2019-01-27PAT乙级95题总结

类型一：模拟类

按照题目要求编码，主要考察**代码能力**，先列出实际操作中较难题目。

题号	题目	说明
B1050	螺旋矩阵（25 分）	逻辑，二维vector
B1068	万绿丛中一点红（20 分）	涉及图的周围八个方向索引技巧
B1088	三人行（20 分）	类型强转和自动转化，题目有小bug

1050 螺旋矩阵 （25 分）

本题要求将给定的 N 个正整数按非递增的顺序，填入“螺旋矩阵”。所谓“螺旋矩阵”，是指从左上角第 1 个格子开始，按顺时针螺旋方向填充。要求矩阵的规模为 m 行 n 列，满足条件：m×n 等于 N；m≥n；且 m−n 取所有可能值中的最小值。

输入格式：

输入在第 1 行中给出一个正整数 N，第 2 行给出 N 个待填充的正整数。所有数字不超过 10^4，相邻数字以空格分隔。

输出格式：

输出螺旋矩阵。每行 n 个数字，共 m 行。相邻数字以 1 个空格分隔，行末不得有多余空格。

输入样例：

```
12
37 76 20 98 76 42 53 95 60 81 58 93
```

输出样例：

```
98 95 93
42 37 81
53 20 76
58 60 76
```

关键点

1. 条件：满足条件：m×n 等于 N；m≥n；且 m−n 取所有可能值中的最小值，m*n怎么求？

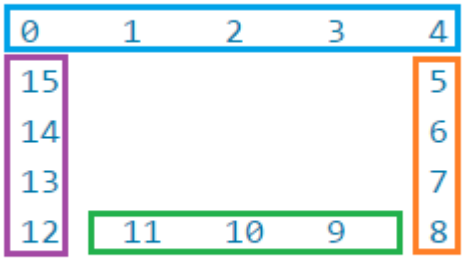
> 首先，对N（总数）开方，所得应该是m（行数）和n（列数）最接近时的n值，并且应该是n的最大值，但这个n不一定满足`N%n==0`，也即不能得到**整数**的行，所以应该再用循环判断n~1中满足`N%n==0`条件的情况，那么就可以得到m和n最接近情况下的正**整数**m和n了。

2. 如何声明