.print(dp[A.length-1][N]\*dw);

}

}

class Good{

public int v;//价值

public int p;//重要度

public int q;//是否是附件，>0附件，=0主件

public int a1;//附件1编号

public int a2;//附件2编号

public Good(int v,int p,int q){

this.v = v;

this.p = p;

this.q = q;

}

}

# HJ17 坐标移动

[坐标移动](javascript:void(0);)

**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）

注意请处理多组输入输出

**输入描述:**

一行字符串

**输出描述:**

最终坐标，以逗号分隔

示例1

**输入**

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

**输出**

10,-10

import java.io.\*;

public class Main{

public static void main(String[] args ) throws Exception{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String s;

while((s = br.readLine()) != null){

String[] str = s.split(";");

int x = 0;

int y = 0;

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

int change = 0;

char[] cha = str[i].toCharArray();

for(int j = 1;j <cha.length;j++){

if(cha[j] >= '0' &&cha[j] <= '9'){

change = change\*10 +(cha[j]-'0');

}else{

break;

}

}

switch (cha[0]){

case 'A' :

x -= change;

break;

case 'D' :

x += change;

break;

case 'S':

y -= change;

break;

case 'W':

y += change;

break;

}

}

System.out.println(x+","+y);

}

}

}

# HJ18 识别有效的IP地址和掩码并进行分类统计

[识别有效的IP地址和掩码并进行分类统计](javascript:void(0);)

**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）

注意二进制下全是1或者全是0均为非法

注意：

1. 类似于【0.\*.\*.\*】和【127.\*.\*.\*】的IP地址不属于上述输入的任意一类，也不属于不合法ip地址，计数时可以忽略

2. 私有IP地址和A,B,C,D,E类地址是不冲突的

**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

解法一：

import java.io.\*;

public class Main{

public static void main(String[] args)throws IOException{

BufferedReader bf=new BufferedReader(new InputStreamReader(System.in));

int a=0,b=0,c=0,d=0,e=0,err=0,pri=0;

String str;

while((str=bf.readLine())!=null){

int index=str.indexOf('~');

long ip=ipToNum(str.substring(index+1));

if(ip<=0||ip>=0XFFFFFFFFL||(((ip ^ 0XFFFFFFFFL) + 1) | ip) != ip){ err+=1; continue;}

ip=ipToNum(str.substring(0,index)); long partOfIP=ip>>24;

if(ip<=0L||ip>=0XFFFFFFFFL){ err+=1;

}else if(partOfIP>=1&&partOfIP<=126){ a+=1; if(partOfIP==10){ pri+=1;}

}else if(partOfIP>=128&&partOfIP<=191){ b+=1; if(ip>>20==0xAC1){ pri+=1; }

}else if(partOfIP>=192&&partOfIP<=223){ c+=1; if(ip>>16==0xC0A8){ pri+=1; }

}else if(partOfIP>=224&&partOfIP<=239){ d+=1;}else if(partOfIP>=240&&partOfIP<=255){ e+=1; }

}

System.out.println(a+" "+b+" "+c+" "+d+" "+e+" "+err+" "+pri);

}

public static long ipToNum(String ip){

long num=0;

int flag=1,temp=0;

char[] chars=ip.toCharArray();

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

if(chars[i]=='.'){ num=num<<8|temp; temp=0;flag++; continue;}

if(flag>=2){ return -1;}

if(chars[i]>='0'&&chars[i]<='9'){

temp=temp\*10+chars[i]-'0';

flag=0;

}else{

flag=3; break;

}

}

if(flag>=2){

return -1;

}

num=num<<8|temp;

return num;

}

}

解法二：

public static class Main {

/\*\*

\* 该代码在牛客网上通过，在idea中无法运行

\* 先判断子网掩码是否正确，方法是将子网掩码转换成32位0/1组成的二进制字符串，那么怎样判断掩码合法呢，只需要判断该二进制中第一个0和最后一个1的位置即可，只要第一个0在的位置比最后一个1的位置较前则掩码非法，反则合法；然后如果掩码合法，接着判断IP是否合法

\*/

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int isA = 0;

int isB = 0;

int isC = 0;

int isD = 0;

int isE = 0;

int isEroor = 0;

int isPrivate = 0;

while (sc.hasNextLine()) {

String str = sc.nextLine();

String[] ips = str.split("~");

//判断掩码是否合法

boolean isMask = false;

String[] maskArr = ips[1].split("\\.");

StringBuilder binaryMask = new StringBuilder();

//形成掩码二进制字符串

for (String s : maskArr) {

StringBuilder binaryStr = new StringBuilder(Integer.toBinaryString(Integer.parseInt(s)));

//凑成4组8位二进制

for (int j = 0; j < 8 - binaryStr.length(); j++) {

binaryStr.insert(0, "0");

}

binaryMask.append(binaryStr);

}

//比较二进制字符串中第一个0的位置和最后一个1的位置来判断掩码是否合法

isMask = binaryMask.indexOf("0") >= binaryMask.lastIndexOf("1");

//掩码合法后再来判断IP是否合法

if (isMask) {

String[] strArr = ips[0].split("\\.");

if (strArr.length == 4) {

int[] ipArr = new int[4];

for (int i = 0; i < 4; i++) {

if (strArr[i].equals("")) {

ipArr[i] = -1;

} else {

ipArr[i] = Integer.parseInt(strArr[i]);

}

}

if ((0 <= ipArr[0] && ipArr[0] <= 255) && (0 <= ipArr[1] && ipArr[1] <= 255) && (0 <= ipArr[2] && ipArr[2] <= 255) && (0 <= ipArr[3] && ipArr[3] <= 255)) {

if (ipArr[0] >= 1 && ipArr[0] <= 126) {

isA++;

if (ipArr[0] == 10)

isPrivate++;

} else if (ipArr[0] >= 128 && ipArr[0] <= 191) {

isB++;

if (ipArr[0] == 172 && (ipArr[1] >= 16 && ipArr[1] <= 31))

isPrivate++;

} else if (ipArr[0] >= 192 && ipArr[0] <= 223) {

isC++;

if (ipArr[0] == 192 && ipArr[1] == 168)

isPrivate++;

} else if (ipArr[0] >= 224 && ipArr[0] <= 239) {

isD++;

} else if (ipArr[0] >= 240) {

isE++;

}

} else {

isEroor++;

}

} else {

isEroor++;

}

} else {

isEroor++;

}

}

System.out.println(isA + " " + isB + " " + isC + " " + isD + " " + isE + " " + isEroor + " " + isPrivate);

sc.close();

}

}

# HJ19 简单错误记录

[简单错误记录](javascript:void(0);)

**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，最后只用输出最后出现的八条错误记录。对相同的错误记录只记录一条，但是错误计数增加。最后一个斜杠后面的带后缀名的部分（保留最后16位）和行号完全匹配的记录才做算是”相同“的错误记录。

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。

4、循环记录时，只以第一次出现的顺序为准，后面重复的不会更新它的出现时间，仍以第一次为准

**输入描述:**

每组只包含一个测试用例。一个测试用例包含一行或多行字符串。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**

D:\zwtymj\xccb\ljj\cqzlyaszjvlsjmkwoqijggmybr 645

E:\je\rzuwnjvnuz 633

C:\km\tgjwpb\gy\atl 637

F:\weioj\hadd\connsh\rwyfvzsopsuiqjnr 647

E:\ns\mfwj\wqkoki\eez 648

D:\cfmwafhhgeyawnool 649

E:\czt\opwip\osnll\c 637

G:\nt\f 633

F:\fop\ywzqaop 631

F:\yay\jc\ywzqaop 631

**输出**

rzuwnjvnuz 633 1

atl 637 1

rwyfvzsopsuiqjnr 647 1

eez 648 1

fmwafhhgeyawnool 649 1

c 637 1

f 633 1

ywzqaop 631 2

import java.io.\*;

import java.util.\*;

public class Main{

public static void main(String [] args) throws Exception{

BufferedReader buffer = new BufferedReader(new InputStreamReader(System.in));

String str;

LinkedHashMap<String,Integer> data = new LinkedHashMap<String,Integer>();

while((str = buffer.readLine())!=null){

int idx1 = str.lastIndexOf(" ");

int idx2 = str.lastIndexOf("\\");

String key = (idx1-idx2)>16?str.substring(idx1-16):str.substring(idx2+1);

data.put(key,data.getOrDefault(key,0)+1);

}

int count=0;

for (String key:data.keySet()){

count++;

if(count>(data.size()-8)){

System.out.println(key+" "+data.get(key));

}

}

}

}

# HJ20 密码验证合格程序

[密码验证合格程序](javascript:void(0);)

**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.不能有相同长度大于2的子串重复

**输入描述:**

一组或多组长度超过2的字符串。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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

// 密码验证合格程序

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String input = null;

StringBuffer sb = new StringBuffer();

while (null != (input = reader.readLine())) {

//设置四种类型数据初始为空即false，有数据了就更改为true

boolean[] flag = new boolean[4];

char[] chars = input.toCharArray();

// 第一个条件

if (chars.length < 9) {

sb.append("NG").append("\n");

continue;

}

// 第二个条件

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

if ('A' <= chars[i] && chars[i] <= 'Z') {

flag[0] = true;

} else if ('a' <= chars[i] && chars[i] <= 'z') {

flag[1] = true;

} else if ('0' <= chars[i] && chars[i] <= '9') {

flag[2] = true;

} else {

flag[3] = true;

}

}

int count = 0;

for (int i = 0; i < 4; i++) {

if (flag[i]) {

count++;

}

}

// 第三个条件

boolean sign = true; //不存在两个大于2的子串相同，即！（i=i+3,i+1=i+4,i+2=i+5）

for (int i = 0; i < chars.length - 5; i++) {

for (int j = i + 3; j < chars.length - 2; j++) {

if (chars[i] == chars[j] && chars[i + 1] == chars[j + 1] && chars[i + 2] == chars[j + 2]) {

sign = false;

}

}

}

if (count >= 3 && sign) {

sb.append("OK").append("\n");

} else {

sb.append("NG").append("\n");

}

}

System.out.println(sb.toString());

}

}

# HJ21 简单密码破解

[简单密码](javascript:void(0);)

**题目描述**

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

import java.io.\*;

import java.util.\*;

public class Main{

public static void main(String args[]) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str1 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";

String str2 = "bcdefghijklmnopqrstuvwxyza222333444555666777788899990123456789";

String a = br.readLine();

char[] c= a.toCharArray();

StringBuilder sb = new StringBuilder();

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

for(int j = 0; j<str1.length(); j++){

if(c[i] == str1.charAt(j)){

sb.append(str2.charAt(j));

break;

}

}

}

System.out.println(sb.toString());

}

}

# HJ22 汽水瓶

[汽水瓶](javascript:void(0);)

**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

import java.io.\*;

public class Main{

public static void main(String[] args)throws Exception{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

while((str = br.readLine())!= null){

int g = 0;

g = Integer.parseInt(str);

if(g == 0){

return;

}

int count = 0;

while (g != 0){

int f = g/3;

count+=f;

g =g%3 +f;

if(g < 3){

break;

}

}

if(g==2){

count+=1;

}

System.out.println(count);

}

}

}

# HJ23 删除字符串中出现次数最少的字符

[删除字符串中出现次数最少的字符](javascript:void(0);)

**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。

注意每个输入文件有多组输入，即多个字符串用回车隔开

**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

aabcddd

**输出**

dd

aaddd

import java.io.\*;

public class Main{

public static void main(String args[]) throws Exception{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

while((str=br.readLine())!=null){

int num[] = new int[26];

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

for(int j=0;j<26;j++){

if((int)(str.charAt(i)-'a')==j)

num[j]++;

}

}

int min = Integer.MAX\_VALUE;

for(int i=0;i<26;i++){

if(num[i]!=0&&num[i]<min)

min=num[i];

}

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

if(num[str.charAt(i)-'a']!=min)

System.out.print(str.charAt(i));

}

System.out.println();

}

}

}

# HJ24 合唱队

[合唱队](javascript:void(0);)

**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-1<Ti>Ti+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。

注意不允许改变队列元素的先后顺序

请注意处理多组输入输出！

**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

import java.io.\*;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

while ((str = br.readLine()) != null) {

if (str.equals("")) continue;

int n = Integer.parseInt(str);

int[] heights = new int[n];

String[] str\_heights = br.readLine().split(" ");

// 当仅有一个人时，其自己组成一个合唱队，出列0人

if (n <= 1) System.out.println(0);

else {

for (int i = 0; i < n; i++) heights[i] = Integer.parseInt(str\_heights[i]);

// 记录从左向右的最长递增子序列和从右向左的最长递增子序列

int[] seq = new int[n], rev\_seq = new int[n];

int[] k = new int[n]; // 用于记录以i为终点的从左向右和从右向走的子序列元素个数

seq[0] = heights[0]; // 初始化从左向右子序列首元素为第一个元素

int index = 1; // 记录当前子序列的长度

for (int i = 1; i < n; i++) {

if (heights[i] > seq[index-1]) { // 当当前元素大于递增序列最后一个元素时

k[i] = index; // 其左边元素个数

seq[index++] = heights[i]; // 更新递增序列

} else { // 当当前元素位于目前维护递增序列之间时

// 使用二分搜索找到其所属位置

int l = 0, r = index - 1;

while (l < r) {

int mid = l + (r - l) / 2;

if (seq[mid] < heights[i]) l = mid + 1;

else r = mid;

}

seq[l] = heights[i]; // 将所属位置值进行替换

k[i] = l; // 其左边元素个数

}

}

// 随后，再从右向左进行上述操作

rev\_seq[0] = heights[n-1];

index = 1;

for (int i = n - 2; i >= 0; i--) {

if (heights[i] > rev\_seq[index-1]) {

k[i] += index;

rev\_seq[index++] = heights[i];

} else {

int l = 0, r = index - 1;

while (l < r) {

int mid = l + (r - l) / 2;

if (rev\_seq[mid] < heights[i]) l = mid + 1;

else r = mid;

}

rev\_seq[l] = heights[i];

k[i] += l;

}

}

int max = 1;

for (int num: k)

if (max < num) max = num;

// max+1为最大的k

System.out.println(n - max - 1);

}

}

}

}

# HJ25 数据分类处理

[数据分类处理](javascript:void(0);)

**题目描述**

信息社会，有海量的数据需要分析处理，比如公安局分析身份证号码、 QQ 用户、手机号码、银行帐号等信息及活动记录。

采集输入大数据和分类规则，通过大数据分类处理程序，将大数据分类输出。

请注意本题有多组输入用例。

**输入描述:**

﻿一组输入整数序列I和一组规则整数序列R，I和R序列的第一个整数为序列的个数（个数不包含第一个整数）；整数范围为0~0xFFFFFFFF，序列个数不限

**输出描述:**

﻿从R依次中取出R<i>，对I进行处理，找到满足条件的I：

I整数对应的数字需要连续包含R<i>对应的数字。比如R<i>为23，I为231，那么I包含了R<i>，条件满足。

按R<i>从小到大的顺序:

(1)先输出R<i>；

(2)再输出满足条件的I的个数；

(3)然后输出满足条件的I在I序列中的位置索引(从0开始)；

(4)最后再输出I。

附加条件：

(1)R<i>需要从小到大排序。相同的R<i>只需要输出索引小的以及满足条件的I，索引大的需要过滤掉

(2)如果没有满足条件的I，对应的R<i>不用输出

(3)最后需要在输出序列的第一个整数位置记录后续整数序列的个数(不包含“个数”本身)

序列I：15,123,456,786,453,46,7,5,3,665,453456,745,456,786,453,123（第一个15表明后续有15个整数）

序列R：5,6,3,6,3,0（第一个5表明后续有5个整数）

输出：30, 3,6,0,123,3,453,7,3,9,453456,13,453,14,123,6,7,1,456,2,786,4,46,8,665,9,453456,11,456,12,786

说明：

30----后续有30个整数

3----从小到大排序，第一个R<i>为0，但没有满足条件的I，不输出0，而下一个R<i>是3

6--- 存在6个包含3的I

0--- 123所在的原序号为0

123--- 123包含3，满足条件

示例1

**输入**

15 123 456 786 453 46 7 5 3 665 453456 745 456 786 453 123

5 6 3 6 3 0

**输出**

30 3 6 0 123 3 453 7 3 9 453456 13 453 14 123 6 7 1 456 2 786 4 46 8 665 9 453456 11 456 12 786

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.List;

import java.util.ArrayList;

import java.util.Collections;

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.HashMap;

import java.util.Arrays;

public class Main {

public static void main(String[] args) throws Exception{

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String str;

int inputNum = 0;

while((str = reader.readLine()) != null) {

if("".equals(str)) {

continue;

}

String iText = str;

String rText = reader.readLine();

String[] iArr = iText.split(" ");

String[] rArr = rText.split(" ");

long[] rLArr = new long[Integer.parseInt(rArr[0])];

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

rLArr[i] = Long.parseLong(rArr[i + 1]);

}

Arrays.sort(rLArr);

StringBuilder front = new StringBuilder();

int count = 0;

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

int num = 0;

if(i > 0 && rLArr[i] == rLArr[i- 1]) {

continue;

}

String pattern = rLArr[i] + "";

StringBuilder sb = new StringBuilder();

for(int j=1;j<iArr.length;j++) {

if(iArr[j].indexOf(pattern) != -1) {

num ++;

sb.append(" ").append(j - 1).append(" ").append(iArr[j]);

}

}

if(num > 0) {

count += num \* 2 + 2;

front.append(" ").append(rLArr[i]).append(" ").append(num).append(sb);

}

}

System.out.println(count + front.toString());

}

}

}

# HJ26 字符串排序

字符串排序

**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

注意有多组测试数据，即输入有多行，每一行单独处理（换行符隔开的表示不同行）

**输入描述:**

输入字符串

**输出描述:**

输出字符串

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).

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 s;

while((s = br.readLine()) != null){

char[] ch = s.toCharArray();

char[] chars = new char[ch.length];

int flag = 65, j=0;

while(flag<=90){

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

if((ch[i]>=65&&ch[i]<=90) || (ch[i]>=97&&ch[i]<=122)){

if(ch[i]==flag || ch[i]== flag+32){

chars[j] = ch[i];

j++;

}

}

}

flag++;

}

j=0;

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

if((ch[i]>=65&&ch[i]<=90) || (ch[i]>=97&&ch[i]<=122)){

ch[i] = chars[j];

j++;

}

}

System.out.println(String.valueOf(ch));

}

}

}

# HJ27 查找兄弟单词

[查找兄弟单词](javascript:void(0);)

**题目描述**

定义一个单词的“兄弟单词”为：交换该单词字母顺序，而不添加、删除、修改原有的字母就能生成的单词。

兄弟单词要求和原来的单词不同。例如：ab和ba是兄弟单词。ab和ab则不是兄弟单词。

现在给定你n个单词，另外再给你一个单词str，让你寻找str的兄弟单词里，字典序第k大的那个单词是什么？

注意：字典中可能有重复单词。本题含有多组输入数据。

**输入描述:**

先输入单词的个数n，再输入n个单词。

再输入一个单词，为待查找的单词x

最后输入数字k

**输出描述:**

输出查找到x的兄弟单词的个数m

然后输出查找到的按照字典顺序排序后的第k个兄弟单词，没有符合第k个的话则不用输出。

示例1

**输入**

3 abc bca cab abc 1

**输出**

2

bca

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

import java.util.\*;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader bf=new BufferedReader(new InputStreamReader(System.in));

String s=null;

while((s=bf.readLine())!=null){

String[] words=s.split(" ");

String str=words[words.length-2];

int k=Integer.parseInt(words[words.length-1]);

ArrayList<String> broWords=new ArrayList<>();

for (int i = 1; i < words.length-2; i++) {

if((!words[i].equals(str)) && words[i].length()==str.length()) {

char[] chStr=str.toCharArray();

char[] word=words[i].toCharArray();

int temp=0;

for (int j = 0; j < chStr.length; j++) {

for (int j2 = 0; j2 < word.length; j2++) {

if (word[j]==chStr[j2]) {

chStr[j2]='0';

temp++;

break;

}

}

}

if (temp==chStr.length) {

broWords.add(words[i]);

}

}

}

System.out.println(broWords.size());

if(k>0 && k<broWords.size()) {

Collections.sort(broWords);

System.out.println(broWords.get(k-1));

}

}

}

}

# HJ28 素数伴侣

[素数伴侣](javascript:void(0);)

**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

注意：数据可能有多组

**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

2

3 6

**输出**

2

0

import java.io.\*;

import java.util.\*;

public class Main {

static HashMap<String, Integer> hmap = new HashMap<>();

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String readIn;

while ((readIn = br.readLine()) != null) {

int n = Integer.parseInt(readIn);

String[] str\_nums = br.readLine().split(" ");

ArrayList<Integer> nums = new ArrayList<Integer>();

int odd\_count = 0;

for (String s : str\_nums) {

int i = Integer.valueOf(s);

nums.add(i);

if (i % 2 == 1) odd\_count++;

}

int[] odd\_nums = new int[odd\_count], even\_nums = new int[n - odd\_count];

int even\_idx = 0, odd\_idx = 0;

for (int i : nums) {

if (i % 2 == 0) {

even\_nums[even\_idx++] = i;

} else {

odd\_nums[odd\_idx++] = i;

}

}

int[] num = new int[n - odd\_count];

boolean[] used;

int ans = 0;

for (int i = 0; i < odd\_count; i++) {

used = new boolean[n - odd\_count];

if (find(odd\_nums[i], even\_nums, num, used)) {

ans++;

}

}

System.out.println(ans);

}

}

public static boolean find(int n, int[] evens, int[] num, boolean[] used) {

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

if (isP(n + evens[i]) && !used[i]) {

used[i] = true;

if (num[i] == 0 || find(num[i], evens, num, used)) {

num[i] = n;

return true;

}

}

}

return false;

}

public static boolean isP(int num) {

if (num <= 3) {

return num > 1;

}

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;

}

}

# HJ29 字符串加解密

字符串加解密

**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。

本题含有多组样例输入。

**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

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Arrays;

import java.util.Scanner;

public class Main{

public static void main(String []args) throws Exception{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str="";

while((str=br.readLine())!=null) {

String word=br.readLine();

System.out.println(jiami(str));

System.out.println(jiemi(word));

}

}

public static String jiami(String str) {

char ch[]=str.toCharArray();

StringBuilder sb=new StringBuilder();

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

if(ch[i]>='a'&&ch[i]<='z') {

if(ch[i]=='z')

sb.append('A');

else

sb.append((char)(ch[i]-32+1));

}

else if(ch[i]>='A'&&ch[i]<='Z') {

if(ch[i]=='Z')

sb.append('a');

else

sb.append((char)(ch[i]+32+1));

}

else if(ch[i]>='0'&&ch[i]<='9') {

if(ch[i]=='9')

sb.append('0');

else

sb.append(ch[i]-'0'+1);

}

}

return sb.toString();

}

public static String jiemi(String str) {

char ch[]=str.toCharArray();

StringBuilder sb=new StringBuilder();

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

if(ch[i]>='a'&&ch[i]<='z') {

if(ch[i]=='a')

sb.append('Z');

else

sb.append((char)(ch[i]-32-1));

}

else if(ch[i]>='A'&&ch[i]<='Z') {

if(ch[i]=='A')

sb.append('z');

else

sb.append((char)(ch[i]+32-1));

}

else if(ch[i]>='0'&&ch[i]<='9') {

if(ch[i]=='0')

sb.append('9');

else

sb.append(ch[i]-'0'-1);

}

}

return sb.toString();

}

}

# HJ30 字符串合并处理

[字符串合并处理](javascript:void(0);)

**题目描述**

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”

注意本题含有多组样例输入

**输入描述:**

本题含有多组样例输入。每组样例输入两个字符串，用空格隔开。

**输出描述:**

输出转化后的结果。每组样例输出一行。

示例1

**输入**

dec fab

**输出**

5D37BF

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 s = null;

while((s = bf.readLine())!=null){

String[] str = s.split(" ");

s = str[0] + str[1];

char[] array = sort(s);

System.out.println(transform(array));

}

}

public static char[] sort(String s){

char[] array = s.toCharArray();

int i,j;

for(i=2;i<array.length;i+=2){

if(array[i] < array[i-2]){

char tmp = array[i];

for(j=i;j>0 && array[j-2] > tmp; j-=2){

array[j] = array[j-2];

}

array[j] = tmp;

}

}

for(i=3;i<array.length;i+=2){

if(array[i] < array[i-2]){

char tmp = array[i];

for(j=i;j>1 && array[j-2]>tmp;j-=2){

array[j] = array[j-2];

}

array[j] = tmp;

}

}

return array;

}

public static String transform(char[] array){

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

int num = -1;

if(array[i] >= 'A' && array[i] <= 'F'){

num = array[i]-'A'+10;

}else if(array[i] >= 'a' && array[i] <= 'f'){

num = array[i]-'a'+10;

}else if(array[i] >= '0' && array[i] <= '9'){

num = array[i]-'0';

}

if(num != -1){ // 需要转换

num = (num&1)\*8 + (num&2)\*2 + (num&4)/2 + (num&8)/8;

if(num<10){

array[i] = (char)(num+'0');

}else if(num<16){

array[i] = (char)(num-10+'A');

}

}

}

return new String(array);

}

}

# HJ31 【中级】单词倒排

单词倒排

**题目描述**

对字符串中的所有单词进行倒排。

说明：

1、构成单词的字符只有26个大写或小写英文字母；

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**

I am a student

**输出**

student a am I

解法一：

import java.io.\*;

public class Main{

public static void main(String[] args) throws Exception {

InputStream in = System.in;

int available = in.available();

char[] arr = new char[available];

int off\_word = 0;

int off\_end = arr.length;

char c;

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

c = (char) in.read();

if((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z')) {

arr[off\_word++] = c;

} else if(off\_word > 0) {

System.arraycopy(arr, 0, arr, off\_end - off\_word, off\_word);

off\_end -= off\_word + 1;

off\_word = 0;

arr[off\_end] = ' ';

}

}

System.out.println(String.valueOf(arr, off\_end + 1, arr.length - off\_end - 1));

}

}

解法二：

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 && line.length() > 0) {

//trim() 方法用于删除字符串的头尾空白符。

char[] array = line.trim().toCharArray();

StringBuilder sb = new StringBuilder();

boolean flag = false;

int mark = -1;

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

if ((array[i] >= 'A' && array[i] <= 'Z') || (array[i] >= 'a' && array[i] <= 'z')) {

if (mark == -1)

mark = i;

flag = true;

} else if (flag) {

sb.append(array, i + 1, mark - i).append(' ');

mark = -1;

flag = false;

}

}

if (flag) {

sb.append(array, 0, mark + 1);

System.out.println(sb.toString());

} else {

System.out.println(sb.substring(0, sb.length() - 1));

}

}

}

}

# HJ32 【中级】字符串运用-密码截取

[密码截取](javascript:void(0);)

**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？

本题含有多组样例输入。

**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

import java.io.\*;

import java.util.\*;

public class Main{

public static void main(String[] args)throws Exception{

BufferedReader reader=new BufferedReader(new InputStreamReader(System.in));

String str;

while((str=reader.readLine())!=null){

int max=0;

char[] arr=str.toCharArray();

for(int i=1;i<arr.length;i++){

int left=i-1;

int right=i;

while(left>=0&&right<arr.length&&arr[left]==arr[right] ){

if(right-left+1>max){

max=right-left+1;

}

left--;

right++;

}

left=i-1;

right=i+1;

while(left>=0&&right<arr.length&&arr[left]==arr[right] ){

if(right-left+1>max){

max=right-left+1;

}

left--;

right++;

}

}

System.out.println(max);

}

}

}

# HJ33 整数与IP地址间的转换

[整数与IP地址间的转换](javascript:void(0);)

**题目描述**

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。

本题含有多组输入用例，每组用例需要你将一个ip地址转换为整数、将一个整数转换为ip地址。

**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

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 = null;

while ((str = br.readLine()) != null) {

String[] ip = str.split("\\.");

long num = Long.parseLong(br.readLine());

//转10进制

System.out.println(Long.parseLong(ip[0]) << 24 | Long.parseLong(ip[1]) << 16 |

Long.parseLong(ip[2]) << 8 | Long.parseLong(ip[3]));

//转ip地址

StringBuilder sb = new StringBuilder();

sb.append(String.valueOf((num >> 24) & 255)).append(".").append(String.valueOf((num >> 16) & 255))

.append(".").append(String.valueOf((num >> 8) & 255)).append(".").append(String.valueOf(num & 255));

System.out.println(sb.toString());

}

}

}

# HJ34 图片整理

[图片整理](javascript:void(0);)

**题目描述**

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。

本题含有多组样例输入。

**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

import java.util.\*;

import java.io.\*;

public class Main{

public static void main(String[] args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String s = "";

while((s=br.readLine())!=null){

char[] result = s.toCharArray();

Arrays.sort(result);

System.out.println(String.valueOf(result));

}

}

}

# HJ35 蛇形矩阵

[蛇形矩阵](javascript:void(0);)

**题目描述**

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。

例如，当输入5时，应该输出的三角形为：

1 3 6 10 15

2 5 9 14

4 8 13

7 12

11

请注意本题含有多组样例输入。

**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

解法一：

import java.io.\*;

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 num = Integer.parseInt(str);

StringBuilder sb = new StringBuilder();

for(int i = 1; i <= num; i++){

int start = (i - 1) \* i / 2 + 1;

int step = i + 1;

for(int j = 1; j <= num - i + 1; j++){

sb.append(start).append(" ");

start += step;

step ++ ;

}

sb.setCharAt(sb.length()-1,'\n');

}

sb.deleteCharAt(sb.length()-1);

System.out.println(sb.toString());

}

}

}

解法二：

思路：第0行，j=1，每一位数字为：(j\*j+j)/2

第1行，j=2，每一位数字为：(j\*j+j)/2-1.即为，去掉上一行的第一个数，然后后面的数补上来，并且减1

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();

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

for (int j = (i + 1); j <= num; j++) {

System.out.print(((j\*j + j) / 2) - i + " ");

}

System.out.println("");

}

}

}

}

# HJ36 字符串加密

字符串加密

**题目描述**

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。

本题有多组输入数据。

**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

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 key;

while ((key = br.readLine()) != null) {

char[] chars = key.toLowerCase().toCharArray();

char[] dict = new char[26];

int index = 0;

//tag语法配合continue使用，continue后跳转到此处运行

tag1:

for (char ch : chars) {

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

if (ch == dict[i]) {

continue tag1;

}

}

dict[index] = ch;

index++;

}

char ch = 'a';

tag2:

for (int i = 0; i < 26; i++) {

for (int j = 0; j < index; j++) {

if (dict[j] == ch) {

ch++;

continue tag2;

}

}

dict[index] = ch;

ch++;

index++;

}

String str = br.readLine();

char[] res = str.toCharArray();

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

if(res[i] - 'a'>=0){

res[i] = dict[res[i] - 'a'];

}else{

res[i] = dict[res[i] - 'A'];

}

}

System.out.println(String.valueOf(res));

}

}

}

# HJ37 统计每个月兔子的总数

统计每个月兔子的总数

**题目描述**

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？

本题有多组数据。

**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

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 monthCount = Integer.parseInt(line);

System.out.println(getTotalCount(monthCount));

}

}

public static int getTotalCount(int monthCount){

int a = 1;

int b = 1;

int c = 1;

for(int i=2;i<monthCount;++i){

c = a + b;

a = b;

b = c;

}

return c;

}

}

# HJ38 求小球落地5次后所经历的路程和第5次反弹的高度

[求小球落地5次后所经历的路程和第5次反弹的高度](javascript:void(0);)

**题目描述**

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？

最后的误差判断是小数点6位

**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

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 str;

while ((str = br.readLine()) != null) {

double height = Integer.parseInt(str);

System.out.println(get(height));

System.out.println(height(height));

}

}

public static double height(double h) {

for (int i = 0; i < 5; i++) {

h = h / 2;

}

return h;

}

public static double get(double h) {

return h + h + h / 2 + h / 4 + h / 8;

}

}

# HJ39 判断两个IP是否属于同一子网

[判断两个IP是否属于同一子网](javascript:void(0);)

**题目描述**

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。

输入一个子网掩码以及两个ip地址，判断这两个ip地址是否是一个子网络。

若IP地址或子网掩码格式非法则输出1，若IP1与IP2属于同一子网络输出0，若IP1与IP2不属于同一子网络输出2。

**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

255.0.0.0

193.194.202.15

232.43.7.59

**输出**

1

2

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.\*;

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) {

boolean flag = false;

String[] mask = str.trim().split("\\.");

String[] ip1 = br.readLine().trim().split("\\.");

String[] ip2 = br.readLine().trim().split("\\.");

if(mask.length != 4 || ip1.length != 4 || ip2.length != 4){

System.out.println(1);

continue;

}

int[] i1 = new int[4];

int[] i2 = new int[4];

int[] m = new int[4];

for (int i = 0; i < 4; i++) {

i1[i] = Integer.valueOf(ip1[i]);

i2[i] = Integer.valueOf(ip2[i]);

m[i] = Integer.valueOf(mask[i]);

}

if(!isValidIp(i1) || !isValidIp(i2) || !isValidMask(m)){

System.out.println(1);

continue;

}

for (int i = 0; i < 4; i++) {

if(!((i1[i] & m[i]) == (i2[i] & m[i]))){

System.out.println(2);

flag = true;

break;

}

}

if(!flag){

System.out.println(0);

}

}

}

public static boolean isValidIp(int[] s){

for (int i = 0; i < 4; i++) {

if(s[i] > 255 || s[i] < 0){

return false;

}

}

return true;

}

public static boolean isValidMask(int[] a){

return a[0] >= 0 && a[0] < 256 &&

a[1] >= 0 && a[1] < 256 &&

a[2] >= 0 && a[2] < 256 &&

a[3] >= 0 && a[3] < 256 &&

a[0] >= a[1] && a[1] >= a[2] && a[2] >= a[3];

}

}

# HJ40 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

[统计字符](javascript:void(0);)

**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。

本题包含多组输入。

**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

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 EnglishCharCount=0;

int BlankCharCount=0;

int NumberCharCount=0;

int OtherCharCount=0;

char[] chs = line.toCharArray();

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

if((chs[i]>='a'&&chs[i]<='z')||(chs[i]>='A'&&chs[i]<='Z')){

++EnglishCharCount;

continue;

}

else if(chs[i]==' '){

++BlankCharCount;

continue;

}

else if(chs[i]>='0'&&chs[i]<='9'){

++NumberCharCount;

continue;

}

else

++OtherCharCount;

}

System.out.println(EnglishCharCount);

System.out.println(BlankCharCount);

System.out.println(NumberCharCount);

System.out.println(OtherCharCount);

}

}

}

# HJ41 称砝码

[称砝码](javascript:void(0);)

**题目描述**

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

**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

解法一：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

/\*

\* 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)

\*/

public class Main {

public static void main(String[] args) throws NumberFormatException, IOException {

// 1.高效读数据

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String mark = null;

while ((mark = br.readLine()) != null) {

int n = Integer.parseInt(mark);

String[] s1 = br.readLine().split(" ");

String[] s2 = br.readLine().split(" ");

int[] weight = new int[s1.length];

int[] nums = new int[s2.length];

int sum = 0;

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

weight[i] = Integer.parseInt(s1[i]);

nums[i] = Integer.parseInt(s2[i]);

sum += weight[i] \* nums[i];

}

System.out.println(fama(n, weight, nums, sum));

}

}

public static int fama(int n, int[] weight, int[] nums, int sum){

boolean[] weg = new boolean[sum+1];

weg[0] =true;

weg[sum] = true;

int top = 0;

// 三重循环，分别表示 n个砝码，weight.length 个种类，每个种类有 num.length 个数量

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

for (int j = 0; j < nums[i]; j++) {

for(int k=top + weight[i]; k>= weight[i];k--){

// weg[k-weight[i]]== true,意思就是 k-weight[i] + weight[i] == k,可以称出 k 重量

if(weg[k -weight[i] ]){

weg[k] = true;

}

}

top += weight[i];

}

}

int count = 0;

for (boolean b : weg) {

if (b) count++;

}

return count;

}

}

解法二：

public static class Main {

/\*\*

\* 修正：看别人的做法，是第一个砝码的多种重量放入set，然后是下一个砝码多种可能重量。

\* ！！！这里面比较重要的是根据的是 <目前已有的重量> ，用这些重量加上下一个砝码的所有可能的重量，

\* 以重量为根据，而不是以砝码的多种排列组合为依据.

\* 所以说这个三重循环，第一个是遍历到第几个砝码，第二个是当前的set里面的重量，第三个是当前砝码的所有可能重量

\*/

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

while (sc.hasNext()) {

List<Integer> weightList = new ArrayList<>();

List<Integer> numList = new ArrayList<>();

Set<Integer> set = new HashSet<>();

int num = sc.nextInt();

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

weightList.add(sc.nextInt());

}

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

numList.add(sc.nextInt());

}

set.add(0);

for (int i = 0; i < num; i++) { //i循环是选择第几个砝码，依次放入set

List<Integer> currentList = new ArrayList<>(set);

for (Integer integer : currentList) { //循环是当前set里面的重量，因为set无法用index遍历所以换成List

for (int k = 1; k <= numList.get(i); k++) { //k循环是当前砝码的可能重量，全部加到set目前所有的数值上面

set.add(integer + weightList.get(i) \* k); //因为是要计算所有当前的set重量，所以用了无序的HashSet也没事

}

}

}

System.out.println(set.size());

}

}

}

# HJ42 学英语

[学英语](javascript:void(0);)

**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。

本题含有多组输入数据。

**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

import java.io.\*;

public class Main{

public static void main(String[] args) throws Exception {

String[] numstr = new String[]{null, "one", "two", "three", "four", "five", "six", "seven", "eight", "nine", "ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"};

String[] teens = new String[]{null, null, "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety"};

String[] points = new String[]{"", "thousand", "million", "billion"};

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str;

while((str = br.readLine()) != null) {

char[] arr = str.toCharArray();

int len = arr.length;

StringBuilder sb = new StringBuilder();

boolean andflag = false;

for(int i = (len - 1) / 3; i >= 0; i--) {

int m = len - 1 - i \* 3;

int h = m < 2 ? 0 : (arr[m - 2] - '0');

int t = (m < 1 ? 0 : ((arr[m - 1] - '0') \* 10)) + arr[m] - '0';

if(h == 0 && t == 0) {

continue;

}

if(t == 0 && sb.length() > 0) {

sb.append(" and");

}

if(h > 0) {

sb.append(' ').append(numstr[h]).append(" hundred");

if(t > 0)

sb.append(" and");

}

if(t > 20) {

sb.append(' ').append(teens[t / 10]);

if(t % 10 > 0)

sb.append(' ').append(numstr[t % 10]);

} else if (t > 0) {

sb.append(' ').append(numstr[t]);

}

sb.append(' ').append(points[i]);

}

sb.deleteCharAt(0);

System.out.println(sb.toString());

}

}

}

# HJ43 迷宫问题

[迷宫问题](javascript:void(0);)

**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。

本题含有多组数据。

**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**

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**

(0,0)

(1,0)

(2,0)

(2,1)

(2,2)

(2,3)

(2,4)

(3,4)

(4,4)

import java.io.\*;

public class Main{

public static void main(String[] args)throws Exception {

BufferedReader br= new BufferedReader(new InputStreamReader(System.in));

String str="";

while((str=br.readLine())!=null) {

String[] strs= str.split(" ");

int m= Integer.parseInt(strs[0]);

int n= Integer.parseInt(strs[1]);

int[][] nums=new int[m][n];

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

String strr = br.readLine();

String[] b = strr.split(" ");

for(int j=0;j<n;j++){

nums[i][j] = Integer.parseInt(b[j]);

}

}

String[] res={""};

peocess(nums,0,0,"(0,0)\n",res);

if(res[0].length()==0) System.out.println("(0,0)");

else System.out.println(res[0].substring(0,res[0].length()-1));

}

}

public static void peocess(int[][]arr,int x,int y,String path,String[]res) {

if ((x==arr.length-1)&&(y==arr[x].length-1)) {

if (arr[x][y]==0)

res[0]=path;

return;

}

if (x>arr.length-1||y>arr[x].length-1)

return;

if (x+1<arr.length&&arr[x+1][y]==0)

peocess(arr,x+1,y,path+"("+(x+1)+","+y+")\n",res);

if (y+1<arr[x].length&&arr[x][y+1]==0)

peocess(arr,x,y+1,path+"("+x+","+(y+1)+")\n",res);

}

}

# HJ44 Sudoku-Java

[Sudoku-Java](javascript:void(0);)

**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

import java.io.\*;

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){

char[][] arr = new char[9][];

arr[0] = str.toCharArray();

for(int i = 1 ; i<9 ; i++){

arr[i] = br.readLine().toCharArray();

}

dealnum(arr);

StringBuffer sb = new StringBuffer();

for (int i = 0;i<9;i++){

sb.append(String.valueOf(arr[i]) + '\n');

}

System.out.println(sb.toString());

}

}

private static boolean dealnum(char[][] arr) {

int x = 0, y = 0;

s: for(; x < 9; x++) {

for(y = 0; y < 17; y += 2) {

if(arr[x][y] == '0') {

break s;

}

}

}

if(x < 9 && y < 17) {

boolean[] stu = new boolean[10];

for(int i = 0; i < 9; i ++) {

stu[arr[x][i << 1] - '0'] = true;

stu[arr[i][y] - '0'] = true;

}

int lx = (x / 3) \* 3, ly = (y / 6) \* 6;

for (int i = lx; i < lx + 3; i++) {

for (int j = ly; j < ly + 5; j += 2) {

stu[arr[i][j] - '0'] = true;

}

}

for(int i = 1; i < 10; i++) {

if(stu[i]) continue;

arr[x][y] = (char)(i + '0');

if(dealnum(arr)) return true;

arr[x][y] = '0';

}

return false;

}

return true;

}

}

# HJ45 名字的漂亮度

[名字的漂亮度](javascript:void(0);)

**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”。

本题含有多组数据。

**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

import java.io.\*;

import java.util.\*;

public static class Main {

/\*\*

\* 题目理解为：

\* lisi=iils=26+26+25+24

\* zhangsan=aannzhgs=26+26+25+25+24+23+22+21

\*/

public static void main(String[] args) throws IOException {

BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));

String value;

while ((value = bf.readLine()) != null) {

int num = Integer.parseInt(value);

for (int index = 0; index < num; index++) {

String name = bf.readLine();

char[] nameChars = name.toCharArray();

int[] charMaxs = new int[150];

for (char nameChar : nameChars) {

charMaxs[nameChar]++;

}

Arrays.sort(charMaxs);

int max = 26;

int total = 0;

for (int indb = charMaxs.length - 1; indb >= 0; indb--) {

if (charMaxs[indb] == 0)

break;

total = total + charMaxs[indb] \* max;

max--;

}

System.out.println(total);

}

}

}

}

# HJ46 按字节截取字符串

[截取字符串](javascript:void(0);)

**题目描述**

输入一个字符串和一个整数k，截取字符串的前k个字符并输出

本题输入含有多组数据

**输入描述:**

第一行输入待截取的字符串

第二行输入一个正整数k，代表截取的长度

**输出描述:**

截取后的字符串

示例1

**输入**

abABCcDEF

6

**输出**

abABCc

import java.util.Scanner;

import java.io.\*;

public class Main {

public static void main(String[]args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String res;

int len;

while ((res = br.readLine())!= null) {

len = Integer.parseInt(br.readLine());

System.out.println(res.substring(0, len));

}

}

}

# HJ47 线性插值

[线性插值](javascript:void(0);)

**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。

**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

解法一：

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 line;

while ((line=bufferedReader.readLine()) != null){

int total = Integer.parseInt(line.split(" ")[0]);

if(total>0){

StringBuffer sb = new StringBuffer();

String thisline = bufferedReader.readLine();

String[] elements = thisline.split(" ");

int num = Integer.parseInt(elements[0]);

int value = Integer.parseInt(elements[1]);

int preNum = num;

int preValue = value;

sb.append(thisline).append("\n");

for(int i=1;i<total;i++){

thisline = bufferedReader.readLine();

elements = thisline.split(" ");

num = Integer.parseInt(elements[0]);

value = Integer.parseInt(elements[1]);

if(num==preNum){

continue;

}

for(int j=preNum+1; j<num; j++){

sb.append(j).append(" ").append(preValue + (value - preValue) / (num - preNum) \* (j - preNum)).append("\n");

}

preNum=num;

preValue=value;

sb.append(thisline).append("\r\n");

}

System.out.print(sb.toString());

}

}

}

}

解法二：

/\*\*

\* 只通过70%是因为当遇到重复项时,我们应该丢弃后出现的(这是题目要求),这时应该用一对变量来保存

\* 第一次出现的重复值,

\* 比如(n没啥用)

\* 4 n

\* 23 5

\* 46 25 (程序中用KEY和VALUE来保存他们)

\* 46 32 (舍弃该对数值)

\* 82 46 (应用82和46进行比较)

\* 对着这个例子把我的程序走一遍就全懂了!就是这么简单,花了我一上午

\*/

public static class Main {

/\*\*

\* 测试用例：

\* 输入：

\* 2

\* 4 4

\* 7 10

\* 输出：

\* 4 4

\* 5 6

\* 6 8

\* 7 10

\*/

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

while (sc.hasNext()) {

int n = sc.nextInt();

int m = sc.nextInt();

//二维数组存储键值对

int[][] data = new int[n][2];

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

data[i][0] = sc.nextInt();

data[i][1] = sc.nextInt();

}

ArrayList<String> list = new ArrayList<String>();

//若出现重复值,利用这两个变量存储前一个值,丢弃后出现的重复值(如题目要求)

int KEY = data[0][0];

int VALUE = data[0][1];

list.add(KEY + " " + VALUE);

for (int i = 1; i <= n - 1; i++) {

int diff = data[i][0] - KEY - 1;

if (diff > 0) {

for (int j = 1; j <= diff; j++) {

int key = KEY + j;

int value = VALUE + (data[i][1] - VALUE) / (data[i][0] - KEY) \* j;

list.add(key + " " + value);

}

list.add(data[i][0] + " " + data[i][1]);

KEY = data[i][0];

VALUE = data[i][1];

//两个键相等则直接跳过,此时KEY,VVALUE保存的还是上一个值

} else if (diff == -1)

continue;

else {

list.add(data[i][0] + " " + data[i][1]);

KEY = data[i][0];

VALUE = data[i][1];

}

}

for (String s : list)

System.out.println(s);

}

}

}

# HJ48 从单向链表中删除指定值的节点

[从单向链表中删除指定值的节点](javascript:void(0);)

**题目描述**

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

链表长度不大于1000，每个节点的值不大于10000。

本题含有多组样例。

**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，每个数后都要加空格

示例1

**输入**

5

2

3 2

4 3

5 2

1 4

3

**输出**

2 5 4 1

import java.io.\*;

public class Main {

//构建节点

public static class Node {

Node next;

int val;

Node(int val) {

this.val = val;

next = null;

}

}

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

while ((str = br.readLine()) != null) {

if (str.equals("")) continue;

String[] params = str.split(" ");//str.split(" ")将字符串按空格分开并返回为字符串数组的形式

int n = Integer.parseInt(params[0]);

Node head = new Node(Integer.parseInt(params[1]));//将字符串数字默认转换为整型的十进制数，使用的是Integer类

for (int i = 1; i < n; i++) {

int pre = Integer.parseInt(params[2\*i+1]), nxt = Integer.parseInt(params[2\*i]);

Node temp = head;

while (temp.val != pre) temp = temp.next;//遍历插入点的位置

Node node = new Node(nxt);

node.next = temp.next;

temp.next = node;

}

int del = Integer.parseInt(params[2\*n]);

StringBuilder ans = new StringBuilder();//StringBuilder类可以更改字符串，不会占用多余的空间，始终是一个数组

Node temp = head;

while (temp != null) {

if (temp.val != del) ans.append(temp.val).append(" ");//append为StringBuilder类中添加字符串的方法

temp = temp.next;

}

// 注意要求每个数后面都加空格

System.out.println(ans.toString());//转换为字符串格式

}

}

}

# HJ49 多线程

[多线程](javascript:void(0);)

**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。

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:**

本题含有多个样例输入。

输入一个int整数

**输出描述:**

对于每组样例，输出多个ABCD

示例1

**输入**

10

**输出**

ABCDABCDABCDABCDABCDABCDABCDABCDABCDABCD

解法一：

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){

int n=Integer.parseInt(str);

StringBuilder output=new StringBuilder();

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

output.append("ABCD");

}

System.out.println(output.toString());

}

}

}

解法二：

//线程池

import java.util.Scanner;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

class Myrunnable implements Runnable{

public String output;

public Myrunnable(String output){

this.output = output;

}

public void run(){

System.out.print(output);

}

}

public class Main {

public static void main(String[] args) {

Scanner in=new Scanner(System.in);

ExecutorService threadPool = Executors.newSingleThreadExecutor();

while(in.hasNextInt()) {

int n=in.nextInt();

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

threadPool.execute(new Myrunnable("A"));

threadPool.execute(new Myrunnable("B"));

threadPool.execute(new Myrunnable("C"));

threadPool.execute(new Myrunnable("D"));

}

try{

Thread.sleep(100);

}catch (InterruptedException e){

e.printStackTrace();

}

System.out.println();

}

threadPool.shutdown();

}

}

# HJ50 四则运算

四则运算

**题目描述**

输入一个表达式（用字符串表示），求这个表达式的值。

保证字符串中的有效字符包括[‘0’-‘9’],‘+’,‘-’, ‘\*’,‘/’ ,‘(’， ‘)’,‘[’, ‘]’,‘{’ ,‘}’。且表达式一定合法。

**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

import java.io.IOException;

import java.io.InputStream;

public class Main{

public static void main(String[] args) throws IOException {

InputStream in = System.in;

System.out.println(new ExprDemo().expr(in));

}

public static class ExprDemo {

public char lastsign1 = 0, lastsign2 = 0;

public int temp1 = 0, temp2 = 0;

private static final char TEMPCHAR = 0;

public int expr(InputStream in) throws IOException {

int result = 0;

char c;

a: while((c = (char)in.read()) != '\n') {

switch (c) {

case ')':

case ']':

case '}': break a;

case '(':

case '[':

case '{': temp2 = new ExprDemo().expr(in); break;

case '+':

case '-':

jisuan1(TEMPCHAR);

result = jisuan2(c, result);

break;

case '\*':

case '/':

jisuan1(c);

break;

default: temp2 = temp2 \* 10 + c - '0'; break;

}

}

jisuan1(TEMPCHAR);

result = jisuan2(TEMPCHAR, result);

return result;

}

private void jisuan1(char c) {

switch (lastsign2) {

case 0: temp1 = temp2; break;

case '\*': temp1 \*= temp2; break;

case '/': temp1 /= temp2; break;

default: break;

}

temp2 = 0;

lastsign2 = c;

}

private int jisuan2(char c, int result) {

switch (lastsign1) {

case 0: result = temp1; break;

case '-': result -= temp1; break;

case '+': result += temp1; break;

default: break;

}

temp1 = 0;

lastsign1 = c;

return result;

}

}

}

# HJ51 输出单向链表中倒数第k个结点

[输出单向链表中倒数第k个结点](javascript:void(0);)

**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。

正常返回倒数第k个结点指针，异常返回空指针

本题有多组样例输入。

**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

解法一：

import java.io.\*;

public class Main {

public static void main(String[] args) throws Exception{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

while((str = br.readLine())!=null){

int num = Integer.parseInt(str);

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

int num2 = Integer.parseInt(br.readLine().trim());

if(num2<=0||num2>numStr.length){

System.out.println(num2);

}else{

System.out.println(numStr[numStr.length-num2]);

}

}

}

}

解法二：

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

while (in.hasNext()) {

ListNode node = new ListNode();

node.next = null;

int N = in.nextInt();

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

ListNode p = new ListNode();

int x = in.nextInt();

p.next = node.next;

p.data = x;

node.next = p;

}

int k = in.nextInt();

ListNode kthNode = getKthNode(node,k);

System.out.println(kthNode.data);

}

}

public static ListNode getKthNode(ListNode node,int k){

ListNode front = node,behind = node;

int x=0;

while(front.next!=null && x<=k){

x++;

front = front.next;

}

return front;

}

}

class ListNode {

public int data;

public ListNode next;

}

# HJ52 计算字符串的距离

计算字符串的距离

**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。

本题含有多组输入数据。

**输入描述:**

每组用例一共2行，为输入的两个字符串

**输出描述:**

每组用例输出一行，代表字符串的距离

示例1

**输入**

abcdefg

abcdef

abcde

abcdf

abcde

bcdef

**输出**

1

1

2

import java.io.\*;

public class Main{

public static void main(String[] args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str1;

while((str1=br.readLine())!=null){

String str2 = br.readLine();

System.out.println(stringDistance(str1.toCharArray(),str2.toCharArray()));

}

}

private static int stringDistance(char[] a, char[] b) {

int[][] len = new int[a.length + 1][b.length + 1];

for (int i = 0; i < a.length + 1; i++) {

len[i][0] = i;

}

for (int j = 0; j <b.length + 1; j++) {

len[0][j] = j;

}

for (int i = 1; i < a.length + 1; i++) {

for (int j = 1; j < b.length + 1; j++) {

if (a[i - 1] == b[j - 1]) {

len[i][j] = len[i - 1][j - 1];

} else {

len[i][j] = Math.min(Math.min(len[i - 1][j], len[i - 1][j - 1]), len[i][j - 1]) + 1;

}

}

}

return len[a.length][b.length];

}

}

# HJ53 iNOC产品部-杨辉三角的变形

杨辉三角的变形

**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)

本题有多组输入数据，输入到文件末尾，请使用while(cin>>)等方式读入

**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

2

**输出**

3

-1

题解：规律为:

| n | （对4求余的结果）%4 | print |
| --- | --- | --- |
| 4、8、12…… | 0 | 3 |
| 5、9、13…… | 1 | 2 |
| 6、10、14 …… | 2 | 4 |
| 7、11、15…… | 3 | 2 |

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.\*;

public class Main {

// iNOC产品部-杨辉三角的变形

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(getResult(line));

}

}

public static int getResult(String str) {

int i = Integer.valueOf(str);

if (i <= 2) {

return -1;

} else if (i % 2 == 1) {

return 2;

} else {

if (i % 4 == 0) {

return 3;

} else {

return 4;

}

}

}

}

# HJ54 表达式求值

[表达式求值](javascript:void(0);)

**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

import java.io.\*;

public class Main{

public static void main(String[] args) throws Exception {

InputStream in = System.in;

System.out.println(new ExprDemo().expr(in));

}

public static class ExprDemo {

public char lastsign1 = 0, lastsign2 = 0;

public int temp1 = 0, temp2 = 0;

private static final char tempchar = 0;

public int expr(InputStream in) throws IOException {

int result = 0;

char c;

a: while((c = (char)in.read()) != '\n') {

switch (c) {

case ')': break a;

case '(': temp2 = new ExprDemo().expr(in); break;

case '+':

case '-':

jisuan1(tempchar);

result = jisuan2(c, result);

break;

case '\*':

case '/':

jisuan1(c);

break;

default: temp2 = temp2 \* 10 + c - '0'; break;

}

}

jisuan1(tempchar);

result = jisuan2(tempchar, result);

return result;

}

private void jisuan1(char c) {

switch (lastsign2) {

case 0: temp1 = temp2; break;

case '\*': temp1 \*= temp2; break;

case '/': temp1 /= temp2; break;

default: break;

}

temp2 = 0;

lastsign2 = c;

}

private int jisuan2(char c, int result) {

switch (lastsign1) {

case 0: result = temp1; break;

case '-': result -= temp1; break;

case '+': result += temp1; break;

default: break;

}

temp1 = 0;

lastsign1 = c;

return result;

}

}

}

# HJ55 （练习用）挑7

[挑7](javascript:void(0);)

**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

10

**输出**

3

1

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 input = "";

while ((input = bufferedReader.readLine()) != null) {

int count=0;

int num=Integer.parseInt(input);

for(int i=7;i<=num;i++){

if(i%7==0||contain7(i)){

count++;

}

}

System.out.println(count);

}

}

public static boolean contain7(int n) {

while (n > 0) {

if (n % 10 == 7) {

return true;

} else {

n /= 10;

}

}

return false;

}

}

# HJ56 iNOC产品部--完全数计算

[完全数计算](javascript:void(0);)

**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。

输入n，请输出n以内(含n)完全数的个数。计算范围, 0 < n <= 500000

本题输入含有多组样例。

**输入描述:**

输入一个数字n

**输出描述:**

输出不超过n的完全数的个数

示例1

**输入**

1000

7

100

**输出**

3

1

2

import java.io.\*;

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){

System.out.println(conut2(Integer.valueOf(str)));

}

}

public static int conut2(int n){

if(n<6){

return 0;

} else if(n<28){

return 1;

}else if(n<496){

return 2;

} else if(n<8128){

return 3;

} else if(n<33550336){

return 4;

} else {

return -1;

}

}

//解法二

public static int count(int n){

int result = 0;

for(int i =1;i<n;i++){

int sum = 0;

for(int j=1;j<=i/2;j++){

if(i%j==0){

sum += j;

}

}

if(sum == i){

result ++;

}

}

return result;

}

}

# HJ57 无线OSS－高精度整数加法

高精度整数加法

**题目描述**

输入两个用字符串表示的整数，求它们所表示的数之和。

字符串的长度不超过10000。

本题含有多组样例输入。

**输入描述:**

输入两个字符串。保证字符串只含有'0'~'9'字符

**输出描述:**

输出求和后的结果

示例1

**输入**

9876543210

1234567890

**输出**

11111111100

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;

while ((line = br.readLine()) != null && line.length() > 0) {

System.out.println(add(line.trim(), br.readLine().trim()));

}

}

static String add(String s1, String s2) {

if (s1.length() == 0 || s2.length() == 0)

return "";

boolean neg1 = s1.charAt(0) == '-';

boolean neg2 = s2.charAt(0) == '-';

if (!(neg1 ^ neg2)) {

if (s1.length() < s2.length()) {

String temp = s1;

s1 = s2;

s2 = temp;

}

} else if (neg1) {

if (s1.length() < s2.length() + 1) {

String temp = s1;

s1 = s2;

s2 = temp;

neg1 = false;

neg2 = true;

}

} else if (neg2) {

if (s1.length() + 1 < s2.length()) {

String temp = s1;

s1 = s2;

s2 = temp;

neg1 = true;

neg2 = false;

}

}

int[] lmax = new int[neg1 ? s1.length() - 1 : s1.length()];

for (int i = neg1 ? 1 : 0; i < lmax.length; ++i)

lmax[i] = s1.charAt(i) - '0';

int[] lmin = new int[neg2 ? s2.length() - 1 : s2.length()];

for (int i = neg2 ? 1 : 0; i < lmin.length; ++i)

lmin[i] = s2.charAt(i) - '0';

int i = lmax.length - 1, j = lmin.length - 1;

if (!(neg1 ^ neg2)) {

int[] carry = new int[1];

while (j >= 0) {

add(lmax, i, lmin[j], carry);

--i;

--j;

}

StringBuilder sb = new StringBuilder();

if (neg1)

sb.append('-');

if (carry[0] == 1)

sb.append(1);

for (i = 0; i < lmax.length; ++i)

sb.append(lmax[i]);

return sb.toString();

} else {

int flag = 0;

boolean neg = true;

if (i == j) {

flag = -1;

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

if (lmax[k] > lmin[k]) {

flag = 0;

neg = neg1;

break;

} else if (lmax[k] < lmin[k]) {

flag = 1;

neg = neg2;

break;

}

}

}

if (flag == -1)

return "0";

if (flag == 1) {

int[] temp = lmax;

lmax = lmin;

lmin = temp;

}

while (j >= 0) {

minus(lmax, i, lmin[j]);

--i;

--j;

}

int L = 0;

for (i = 0; i < lmax.length; ++i) {

if (lmax[i] == 0) {

++L;

} else {

break;

}

}

StringBuilder sb = new StringBuilder();

if (neg)

sb.append('-');

for (i = L; i < lmax.length; ++i)

sb.append(lmax[i]);

return sb.toString();

}

}

static void add(int[] lmax, int i, int val, int[] carry) {

if (i == -1) {

carry[0] = 1;

return;

}

lmax[i] += val;

if (lmax[i] >= 10) {

lmax[i] = lmax[i] - 10;

add(lmax, --i, 1, carry);

}

}

static void minus(int[] max, int i, int val) {

max[i] -= val;

if (max[i] < 0) {

max[i] = max[i] + 10;

minus(max, --i, 1);

}

}

}

# HJ58 输入n个整数，输出其中最小的k个

输入n个整数，输出其中最小的k个

**题目描述**

输入n个整数，输出其中最小的k个。

本题有多组输入样例，请使用循环读入，比如while(cin>>)等方式处理

**输入描述:**

第一行输入两个整数n和k

第二行输入一个整数数组

**输出描述:**

输出一个从小到大排序的整数数组

示例1

**输入**

5 2

1 3 5 7 2

**输出**

1 2

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

while ((str = br.readLine()) != null) {

if (str.equals("")) continue;

String[] params = str.split(" ");

int n = Integer.parseInt(params[0]), k = Integer.parseInt(params[1]);

int[] res = new int[n];

int start = 0, index = 0;

if (params.length > 2) start = 2;

else params = br.readLine().split(" ");

for (int i = start; i < params.length; i++) {

res[index++] = Integer.parseInt(params[i]);

}

Arrays.sort(res);

StringBuilder ans = new StringBuilder();

for (int i = 0; i < k; i++) ans.append(res[i]).append(" ");

System.out.println(ans.toString().trim());

}

}

}

# HJ59 找出字符串中第一个只出现一次的字符

找出字符串中第一个只出现一次的字符(题面已经更新)

**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:**

输入几个非空字符串

**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

aabb

**输出**

o

-1

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 input;

while ((input = br.readLine()) != null) {

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

char c = input.charAt(i);

//indexOf：字符首次出现对位置，lastIndexOf：字符下一次出现的位置

if (input.indexOf(c) == input.lastIndexOf(c)) {

System.out.println(c);

break;

}

if (i == input.length() - 1) {

System.out.println(-1);

}

}

}

}

}

# HJ60 查找组成一个偶数最接近的两个素数

[查找组成一个偶数最接近的两个素数](javascript:void(0);)

**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。

本题含有多组样例输入。

**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

import java.io.\*;

import java.util.\*;

public class Main{

public static void main(String[] args) throws Exception{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String content ;

while(null != (content = br.readLine())){

int num = Integer.parseInt(content);

for(int i = num/2;i>2;i--){

if(check(i) && check(num-i)){

System.out.println(i);

System.out.println(num-i);

break;

}

}

}

}

public static boolean check(int num){

for(int i = 2;i\*i<=num;i++){

if(num%i == 0){

return false;

}

}

return true;

}

}

# HJ61 放苹果

放苹果

**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。

数据范围：0<=m<=10，1<=n<=10。

本题含有多组样例输入。

**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

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.\*;

public class Main{

public static void main(String[] args) throws Exception{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = " ";

while((str = br.readLine()) != null){

String[] s = str.split(" ");

int m = Integer.parseInt(s[0]);

int n = Integer.parseInt(s[1]);

System.out.println(count(m,n));

}

}

/\*\*

\* 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){

if(n == 1 || m == 0)return 1;

else if(n > m)return count(m,m);

else return count(m,n - 1) + count(m - n,n);

}

}

# HJ62 查找输入整数二进制中1的个数

[查找输入整数二进制中1的个数](javascript:void(0);)

**题目描述**

输入一个正整数，计算它在二进制下的1的个数。

注意多组输入输出！！！！！！

**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

**说明**

5的二进制表示是101，有2个1

import java.io.\*;

public class Main {

public static void main(String[] args) throws Exception{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str;

while((str = br.readLine()) != null){

int n = Integer.parseInt(str);

String s = Integer.toBinaryString(n);

int cnt = 0;

for(char c : s.toCharArray())

if(c == '1')

cnt++;

System.out.println(cnt);

}

}

}

# HJ63 DNA序列

[DNA序列](javascript:void(0);)

**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。

本题含有多组样例输入。

**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

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

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 len = Integer.parseInt(br.readLine());

char[] c = str.toCharArray();

int max = 0;

int start = 0;

for(int i = 0; i < c.length - len; i++) {

int count = 0;

for(int j = 0; j < len; j++) {

if(c[i + j] == 'G' || c[i + j] == 'C')

count ++;

}

if(count > max) {

start = i;

max = count;

}

}

StringBuilder builder = new StringBuilder();

for(int i = start; i < start + len; i++) {

builder.append(c[i]);

}

System.out.println(builder.toString());

}

}

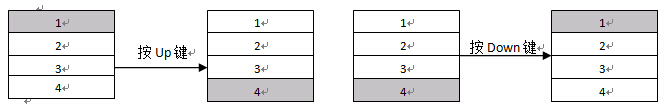
}

# HJ64 MP3光标位置

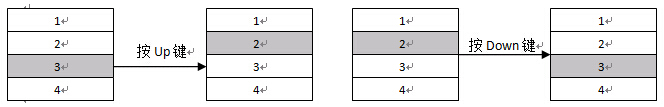
[MP3光标位置](javascript:void(0);)

**题目描述**

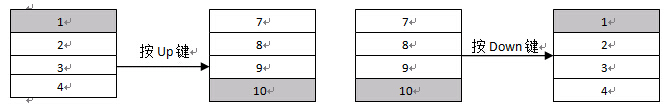
MP3 Player因为屏幕较小，显示歌曲列表的时候每屏只能显示几首歌曲，用户要通过上下键才能浏览所有的歌曲。为了简化处理，假设每屏只能显示4首歌曲，光标初始的位置为第1首歌。



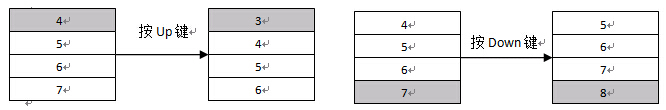
现在要实现通过上下键控制光标移动来浏览歌曲列表，控制逻辑如下：



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

本题含有多组输入数据！

**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

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 = null;

while ((str = br.readLine()) != null) {

String str2 = br.readLine();

int num = Integer.parseInt(str);

char[] array = str2.toCharArray();

int current = 1;

int start = 1;

for (char one : array) {

if (num <= 4) {

if(one == 'U') {

if(current == 1) {

current = num;

}else {

current--;

}

}

else if(one == 'D') {

if(current == num) {

current = 1;

}else {

current++;

}

}

}

if(num > 4) {

if(one == 'U') {

if(current == 1) {

current = num;

start = num-3;

}else if(current == start) {

current--;

start--;

}else {

current--;

}

}else if(one == 'D') {

if(current == num) {

current = 1;

start = 1;

}else if(current == start+3) {

current++;

start++;

}else {

current++;

}

}

}

}

StringBuilder sb = new StringBuilder();

for(int i=1;i<=4;i++) {

if(num >=i) {

sb.append(start + i -1).append(" ");

}

}

System.out.println(sb.toString().trim());

System.out.println(current);

}

}

}

# HJ65 查找两个字符串a,b中的最长公共子串

[查找两个字符串a,b中的最长公共子串](javascript:void(0);)

**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。

注：子串的定义：将一个字符串删去前缀和后缀（也可以不删）形成的字符串。请和“子序列”的概念分开！

本题含有多组输入数据！

**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

import java.io.\*;

import java.util.\*;

public class Main{

public static void main(String[] args) throws Exception{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str;

while((str = br.readLine())!=null){

String ss = br.readLine();

if(str.length()<ss.length()){

System.out.println(res(str,ss));

}else{

System.out.println(res(ss,str));

}

}

}

public static String res(String s,String c){

char[] ch1 = s.toCharArray();

char[] ch2 = c.toCharArray();

int[][] ins = new int[ch1.length + 1][ch2.length + 1];

int max = 0;

int start = 0;

for (int i = 0; i < ch1.length; i++) {

for (int j = 0; j < ch2.length; j++) {

if(ch1[i]==ch2[j]){

ins[i+1][j+1] = ins[i][j]+1;

if(ins[i+1][j+1]>max){

max = ins[i+1][j+1];

start = i-max;

}

}

}

}

return s.substring(start+1,start+max+1);

}

}

# HJ66 配置文件恢复

[配置文件恢复](javascript:void(0);)

**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 delete | no board at all |
| reboot backplane | impossible |
| backplane abort | install first |
| he he | unknown 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：unknown 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 和 reboot backplane，执行结果为：unkown command。

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“unknown 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

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

public class Main {

public static void main(String[] args) throws IOException{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str="";

StringBuilder sb=new StringBuilder();

String[] str1={"reset","reset board","board add","board delete","reboot backplane","backplane abort"};

Out:

while((str=br.readLine())!=null){

String[] str2=str.split(" ");

if(str2.length==1){

String cmd=str1[0];

for(int i=0;i<str2[0].length();i++){

if(str2[0].charAt(i)!=cmd.charAt(i)){

sb.append("unknown command").append("\n");

continue Out;

}

}

sb.append("reset what").append("\n");

}else if(str2.length==2){

if(str2[0].charAt(0)=='r'){

if((str2[0].equals("r")&&str2[1].equals("b"))||(str2[0].equals("re")&&str2[1].equals("b")))

sb.append("unknown command").append("\n");

else if(str2[0].length()<6 && str2[1].length()<6 && str2[0].equals(str1[1].substring(0,str2[0].length()))&&str2[1].equals(str1[1].substring(6,6+str2[1].length())))

sb.append("board fault").append("\n");

else if(str2[0].length()<7 && str2[1].length()<10 && str2[0].equals(str1[4].substring(0,str2[0].length()))&&str2[1].equals(str1[4].substring(7,7+str2[1].length())))

sb.append("impossible").append("\n");

else sb.append("unknown command").append("\n");}

else if(str2[0].charAt(0)=='b'){

if((str2[0].equals("b")&&str2[1].equals("a")))

sb.append("unknown command").append("\n");

else if(str2[0].length()<6 && str2[1].length()<4 && str2[0].equals(str1[2].substring(0,str2[0].length()))&&str2[1].equals(str1[2].substring(6,6+str2[1].length())))

sb.append("where to add").append("\n");

else if(str2[0].length()<6 && str2[1].length()<7 && str2[0].equals(str1[3].substring(0,str2[0].length()))&&str2[1].equals(str1[3].substring(6,6+str2[1].length())))

sb.append("no board at all").append("\n");

else if(str2[0].length()<10 && str2[1].length()<6 && str2[0].equals(str1[5].substring(0,str2[0].length()))&&str2[1].equals(str1[5].substring(10,10+str2[1].length())))

sb.append("install first").append("\n");

}else sb.append("unknown command").append("\n");

}

}

sb.deleteCharAt(sb.length()-1);

System.out.println(sb.toString());

}

}

# HJ67 24点游戏算法

[24点游戏算法](javascript:void(0);)

**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

本题含有多组样例输入。

**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

import java.io.\*;

public class Main {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str;

while((str = br.readLine()) != null) {

String[] numstrs = str.split(" ");

int[] nums = new int[4];

nums[0] = Integer.parseInt(numstrs[0]);

nums[1] = Integer.parseInt(numstrs[1]);

nums[2] = Integer.parseInt(numstrs[2]);

nums[3] = Integer.parseInt(numstrs[3]);

boolean flag = false;

for(int i = 0; i < 4; i++) {

if(is24(nums, 1 << i, nums[i])) {

flag = true;

break;

}

}

System.out.println(flag);

}

}

private static boolean is24(int[] nums, int stu, int val) {

if(val == 24) return true;

else if (stu == 7) return false;

for(int i = 0; i < 4; i++) {

int temp = 1 << i;

if ((temp & stu) == 0) {

temp |= stu;

if(is24(nums, temp, val + nums[i])

||is24(nums, temp, val - nums[i])

||is24(nums, temp, val \* nums[i])

||(val % nums[i] == 0 && is24(nums, temp, val + nums[i])))

return true;

}

}

return false;

}

}

# HJ68 成绩排序

[成绩排序](javascript:void(0);)

**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

jack 70

Tom 70

peter 96

注：0代表从高到低，1代表从低到高

本题含有多组输入数据！

**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

示例1

**输入**

3

0

fang 90

yang 50

ning 70

**输出**

fang 90

ning 70

yang 50

import java.io.\*;

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.trim());

int bool = Integer.parseInt(br.readLine().trim());

String[] name = new String[n];

int[] score = new int[n];

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

str = br.readLine().trim();

String[] temp = str.split(" ");

name[i] = temp[0];

score[i] = Integer.parseInt(temp[1]);

}

if (bool == 0) { // 由高到低

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

for (int j = 0; j < n - 1 - i; j++) {

if (score[j + 1] > score[j]) {

String na = name[j];

name[j] = name[j + 1];

name[j + 1] = na;

int t = score[j];

score[j] = score[j + 1];

score[j + 1] = t;

}

}

}

} else { // 由低到高

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

for (int j = 0; j < n - 1 - i; j++) {

if (score[j + 1] < score[j]) {

String na = name[j];

name[j] = name[j + 1];

name[j + 1] = na;

int t = score[j];

score[j] = score[j + 1];

score[j + 1] = t;

}

}

}

}

StringBuilder sb = new StringBuilder();

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

sb.append(name[i] + " ");

sb.append(score[i]);

sb.append("\n");

}

sb.deleteCharAt(sb.length() - 1);

System.out.println(sb);

}

}

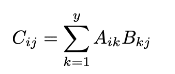
}

# HJ69 矩阵乘法

[矩阵乘法](javascript:void(0);)

**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



矩阵的大小不超过100\*100

**输入描述:**

输入包含多组数据，每组数据包含：

第一行包含一个正整数x，代表第一个矩阵的行数

第二行包含一个正整数y，代表第一个矩阵的列数和第二个矩阵的行数

第三行包含一个正整数z，代表第二个矩阵的列数

之后x行，每行y个整数，代表第一个矩阵的值

之后y行，每行z个整数，代表第二个矩阵的值

**输出描述:**

对于每组输入数据，输出x行，每行z个整数，代表两个矩阵相乘的结果

示例1

**输入**

2

3

2

1 2 3

3 2 1

1 2

2 1

3 3

**输出**

14 13

10 11

import java.io.\*;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

while ((str = br.readLine()) != null) {

if (str.equals("")) continue;

int x = Integer.parseInt(str);

int y = Integer.parseInt(br.readLine());

int z = Integer.parseInt(br.readLine());

int[][] matrix1 = new int[x][y];

int[][] matrix2 = new int[y][z];

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

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

for (int j = 0; j < y; j++) {

matrix1[i][j] = Integer.parseInt(params[j]);

}

}

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

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

for (int j = 0; j < z; j++) {

matrix2[i][j] = Integer.parseInt(params[j]);

}

}

StringBuilder ans = new StringBuilder();

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

for (int j = 0; j < z; j++) {

int temp = 0;

for (int k = 0; k < y; k++) {

temp += matrix1[i][k] \* matrix2[k][j];

}

ans.append(temp).append(" ");

}

ans.deleteCharAt(ans.length()-1).append("\n");

}

System.out.print(ans.toString());

}

}

}

# HJ70 矩阵乘法计算量估算

[矩阵乘法计算量估算](javascript:void(0);)

**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。

本题含有多组样例输入！

**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计算的法则为一个字符串，仅由左右括号和大写字母（'A'~'Z'）组成，保证括号是匹配的且输入合法！

**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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.List;

import java.util.Stack;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

while ((str = br.readLine()) != null) {

int num = Integer.parseInt(str);

int [][] arr = new int[num][2];

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

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

arr[i][0] = Integer.parseInt(sa[0]);

arr[i][1] = Integer.parseInt(sa[1]);

}

int n = arr.length -1;

char [] ca = br.readLine(). toCharArray();

Stack<Integer> stack = new Stack<>();

int sum = 0;

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

char one = ca[i];

if(one == ')') {

stack.push(-1);

}else if(one == '(') {

int n1 = stack.pop();

int n2 = stack.pop();

sum+= arr[n1][0]\*arr[n2][0]\*arr[n2][1];

arr[n1][1] = arr[n2][1];

stack.pop();

stack.push(n1);

}else {

stack.push(n);

n--;

}

}

System.out.println(sum);

}

}

}

# HJ71 字符串通配符

字符串通配符

**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

本题含有多组样例输入！

**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

解法一：

import java.io.\*;

public class Main {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String regx;

while (null != (regx = br.readLine())) {

String content = br.readLine();

System.out.println(match(content.toCharArray(), 0, regx.toCharArray(), 0));

}

}

public static boolean match(char[] content, int offsetContent, char[] regx, int offsetRegx) {

while (offsetContent < content.length && offsetRegx < regx.length) {

if ('\*' == regx[offsetRegx]) {

if (offsetRegx == regx.length - 1) {

return true;

} else if (isMatch(content[offsetContent], regx[offsetRegx + 1]) && match(content, offsetContent, regx, offsetRegx + 1)) {

return true;

} else {

offsetContent++;

}

} else if ('?' == regx[offsetRegx] || content[offsetContent] == regx[offsetRegx]) {

offsetContent++;

offsetRegx++;

} else {

return false;

}

}

return (offsetContent == content.length) && (offsetRegx == regx.length);

}

public static boolean isMatch(char contentChar, char regxChar) {

return '\*' == regxChar || '?' == regxChar || contentChar == regxChar;

}

}

解法二：

import java.util.Scanner;

public class HJ71\_2 {

/\*\*

\* 实现通配符

\* ? 匹配一个字符 [0-9A-Za-z]{1}

\* \* 匹配0个或以上的字符 [0-9A-Za-z]{0,}

\* （字符由英文字母和数字0-9组成，不区分大小写。下同）

\*/

public static class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

while (scanner.hasNextLine()) {

String regx = scanner.nextLine();

String string = scanner.nextLine();

//做相应的替换

regx = regx.replaceAll("\\?", "[0-9A-Za-z]{1}");

regx = regx.replaceAll("\\\*", "[0-9A-Za-z]{0,}");

regx = regx.replaceAll("\\.", "\\\\.");

//matches() 方法用于检测字符串是否匹配给定的正则表达式。

boolean result = string.matches(regx);

System.out.println(result);

}

}

}

/\*\*

\* Java 递归

\*/

public static class Main2 {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

while (in.hasNext()) {

String s1 = in.nextLine();

String s2 = in.nextLine();

System.out.println(helper(s1, s2, 0, 0));

}

}

private static boolean helper(String s1, String s2, int p1, int p2) {

if (p1 == s1.length() && p2 == s2.length()) {

return true;

} else if (p1 == s1.length() || p2 == s2.length()) {

return false;

}

//遇到'\*'两种情况，要不就各跳过一个比较后面，要不就s2继续往后跳先不比较

if (s1.charAt(p1) == '\*') {

return helper(s1, s2, p1, p2 + 1) || helper(s1, s2, p1 + 1, p2 + 1);

//遇到'?'跟两个一样操作一样，直接指针都往后移一个继续比较

} else if (s1.charAt(p1) == '?' || s1.charAt(p1) == s2.charAt(p2)) {

return helper(s1, s2, p1 + 1, p2 + 1);

} else {

return false;

}

}

}

}

# HJ72 百钱买百鸡问题

[百钱买百鸡问题](javascript:void(0);)

**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？

**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

public class Main{

public static void main(String[] args) {

for (int x=0;x<100;x++){

for (int y=0;y<100;y++){

if (14\*x+8\*y==200){

System.out.println(x+" "+y+" "+(100-x-y));

}

}

}

}

}

# HJ73 计算日期到天数转换

[计算日期到天数转换](javascript:void(0);)

**题目描述**

根据输入的日期，计算是这一年的第几天。

测试用例有多组，注意循环输入

**输入描述:**

输入多行，每行空格分割，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

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[] strs = str.split(" ");

int year = Integer.parseInt(strs[0]);

int month = Integer.parseInt(strs[1]);

int day = Integer.parseInt(strs[2]);

int result = day;

for (int i = 1; i < month; i++) {

result += getMonthDays(year, i);

}

System.out.println(result);

}

}

public static int getMonthDays(int year, int month) {

if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12) {

return 31;

}

if (month == 2) {

return year % 4 == 0 ? 29 : 28;

}

return 30;

}

}

# HJ74 参数解析

[参数解析](javascript:void(0);)

**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:\\

import java.io.\*;

public class Main{

public static void main(String[] args)throws IOException{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String s;

while((s=br.readLine())!=null){

char[] chars=s.toCharArray();

StringBuffer ana=new StringBuffer();

int flag=0;

int count=1;

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

if(chars[i]=='\"'){

flag++;

continue;

}

if(chars[i]!=' '){

ana.append(chars[i]);

}

if(chars[i]==' '&&flag%2!=0){

ana.append(chars[i]);

}

if(chars[i]==' '&&flag%2==0){

ana.append("\n");

count++;

}

}

System.out.println(count+"\n"+ana.toString());

}

}

}

# HJ75 公共字串计算

[公共子串计算](javascript:void(0);)

**题目描述**

给定两个只包含小写字母的字符串，计算两个字符串的最大公共子串的长度。

注：子串的定义指一个字符串删掉其部分前缀和后缀（也可以不删）后形成的字符串。

**输入描述:**

输入两个只包含小写字母的字符串

**输出描述:**

输出一个整数，代表最大公共子串的长度

示例1

**输入**

asdfas

werasdfaswer

**输出**

6

import java.io.\*;

public class Main{

public static void main(String[] args)throws Exception{

BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));

String str1 = "";

String str2 = "";

while((str1 = bf.readLine())!= null && (str2 = bf.readLine())!= null){

int max = 0;

char[] cha1 = str1.toCharArray();

char[] cha2 = str2.toCharArray();

for(int i = 0; i < str1.length(); i++){

for(int j = 0; j < str2.length(); j++){

int t1 = i;

int t2 = j;

int count = 0;

while(cha1[t1] == cha2[t2]){

t1++;

t2++;

count++;

max = Math.max(count,max);

if(t1 == cha1.length || t2 == cha2.length) break;

}

}

}

System.out.println(max);

}

}

}

# HJ76 尼科彻斯定理

[尼科彻斯定理](javascript:void(0);)

**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

输入一个正整数m（m≤100），将m的立方写成m个连续奇数之和的形式输出。

本题含有多组输入数据。

**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

public static 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 num = Integer.parseInt(str);

System.out.println(method(num));

}

}

public static String method(int m){

int temp = m\*m ;

StringBuilder sb = new StringBuilder() ;

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

if(m%2!=0){

sb.append(temp-2\*(m/2)+2\*i).append("+") ;

}else{

sb.append(temp-m+1+2\*i).append("+") ; ;

}

}

return sb.substring(0,sb.length()-1);

}

}

# HJ77 火车进站

火车进站

**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，火车站只有一个方向进出，同时停靠在火车站的列车中，只有后进站的出站了，先进站的才能出站。

要求输出所有火车出站的方案，以字典序排序输出。

**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

**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

**说明**

第一种方案：1进、1出、2进、2出、3进、3出

第二种方案：1进、1出、2进、3进、3出、2出

第三种方案：1进、2进、2出、1出、3进、3出

第四种方案：1进、2进、2出、3进、3出、1出

第五种方案：1进、2进、3进、3出、2出、1出

请注意，[3,1,2]这个序列是不可能实现的。

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);

}

}

}

# HJ80 整形数组合并

[整形数组合并](javascript:void(0);)

**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。

输出时相邻两数之间没有空格。

请注意本题有多组样例。

**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

解法一：

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Arrays;

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.length() > 0) {

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

int[] array1 = new int[strs.length];

for (int i = 0; i < array1.length; ++i)

array1[i] = Integer.parseInt(strs[i]);

line = br.readLine();

strs = br.readLine().split(" ");

int[] array2 = new int[strs.length];

for (int i = 0; i < array2.length; ++i)

array2[i] = Integer.parseInt(strs[i]);

System.out.println(combineBySort(array1, array2));

}

}

static String combineBySort(int[] array1, int[] array2) {

int[] outPut = new int[array1.length + array2.length];

Arrays.sort(array1);

Arrays.sort(array2);

int M = array1.length, R = array2.length;

int idx = 0, i = 0, j = 0;

if (array1[i] > array2[j]) {

outPut[idx++] = array2[j++];

} else if (array1[i] < array2[j]) {

outPut[idx++] = array1[i++];

} else {

outPut[idx++] = array1[i++];

j++;

}

while (i < M && j < R) {

if (array1[i] > array2[j]) {

if (outPut[idx - 1] != array2[j])

outPut[idx++] = array2[j];

++j;

} else if (array1[i] < array2[j]) {

if (outPut[idx - 1] != array1[i])

outPut[idx++] = array1[i];

++i;

} else {

if (outPut[idx - 1] != array1[i])

outPut[idx++] = array1[i];

++i;

++j;

}

}

if (i == M) {

while( j < R){

if (outPut[idx - 1] != array2[j])//去重

outPut[idx++] = array2[j];

j++;

}

} else {

for (; i < M; ++i)

if (outPut[idx - 1] != array1[i])

outPut[idx++] = array1[i];

}

StringBuilder sb = new StringBuilder();

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

sb.append(outPut[i]);

return sb.toString();

}

}

解法二：

import java.util.\*;

import java.io.\*;

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) {

StringBuilder sb = new StringBuilder();

int n = Integer.parseInt(str);

String[] str1 = br.readLine().split(" ");

int m =Integer.parseInt(br.readLine());

String[] str2 = br.readLine().split(" ");

int[] num = new int[n+m];

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

num[i]=Integer.parseInt(str1[i]);

}

for(int i =n; i<n+m;i++){

num[i]=Integer.parseInt(str2[i-n]);

}

Arrays.sort(num);

sb.append(num[0]);

for(int i =1;i<n+m;++i) {

if(num[i]==num[i-1]) {

continue;

}else {

sb.append(num[i]);

}

}

System.out.println(sb.toString());

}

}

}

# HJ81 字符串匹配

[字符串字符匹配](javascript:void(0);)

**题目描述**

判断短字符串中的所有字符是否在长字符串中全部出现。

请注意本题有多组样例输入。

**输入描述:**

输入两个字符串。第一个为短字符串，第二个为长字符串。两个字符串均由小写字母组成。

**输出描述:**

如果短字符串的所有字符均在长字符串中出现过，则输出true。否则输出false。

示例1

**输入**

bc

abc

**输出**

true

解法一：

import java.util.\*;

import java.io.\*;

public class Main{

public static void main(String[] args) throws IOException{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str="";

String str2="";

while((str=br.readLine())!=null&&!"".equals(str)){

str2=br.readLine();

char[] arr=str.toCharArray();

boolean judge=true;

for(char c:arr){

if(!str2.contains(String.valueOf(c))){

judge=false;

break;

}

}

System.out.println(judge);

}

br.close();

}

}

解法二：

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 s = br.readLine();

char[] ch1 = str.toCharArray();

boolean flag = true;

for (int i = 0; i < ch1.length; i++) {

if(s.indexOf(ch1[i])==-1){

flag = false;

break;

}

}

if(flag == true){

System.out.println(true);

}else{

System.out.println(false);

}

}

}

}

# HJ82 将真分数分解为埃及分数

[将真分数分解为埃及分数](javascript:void(0);)

**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。

注：真分数指分子小于分母的分数，分子和分母有可能gcd不为1！

如有多个解，请输出任意一个。

请注意本题含有多组样例输入！

**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

2/4

**输出**

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

1/3+1/6

**说明**

第二个样例直接输出1/2也是可以的

解法一：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

/\*\*

\* 真分数转埃及分数

\* 先化简

\* 步骤一： 用b除以a，得商数q1及余数r1,即b=a\*q1+r1

\* 步骤二： a/b=1/(q1+1）+(a-r1)/b(q1+1）

\* 步骤三： 重复步骤2，直到分解完毕

\* 3/7=1/3+2/21=1/3+1/11+1/231

\* @author Admin

\* @date 2020-12-20

\*/

public class Main {

public static void main(String[] args){

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

try {

while((str = br.readLine())!= null){

String[] strArr = str.split("\\/");

int a = Integer.parseInt(strArr[0]);

int b = Integer.parseInt(strArr[1]);

String[] resArr = new String[1];

f(a, b, "", resArr);

System.out.println(resArr[0]);

}

} catch (NumberFormatException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

public static void f(int a, int b, String resStr, String[] resArr){

if(a==1 || b%a==0){

int val = b/a;

resStr += 1+"/"+val;

resArr[0] = resStr;

return ;

}

else{

int q1 = b/a;

int r1 = b%a;

int val1 = q1+1;

resStr += 1+"/"+val1+"+";

a = a - r1;

b = b\*(q1+1);

f(a, b, resStr, resArr);

}

return;

}

}

解法二：

public static class Main {

/\*\*

\* 数学家斐波那契提出的一种求解\*\*\*分数的贪心算法，准确的算法表述应该是这样的：

\* 设某个真分数的分子为a，分母为b;

\* 把c=(b/a+1)作为分解式中第一个\*\*\*分数的分母；

\* 将a-b%a作为新的a；

\* 将b\*c作为新的b；

\* 如果a等于1，则最后一个\*\*\*分数为1/b,算法结束；

\* 如果a大于1但是a能整除b，则最后一个\*\*\*分数为1/(b/a),算法结束；

\* 否则重复上面的步骤。

\*\*/

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

while (in.hasNext()) {

StringBuilder res = new StringBuilder();

String[] arr = in.nextLine().split("/");

int a = Integer.parseInt(arr[0]);

int b = Integer.parseInt(arr[1]);

while (true) {

int c = b / a + 1;

res.append("1/");

res.append(c);

a = a - b % a;

b = b \* c;

res.append("+");

if (a == 1) {

res.append("1/");

res.append(b);

break;

} else if (a > 1 && b % a == 0) {

res.append("1/");

res.append(b / a);

break;

}

}

System.out.println(res);

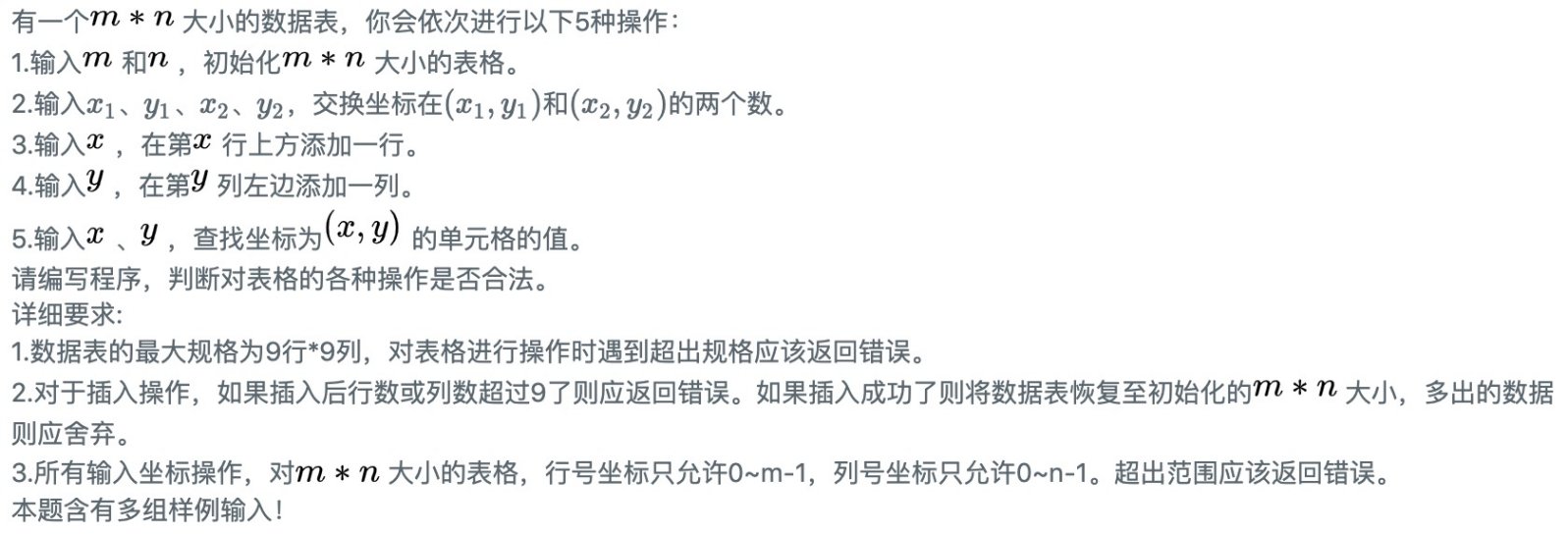
}

}

}

# HJ83 二维数组操作

[二维数组操作](javascript:void(0);)



**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 输入要查询的单元格的坐标

**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 输出查询单元格数据是否成功

示例1

**输入**

4 9

5 1 2 6

0

8

2 3

4 7

4 2 3 2

3

3

4 7

**输出**

0

-1

0

-1

0

0

-1

0

0

-1

**说明**

本组样例共有2组样例输入。

第一组样例：

1.初始化数据表为4行9列，成功

2.交换第5行1列和第2行6列的数据，失败。因为行的范围应该是(0,3)，不存在第5行。

3.在第0行上方添加一行，成功。

4.在第8列左边添加一列，失败。因为列的总数已经达到了9的上限。

5.查询第2行第3列的值，成功。

第二组样例：

1.初始化数据表为4行7列，成功

2.交换第4行2列和第3行2列的数据，失败。因为行的范围应该是(0,3)，不存在第4行。

3.在第3行上方添加一行，成功。

4.在第3列左边添加一列，成功。

5.查询第4行7列的值，失败。因为虽然添加了一行一列，但数据表会在添加后恢复成4行7列的形态，所以行的区间仍然在[0,3]，列的区间仍然在[0,6]，无法查询到(4,7)坐标。

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.\*;

public class Main {

public static void main(String[] args) throws IOException {

fx();

}

//region 83 二维数组操作

private static void fx() throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

while((str = br.readLine()) != null) {

String[] strs = str.split(" ");

//1. 行和列

int row = Integer.parseInt(strs[0]);

int column = Integer.parseInt(strs[1]);

if(row >= 0 && row <= 9 && column >= 0 && column <= 9) {

System.out.println("0");

} else {

System.out.println(-1);

}

//2.单元格的 行 列值

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

int rowValue1 = Integer.parseInt(values[0]);

int columnValue1 = Integer.parseInt(values[1]);

int rowValue2 = Integer.parseInt(values[2]);

int columnValue2 = Integer.parseInt(values[3]);

if(rowValue1 >= 0 && rowValue1 < row && rowValue2 >= 0 && rowValue2 < row

&& columnValue1 >= 0 && columnValue1 < column && columnValue2 >= 0 && columnValue2 < column) {

System.out.println(0);

} else {

System.out.println(-1);

}

//3. 插入的行 的值

int insertRowValue = Integer.parseInt(br.readLine());

if(insertRowValue >= 0 && insertRowValue < row && (row + 1) <= 9) {

System.out.println(0);

} else {

System.out.println(-1);

}

//4. 插入的列 值

int insertColumnValue = Integer.parseInt(br.readLine());

if(insertColumnValue >= 0 && insertColumnValue < column && (column + 1) <= 9) {

System.out.println(0);

} else {

System.out.println(-1);

}

//5. 运动轨迹的单元格

strs = br.readLine().split(" ");

int x = Integer.parseInt(strs[0]);

int y = Integer.parseInt(strs[1]);

if(x >= 0 && x < row && y >= 0 && y < column) {

System.out.println(0);

} else {

System.out.println(-1);

}

}

br.close();

}

}

# HJ84 统计大写字母个数

[统计大写字母个数](javascript:void(0);)

**题目描述**

找出给定字符串中大写字符(即'A'-'Z')的个数。

**输入描述:**

本题含有多组样例输入

对于每组样例，输入一行，代表待统计的字符串

**输出描述:**

对于每组样例，输出一个整数，代表字符串中大写字母的个数

示例1

**输入**

add123#$%#%#O

150175017(&^%&$vabovbao

**输出**

1

0

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

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) {

char[] chars = str.toCharArray();

int count = 0;

for(int i = 0; i < chars.length; i++)

if(65 <= chars[i] && chars[i] <= 90) count ++;

System.out.println(count);

}

}

}

# HJ85 字符串运用-密码截取

[最长回文子串](javascript:void(0);)

**题目描述**

给定一个仅包含小写字母的字符串，求它的最长回文子串的长度。

所谓回文串，指左右对称的字符串。

所谓子串，指一个字符串删掉其部分前缀和后缀（也可以不删）的字符串

（注意：记得加上while处理多个测试用例）

**输入描述:**

输入一个仅包含小写字母的字符串

**输出描述:**

返回最长回文子串的长度

示例1

**输入**

cdabbacc

**输出**

4

**说明**

abba为最长的回文子串

import java.io.\*;

public class Main{

public static void main(String[] args)throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

while((str = br.readLine())!=null && str.length()!=0){

char[] chars = str.toCharArray();

int n = chars.length;

int max = 1;

for(int i = 0; i < n-1; i++){//奇

int left = i - 1;

int right = i + 1;

int length = 1;

while(left>=0 && right<n && chars[left]==chars[right]){

length += 2;

left--;

right++;

}

max = max > length ? max : length;

}

for(int i = 0; i < n-1; i++){//偶

int length = 1;

if(chars[i]==chars[i+1]){

length = 2;

}else

continue;

int left = i - 1;

int right = i + 2;

while(left>=0 && right<n && chars[left]==chars[right]){

length += 2;

left--;

right++;

}

max = max > length ? max : length;

}

System.out.println(max);

}

}

}

# HJ86 求最大连续bit数

[求最大连续bit数](javascript:void(0);)

**题目描述**

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

本题含有多组样例输入。

**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

5

**输出**

2

1

**说明**

3的二进制表示是11，最多有2个连续的1。

5的二进制表示是101，最多只有1个连续的1。

import java.io.\*;

import java.util.\*;

public class Main{

public static void main(String[] args)throws IOException{

BufferedReader bf=new BufferedReader(new InputStreamReader(System.in));

String value;

while((value=bf.readLine())!=null){

int num=Integer.parseInt(value);

char[] chars=Integer.toBinaryString(num).toCharArray();

int[] nums=new int[chars.length];

for(int index=0;index<chars.length;index++){

char chat=chars[index];

if(chat=='1'){

if(index==0)

nums[index]=1;

else{

nums[index]=nums[index-1]+1;

nums[index-1]=0;

}

}

}

Arrays.sort(nums);

System.out.println(nums[nums.length-1]);

}

}

}

# HJ87 密码强度等级

[密码强度等级](javascript:void(0);)

**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\_SECURE

SECURE,

VERY\_STRONG,

STRONG,

AVERAGE,

WEAK,

VERY\_WEAK,

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)

**输入描述:**

本题含有多组输入样例。

每组样例输入一个string的密码

**输出描述:**

每组样例输出密码等级

示例1

**输入**

38$@NoNoNo

123

**输出**

VERY\_SECURE

WEAK

**说明**

第一组样例密码强度为95分。

第二组样例密码强度为25分。

import java.io.\*;

import java.util.\*;

public class Main {

public static void main (String[] args) throws IOException{

BufferedReader bread= new BufferedReader(new InputStreamReader(System.in));

String kywd;

while((kywd = bread.readLine())!=null){

System.out.println(getScore(kywd));

}

}

public static String getScore(String kywd){

char[] kychars = kywd.toCharArray();

int numNum = 0;

int sinNum = 0;

int upNum=0;

int lowNum=0;

int score = 0;

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

if(kychars[i]>0x2F&&kychars[i]<0x3A){

numNum++;

}else if(kychars[i]>0x40&&kychars[i]<0x5B){

upNum++;

}else if(kychars[i]>0x60&&kychars[i]<0x7B){

lowNum++;

}else if((kychars[i]>0x20&&kychars[i]<0x30)||(kychars[i]>0x39&&kychars[i]<0x41)||(kychars[i]>0x5A&&kychars[i]<0x61)||(kychars[i]>0x7A&&kychars[i]<0x7F)){

sinNum++;

}

}

/\*判断长度\*/

if(kychars.length<5){

score+=5;

}else if(kychars.length>4&&kychars.length<8){

score+=10;

}else if(kychars.length>7){

score+=25;

}

/\*判断字母\*/

if(upNum==0&&lowNum==0){

}

if(upNum>0){

score+=10;

}

if(lowNum>0){

score+=10;

}

/\*判断数字\*/

if(numNum==0){

}

if(numNum>0){

score+=10;

}

if(numNum>1){

score+=10;

}

/\*判断符号\*/

if(sinNum==0){

}

if(sinNum>0){

score+=10;

}

if(sinNum>1){

score+=15;

}

/\*奖励\*/

if((upNum==0||lowNum==0)&&(numNum>0)){

score+=2;

if(sinNum>0){

score+=1;

if(upNum==0&&lowNum==0){

score+=2;

}

}

}

/\*评价结果\*/

if(score>=90){

return "VERY\_SECURE";

}

if(score>=80){

return "SECURE";

}

if(score>=70){

return "VERY\_STRONG";

}

if(score>=60){

return "STRONG";

}

if(score>=50){

return "AVERAGE";

}

if(score>=25){

return "WEAK";

}

return "VERY\_WEAK";

}

}

# HJ88 扑克牌大小

[扑克牌大小](javascript:void(0);)

**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

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class Main{

public static void main(String []args) throws Exception{

BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));

String str;

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(" ");

str0 = findString(str0);

str1 = findString(str1);

int len0 = str0.length;

int len1 = str1.length;

if(len0 == 4 || len1 == 4){

if(len0 == 4 && len1 == 4){

int temp = Integer.parseInt(str0[0]) - Integer.parseInt(str1[0]);

if(temp >=0){

System.out.println(ss[0]);

continue;

}

}else if(len0 == 4){

System.out.println(ss[0]);

continue;

}else{

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

continue;

}

}

if(len0 != len1){

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;

}

}

}

private static String[] findString(String[] str){

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

switch(str[i]){

case "J":

str[i] = "11";

break;

case "Q":

str[i] = "12";

break;

case "K":

str[i] = "13";

break;

case "A":

str[i] = "14";

break;

case "2":

str[i] = "15";

break;

}

}

return str;

}

}

# HJ89 24点运算

[24点运算](javascript:void(0);)

**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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

private static String[] op = new String[]{"+","-","\*","/"};

public static void main(String[] args) throws IOException {

BufferedReader re=new BufferedReader(new InputStreamReader(System.in));

String input;

while((input = re.readLine()) != null && !"".equals(input)){

String[] ss = input.split(" ");

int a = getInputNum(ss[0]);

int b = getInputNum(ss[1]);

int c = getInputNum(ss[2]);

int d = getInputNum(ss[3]);

// 只要有joker，直接返回ERROR

if(a==-1||b==-1||c==-1||d==-1){

System.out.println("ERROR");

continue;

}

compute(a,b,c,d);

}

}

/\*\*

\* 24点计算方法穷举

\* @param a

\* @param b

\* @param c

\* @param d

\*/

public static void compute(int a,int b,int c,int d) {

int[] arr={a,b,c,d};

// 运算符穷举数组

String[][] arr1 = symbol();

for(int i=0;i<4;i++){// 第一个数字

for(int j=0;j<4;j++){// 第二个数字

for(int k=0;k<4;k++){// 第三个数字

for(int p=0;p<4;p++){// 第四个数字

if((i!=j)&&(i!=k)&&(i!=p)&&(j!=k)&&(j!=p)&&(k!=p)){// 如果四个数字互不相等才计算，不然一个字符就会出现两次

// 遍历运算符穷举数组

for(String[] str:arr1){

// 依次计算，得出最终结论

int sum = sumNum(arr[i], arr[j], str[0]);

sum=sumNum(sum, arr[k], str[1]);

sum=sumNum(sum, arr[p], str[2]);

if(sum==24){

// 如果结果等于24，返回结果

String str1=change2(arr[i])+str[0]+change2(arr[j])+str[1]+change2(arr[k])+str[2]+change2(arr[p])+"";

System.out.println(str1);

return;

}

}

}

}

}

}

}

// 穷举之后仍然没有结果，返回none

System.out.println("NONE");

}

/\*\*

\* 穷举所有可能的运算符组合

\* @return

\*/

public static String[][] symbol() {

//运算符共三个，每个四种可能性，4\*4\*4中运算符组合，每个组合有三个运算符

String[][] symbol = new String[64][3];

int p =0;

for(int i=0;i<4;i++){// 第一个运算符

for(int j=0;j<4;j++){// 第二个运算符

for(int k=0;k<4;k++){// 第三个运算符

symbol[p++]=new String[]{op[i],op[j],op[k]};

}

}

}

return symbol;

}

/\*\*

\* 两个数字计算结果

\* @param a

\* @param b

\* @param symb

\* @return

\*/

public static int sumNum(int a, int b, String symb) {

switch(symb){

case "+":

return a+b;

case "-":

return a-b;

case "\*":

return a\*b;

case "/":

return a/b;

default:

return 0;

}

}

/\*\*

\* 字符串转数字

\* @param str

\* @return

\*/

public static int getInputNum(String str){

switch(str.toUpperCase()){

case "A":

return 1;

case "J":

return 11;

case "Q":

return 12;

case "K":

return 13;

case "JOKER":

return -1;

default:

return Integer.parseInt(str);

}

}

/\*\*

\* 数字转化为字符串

\* @param i

\* @return

\*/

public static String change2(int i) {

switch(i){

case 1:

return "A";

case 11:

return "J";

case 12:

return "Q";

case 13:

return "K";

default:

return String.valueOf(i);

}

}

}

# HJ90 合法IP

[合法IP](javascript:void(0);)

**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。

注意本题有多组样例输入。

**输入描述:**

输入一个ip地址，保证是xx.xx.xx.xx的形式（xx为整数）

**输出描述:**

返回判断的结果YES or NO

示例1

**输入**

10.138.15.1

255.0.0.255

255.255.255.1000

**输出**

YES

YES

NO

import java.io.\*;

import java.util.\*;

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[] subIP = str.split("\\.");

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

Integer intIP = Integer.valueOf(subIP[i]);

if(intIP >= 0 && intIP<=255){

if(i == subIP.length-1){

System.out.println("YES");

}

continue;

} else{

System.out.println("NO");

break;

}

}

}

}

}

# HJ91 201301 JAVA 题目2-3级

[走方格的方案数](javascript:void(0);)

**题目描述**

请计算n\*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。

本题含有多组样例输入。

**输入描述:**

每组样例输入两个正整数n和m，用空格隔开。(1≤n,m≤8)

**输出描述:**

每组样例输出一行结果

示例1

**输入**

2 2

1 2

**输出**

6

3

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

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) {

int n=Integer.valueOf(line.substring(0,line.indexOf(" ")));

int m=Integer.valueOf(line.substring(line.indexOf(" ")+1));

System.out.println(getCount(n, m));

}

}

/\*\*

\* 获取路线数量

\* @param n

\* @param m

\* @return

\*/

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){

// 边上的每一个点的可能性都是1，因为走到边上，就只能顺着边往下走了，可能性只能是1

dp[i][j]=1;

}else{

// 往后每一个点，都是后面两个点的可能性之和，因为它可以选择任意一个点来走，可能性就是下两个点的可能性相加

dp[i][j]=dp[i][j-1]+dp[i-1][j];

}

}

}

// 最终加到最后一个点，可能性就是所有的路线数量

return dp[n][m];

}

}

# HJ92 在字符串中找出连续最长的数字串

[在字符串中找出连续最长的数字串](javascript:void(0);)

**题目描述**

输入一个字符串，返回其最长的数字子串，以及其长度。若有多个最长的数字子串，则将它们全部输出（按原字符串的相对位置）

本题含有多组样例输入。

**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，中间用逗号间隔。如果有相同长度的串，则要一块儿输出（中间不要输出空格）。

示例1

**输入**

abcd12345ed125ss123058789

a8a72a6a5yy98y65ee1r2

**输出**

123058789,9

729865,2

解法一：

import java.io.\*;

import java.util.\*;

import java.lang.\*;

public class Main{

public static void main(String[] args)throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String s;

while((s=br.readLine())!=null){

char[] ch = s.toCharArray();

int max= Integer.MIN\_VALUE;

char[] temp = new char[ch.length];

int y = 0;

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

if(ch[i]>='0'&&ch[i]<='9'){

int count = 1;

for(int j=i+1;j<ch.length;j++,i++){

if(ch[j]>='0'&&ch[j]<='9'){

count++;

}

else{

break;

}

}

if(count>max){

int x=0;

max = count;

for(int k=i+1-max;k<=i;k++){

temp[x++] = ch[k];

y = x;

}

}

else if(count==max){

for(int k=i+1-max;k<=i;k++){

temp[y++] = ch[k];

}

}

}

}

if(y==0){

System.out.println(null+","+0);

}

else{

for(int z=0;z<y-1;z++){

System.out.print(temp[z]);

}

System.out.println(temp[y-1]+","+max);

}

}

}

}

解法二：

public static class Main{

//使用正则

public static void main(String[] args) throws Exception{

Scanner sc = new Scanner(System.in);

while(sc.hasNextLine()){

String line = sc.nextLine();

String[] ss = line.split("[^0-9]+");

int max = 0;

ArrayList<String> list = new ArrayList<>();

for(String s : ss){

if(s.length() > max){

max = s.length();

//清空集合

list.clear();

list.add(s);

}else if(s.length() == max){

list.add(s);

}

}

StringBuilder sb = new StringBuilder();

for(String item : list){

sb.append(item);

}

sb.append(",").append(max);

System.out.println(sb);

}

}

}

# HJ93 201301 JAVA题目0-1级

数组分组

**题目描述**

输入int型数组，询问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，输出true；不满足时输出false。

本题含有多组样例输入。

**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

3

3 5 8

**输出**

true

**说明**

第一个样例：

第一组：5 -5 1

第二组：1

第二个样例：由于3和5不能放在同一组，所以不存在一种分法。

解法一：

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.Arrays;0

import java.util.List;

import java.io.IOException;

/\*\*

\* 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）

\*/

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader read = new BufferedReader(new InputStreamReader(System.in));

String input;

while ((input = read.readLine()) != null){

int n = Integer.parseInt(input);

int[] array = new int[n];

String[] strs = read.readLine().split(" ");

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

array[i] = Integer.parseInt(strs[i]);

}

System.out.println(solution(array));

}

}

public static boolean solution(int[] array) {

if(array.length < 2)

return false;

int sum = 0, sum3 = 0;

ArrayList<Integer> list = new ArrayList<>();

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

sum += array[i];

if(array[i] % 3 == 0 && array[i] != 0)

sum3 += array[i];

else if(array[i] % 5 == 0 && array[i] != 0)

continue;

else

list.add(array[i]);

}

if(sum % 2 != 0)

return false;

int num1 = sum / 2 - sum3; //第一个数组差的数

return dfs(list, num1, 0);

//在剩下的数字中找到和为num1的，如果没有返回false

}

private static boolean dfs(List<Integer> bag, int target, int i) {

if (i == bag.size())

return target == 0;

// dfs搜索，对每个元素，选择或者放弃

return dfs(bag, target, i + 1) || dfs(bag, target - bag.get(i), i + 1);

}

}

解法二：

public static class Main {

/\*\*

\* HJ93 201301 JAVA题目0-1级

\* 数组分组

\* 题目描述

\* 输入int型数组，询问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，输出true；不满足时输出false。

\* 本题含有多组样例输入。

\* 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

\* 3

\* 3 5 8

\* 输出

\* true

\* 说明

\* 第一个样例：

\* 第一组：5 -5 1

\* 第二组：1

\* 第二个样例：由于3和5不能放在同一组，所以不存在一种分法。

\*/

public static void main(String[] args) {

/\*

思路：

先把三和五的倍数都挑出来，算好两边的和sum3和sum5，所有数总和为sum，求出target = sum/2 - sum3，把剩余的数放在一起，转化成在一个list中找能不能凑出target

\*/

Scanner in = new Scanner(System.in);

while (in.hasNext()) {

LinkedList<Integer> list = new LinkedList<>();

int n = in.nextInt();

//这里用不上sum5

int sum3 = 0;

int sum = 0;

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

int cur = in.nextInt();

if (cur % 3 == 0) sum3 += cur;

else list.add(cur);

sum += cur;

}

int target = sum / 2 - sum3;

if (sum % 2 != 0) System.out.println("false");

else System.out.println(helper(list, target));

}

}

private static boolean helper(LinkedList<Integer> list, int target) {

return helper(0, list, target);

}

private static boolean helper(int l, LinkedList<Integer> list, int target) {

if (l == list.size()) return target == 0;

return helper(l + 1, list, target - list.get(l)) || helper(l + 1, list, target);

}

}

# HJ94 记票统计

记票统计

**题目描述**

请实现一个计票统计系统。你会收到很多投票，其中有合法的也有不合法的，请统计每个候选人得票的数量以及不合法的票数。

本题有多组样例输入。

**输入描述:**

输入候选人的人数n，第二行输入n个候选人的名字（均为大写字母的字符串），第三行输入投票人的人数，第四行输入投票。

**输出描述:**

按照输入的顺序，每行输出候选人的名字和得票数量，最后一行输出不合法的票数。

示例1

**输入**

4

A B C D

8

A D E CF A GG A B

**输出**

A : 3

B : 1

C : 0

D : 1

Invalid : 3

import java.io.\*;

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 candidate = Integer.parseInt(str);

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

int voter = Integer.parseInt(br.readLine());

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

int invalid = 0;

int[] number = new int[candidate];

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

for(int j =0; j<vote.length; j++){

if(name[i].equals(vote[j])){

number[i]++;

}

}

voter -= number[i];

}

invalid = voter;

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

System.out.println(name[i] + " : " + number[i]);

}

System.out.println("Invalid : " + invalid);

}

}

}

# HJ95 人民币转换

[人民币转换](javascript:void(0);)

**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。

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。

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“。

4、10应写作“拾”，100应写作“壹佰”。例如，1010.00应写作“人民币壹仟零拾元整”，110.00应写作“人民币壹佰拾元整”

5、十万以上的数字接千不用加“零”，例如，30105000.00应写作“人民币叁仟零拾万伍仟元整”

本题含有多组样例输入。

**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

10012.02

**输出**

人民币拾伍万壹仟壹佰贰拾壹元壹角伍分

人民币壹万零拾贰元贰分

import java.util.\*;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

public class Main{

private static String[] chineseNum = {"零","壹","贰","叁","肆","伍","陆","柒","捌","玖"};

private static String[] chineseUnit = {null,"拾","佰","仟","万","拾","佰","仟","亿"};

public static void main(String[] args)throws IOException{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str;

while((str=br.readLine())!=null){

String zn;

String sn=null;

if(str.contains(".")){

String[] split=str.split("\\.");

zn=split[0];

sn=split[1];

}else{

zn=str;

}

String res=format(zn);

StringBuilder sb=new StringBuilder(res);

if(!"人民币".equals(res)){

sb.append("元");

}

if(sn==null||"00".equals(sn))

sb.append("整");

else{

int jn=Integer.parseInt(String.valueOf(sn.charAt(0)));

if(jn!=0){

sb.append(chineseNum[jn]);

sb.append("角");

}

int fn=Integer.parseInt(String.valueOf(sn.charAt(1)));

if(fn!=0){

sb.append(chineseNum[fn]);

sb.append("分");

}

}

System.out.println(sb.toString());

}

}

public static String format(String zn){

StringBuilder sb=new StringBuilder("人民币");

boolean hasZero=false;

if("0".equals(zn)){

return sb.toString();

}

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

int unitIndex=zn.length()-i-1;

int n=Integer.parseInt(String.valueOf(zn.charAt(i)));

switch(n){

case 0:

if(!hasZero&&unitIndex!=4){

sb.append(chineseNum[0]);

hasZero=true;

}

break;

case 1:

if(!"拾".equals(chineseUnit[unitIndex])){

sb.append(chineseNum[1]);

}

hasZero=false;

break;

default:

sb.append(chineseNum[n]);

hasZero=false;

break;

}

if(chineseUnit[unitIndex]!=null){

if(!hasZero) sb.append(chineseUnit[unitIndex]);

}

}

if(sb.charAt(sb.length()-1)=='零')

return sb.substring(0,sb.length()-1);

else

return sb.toString();

}

}

# HJ96 表示数字

[表示数字](javascript:void(0);)

**题目描述**

将一个字符中所有的整数前后加上符号“\*”，其他字符保持不变。连续的数字视为一个整数。

注意：本题有多组样例输入。

**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

5151

**输出**

Jkdi\*234\*klowe\*90\*a\*3\*

\*5151\*

import java.io.\*;

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){

char[] c = str.toCharArray();

StringBuilder sb = new StringBuilder();

int i = 0;

while(i < c.length){

if (c[i] >= '0' && c[i] <= '9'){

sb.append("\*");

sb.append(c[i]);

while(++i < c.length && c[i] >= '0' && c[i] <= '9'){

sb.append(c[i]);

}

sb.append("\*");

}else{

sb.append(c[i]);

i++;

}

}

System.out.println(sb.toString());

}

}

}

# HJ97 记负均正

[记负均正](javascript:void(0);)

**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。

本题有多组输入用例。

**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 3.0

import java.io.\*;

import java.util.\*;

/\*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。\*/

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);

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

int negative=0;

int positive=0;

int sum=0;

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

int num=Integer.parseInt(nums[i]);

if(num<0){

negative++;

}else if(num>0){

sum+=num;

positive++;

}

}

System.out.println(negative+" "+Math.round(sum\*10.0/positive)/10.0);

}

}

}

# HJ98 自动售货系统

[自动售货系统](javascript:void(0);)

**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。



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。



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

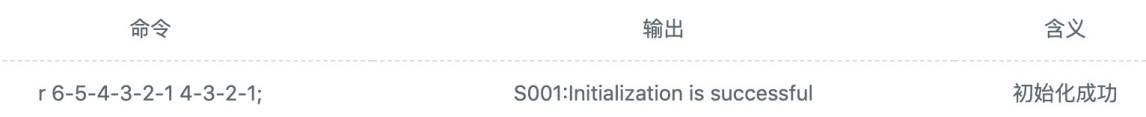


商品和各种面额钱币取值范围只是作为初始化命令的限制，其它场景下不限制取值范围；考试框架已经实现取值范围的检查，考生不需要关注。

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：



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”；

（4） 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 > E005；

输出说明：如果投币成功，输出“S002:Pay success,balance=X”。

例：



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:



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；

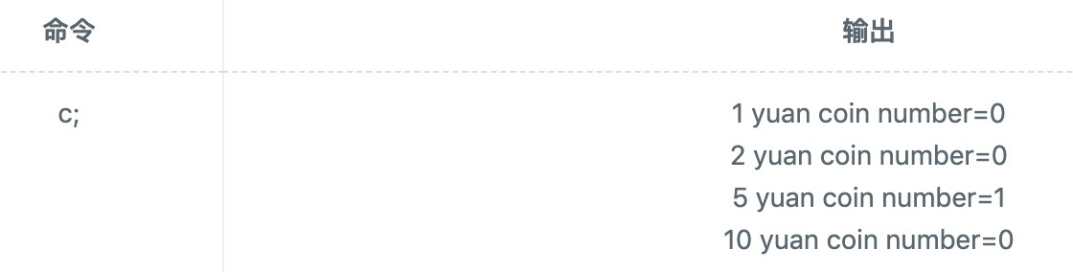
约束说明：

（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:



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:



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：



“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：



**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 22-18-21-21-7-20 3-23-10-6;c;q0;p 1;b A6;c;b A5;b A1;c;q1;p 5;

r 28-12-11-1-16-10 19-30-8-11;b A1;p 1;

**输出**

S001:Initialization is successful

E009:Work failure

E010:Parameter error

S002:Pay success,balance=1

E008:Lack of balance

1 yuan coin number=1

2 yuan coin number=0

5 yuan coin number=0

10 yuan coin number=0

E008:Lack of balance

E008:Lack of balance

E009:Work failure

E010:Parameter error

S002:Pay success,balance=5

S001:Initialization is successful

E008:Lack of balance

S002:Pay success,balance=1

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[] strs=str.split(";");

int[] price=new int[] {2,3,4,5,8,6};

int[] goods=new int[6];

int[] money=new int[4];

int sum=0;

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

String s=strs[i];

if(s.startsWith("r ")) {

String[] ss=s.substring(2).replace(" ", "-").split("-");

for(int j=0;j<goods.length;j++) {

goods[j]=Integer.parseInt(ss[j]);

}

for(int j=0;j<money.length;j++) {

money[j]=Integer.parseInt(ss[j+goods.length]);

}

System.out.println("S001:Initialization is successful");

}else if(s.startsWith("p ")) {

int input=Integer.parseInt(s.substring(2));

if(input !=1&&input!=2&&input!=5&&input!=10) {

System.out.println("E002:Denomination error");

continue;

}

if((money[0]+money[1]\*2<input)&&(input==5||input==10)) {

System.out.println("E003:Change is not enough, pay fail");

continue;

}

boolean flag=false;

for(int g:goods) {

if(g!=0) {

flag=true;

}

}

if(!flag) {

System.out.println("E005:All the goods sold out");

continue;

}

sum+=input;

if(input==10) {

money[3]++;

}else if(input==5) {

money[2]++;

}else if(input==2) {

money[1]++;

}else if(input==1) {

money[0]++;

}

System.out.println("S002:Pay success,balance="+sum);

continue;

}else if(s.startsWith("b ")) {

String temp=s.substring(2);

if(!"A1".equals(temp)&&!"A2".equals(temp)&&!"A3".equals(temp)&&!"A4".equals(temp)&&!"A5".equals(temp)&&!"A6".equals(temp)) {

System.out.println("E006:Goods does not exist");

continue;

}

int index=Integer.parseInt(temp.substring(1))-1;

if(goods[index]==0) {

System.out.println("E007:The goods sold out");

continue;

}

if(price[index]>sum) {

System.out.println("E008:Lack of balance");

continue;

}

goods[index]--;

sum-=price[index];

System.out.println("S003:Buy success,balance="+sum);

continue;

}else if(s.startsWith("c")){

if(sum==0) {

System.out.println("E009:Work failure");

continue;

}else {

StringBuilder sb=new StringBuilder();

if(sum/10>=money[3]) {

sum-=money[3]\*10;

sb.append("10 yuan coin number=").append(money[3]);

money[3]=0;

}else {

money[3]-=sum/10;

sb.append("10 yuan coin number=").append(sum/10);

sum-=(sum/10)\*10;

}

if(sum/5>=money[2]) {

sum-=money[2]\*5;

sb.insert(0,"\n").insert(0, money[2]).insert(0, "5 yuan coin number=");

money[2]=0;

}else {

money[2]-=sum/5;

sb.insert(0,"\n").insert(0, sum/5).insert(0, "5 yuan coin number=");

sum-=(sum/5)\*5;

}

if(sum/2>=money[1]) {

sum-=money[1]\*2;

sb.insert(0,"\n").insert(0, money[1]).insert(0, "2 yuan coin number=");

money[1]=0;

}else {

money[1]-=sum/2;

sb.insert(0,"\n").insert(0, sum/2).insert(0, "2 yuan coin number=");

sum-=(sum/2)\*2;

}

if(sum>=money[0]) {

sum-=money[0];

sb.insert(0,"\n").insert(0, money[0]).insert(0, "1 yuan coin number=");

money[0]=0;

}else {

money[2]-=sum;

sb.insert(0,"\n").insert(0, sum).insert(0, "1 yuan coin number=");

sum=0;

}

System.out.println(sb);

sum=0;

continue;

}

}else if(s.startsWith("q")) {

if("q 0".equals(s)){

StringBuilder sb=new StringBuilder();

for(int j=0;j<goods.length;j++) {

sb.append("A").append(j+1).append(" ").append(goods[j]).append(" ").append(price[j]).append("\n");

}

System.out.print(sb);

continue;

}else if("q 1".equals(s)) {

System.out.println("1 yuan coin number="+money[0]+"\n"+"2 yuan coin number="+money[1]+"\n"+"5 yuan coin number="+money[3]+"\n"+"10 yuan coin number="+money[4]);

continue;

}else {

System.out.println("E010:Parameter error");

}

}

}

}

}

}

# HJ99 自守数

[自守数](javascript:void(0);)

**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

**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

import java.io.\*;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = null;

while ((str = br.readLine()) != null) {

int n=Integer.valueOf(str);

int count=0;

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

int temp=i;

int j=1;

while(temp!=0){

temp=temp/10;

j=j\*10;

}

if((i\*i-i)%j==0){

count++;

}

}

System.out.println(count);

}

}

}

# HJ100 等差数列

[等差数列](javascript:void(0);)

**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

本题为多组输入，请使用while(cin>>)等形式读取数据

**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

import java.io.\*;

public class Main{

public static void main(String[] args) throws Exception{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = "";

while((str=br.readLine())!=null){

int n = Integer.parseInt(str);

System.out.println((3\*n\*n+n)/2);

}

}

}

# HJ101 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

[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](javascript:void(0);)

**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）

本题有多组输入，请使用while(cin>>)处理

**输入描述:**

第一行输入数组元素个数

第二行输入待排序的数组，每个数用空格隔开

第三行输入一个整数0或1。0代表升序排序，1代表降序排序

**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

5

1 2 3 4 5

1

**输出**

1 2 3 4 9 25 55 64

5 4 3 2 1

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String inputCount;

while ((inputCount = br.readLine()) != null) {

int count = Integer.parseInt(inputCount);

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

int flag = Integer.parseInt(br.readLine());

int[] num = new int[input.length];

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

num[i] = Integer.parseInt(input[i]);

}

quickSort(num,0,num.length - 1);

StringBuilder sb = new StringBuilder();

if(flag == 0){

for (int j = 0; j < num.length; j++) {

sb.append(num[j]).append(" ");

}

}else{

for (int k = num.length - 1; k >= 0; k--) {

sb.append(num[k]).append(" ");

}

}

System.out.println(sb.substring(0,sb.length()-1));

}

}

public static void quickSort(int[] num, int L, int R) {

if (L >= R) {

return;

}

int p = partition(num, L, R);

quickSort(num, L, p - 1);

quickSort(num, p+1, R);

}

public static int partition(int[] num, int L, int R) {

int key = num[L];

int pivot = L;

for (int i = L + 1; i <= R; i++) {

if (num[i] < key) {

int temp = num[++pivot];

num[pivot] = num[i];

num[i] = temp;

}

}

int tt = num[pivot];

num[pivot] = num [L];

num[L] = tt;

return pivot;

}

}

# HJ102 字符统计

[字符统计](javascript:void(0);)

**题目描述**

输入一个只包含小写英文字母和数字的字符串，按照不同字符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ASCII码由小到大排序输出。

本题含有多组样例输入

**输入描述:**

一个只包含小写英文字母和数字的字符串。

**输出描述:**

一个字符串，为不同字母出现次数的降序表示。若出现次数相同，则按ASCII码的升序输出。

示例1

**输入**

aaddccdc

1b1bbbbbbbbb

**输出**

cda

b1

**说明**

第一个样例里，c和d出现3次，a出现2次，但c的ASCII码比d小，所以先输出c，再输出d，最后输出a.

public static 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) {

char[] chArr = str.toCharArray();

int[] temp = new int[150];

for (char c : chArr) {

temp[c]++;

}

int max = 0;

for (int i : temp) {

if (max < i) {

max = i;

}

}

StringBuilder sbf = new StringBuilder();

while (max != 0) {

for (int j = 0; j < temp.length; j++) {

if (temp[j] == max) {

sbf.append((char) j);

}

}

max--;

}

System.out.println(sbf);

}

}

}

# HJ103 Redraiment的走法

Redraiment的走法

**题目描述**

Redraiment是走梅花桩的高手。Redraiment可以选择任意一个起点，从前到后，但只能从低处往高处的桩子走。他希望走的步数最多，你能替Redraiment研究他最多走的步数吗？

本题含有多组样例**输入**

**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

3

3 2 1

**输出**

3

1

**说明**

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。

import java.io.\*;

import java.util.\*;

public static class Main {

public static void main(String[] args) throws IOException {

BufferedReader read = new BufferedReader(new InputStreamReader(System.in));

String input;

while ((input = read.readLine()) != null) {

int num = Integer.parseInt(input);

String[] strs = read.readLine().split(" ");

int[] nums = new int[num];

int max = 0;

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

nums[i] = Integer.parseInt(strs[i]);

}

int[] result = new int[num];

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

result[i] = 1;

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

if (nums[j] < nums[i]) {

result[i] = Math.max(result[i], result[j] + 1);

}

}

}

max = 1;

for (int j : result) {

if (j > max) {

max = j;

}

}

System.out.println(max);

}

}

}

# HJ105 记负均正II

[记负均正II](javascript:void(0);)

**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，结果保留一位小数，如果没有非负数，则平均值为0

本题有多组输入数据，输入到文件末尾，请使用while(cin>>)读入数据范围小于1e6

**输入描述:**

输入任意个整数，每行输入一个。

**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

import java.io.\*;

import java.util.\*;

public class Main{

public static void main(String[] args) throws IOException{

BufferedReader br= new BufferedReader(new InputStreamReader(System.in));

String str;

int count=0;

int total=0;

int countNot=0;

while((str=br.readLine())!=null){

int num=Integer.parseInt(str);

if(num<0){

count++;

}else{

total+=num;

countNot++;

}

}

System.out.println(count);

if(countNot==0){

System.out.println(0.0);

}else{

long s=Math.round(total\*10.0/countNot);

System.out.println(s/10+"."+s%10);

}

}

}

# HJ106 字符逆序

[字符逆序](javascript:void(0);)

**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。

**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

import java.io.IOException;

public class Main {

public static void main(String[] args) throws IOException {

int length = System.in.available();

if(length > 2) {

byte[] bts = new byte[length-1];

int flag = System.in.read(bts);

if(flag!= -1) {

byte[] tmp = new byte[bts.length];

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

tmp[i] = bts[tmp.length - i - 1];

}

System.out.println(new String(tmp));

}

}

}

}

# HJ107 求解立方根

[求解立方根](javascript:void(0);)

**题目描述**

计算一个数字的立方根，不使用库函数。

保留一位小数。

**输入描述:**

待求解参数，为double类型（一个实数）

**输出描述:**

输入参数的立方根。保留一位小数。

示例1

**输入**

216

**输出**

6.0

方法一：

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));

double input=Double.parseDouble(br.readLine());

boolean flag=false;

if(input<0) {

flag=true;

input=-input;

}

double distance=input;//误差或者增加的距离

double index=0;

double last=0;

while(true) {

last=index\*index\*index;

if(last>input) {

index-=distance;

distance/=10;

}

if(distance<0.001) {

break;

}

index+=distance;

}

double result=(int)((index+0.05)\*10)/10.0;

if(flag) {

result=0-result;

}

System.out.println(result);

}

}

方法二：

public static class Main {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str;

while ((str = br.readLine()) != null) {

double d = Double.parseDouble(str);

System.out.println(getCubeRoot(d));

}

}

public static double getCubeRoot(double num) {

boolean f = false;

boolean l = false;

if (num < 0) {

f = true;

num = num \* (-1);

}

if (num > 0 && num < 1) {

l = true;

num = 1 / num;

}

double start = 0.0;

double end = num;

double middel = start;

while (start < end) {

middel = (start + end) / 2;

double r = middel \* middel \* middel;

double diff = Math.abs(r - num);

if (diff < 0.0001) {

break;

} else if (r > num) {

end = middel;

} else {

start = middel;

}

}

if (f) {

middel = middel \* -1;

}

if (l) {

middel = 1 / middel;

}

//Math.round对正数四舍五入

return Math.round(middel \* 10.0) / 10.0;

}

}

# HJ108 求最小公倍数

[求最小公倍数](javascript:void(0);)

**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

import java.io.\*;

import java.util.\*;

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[] strArr = str.split(" ");

int n = Integer.parseInt(strArr[0]);

int m = Integer.parseInt(strArr[1]);

int j = m\*n;

if(n>m){

int temp = m;

m = n;

n =temp;

}

//最大公约数

while(n!=0){

int r = m%n;

m = n;

n = r;

}

int max = j/m;

System.out.println(max);

}

}

}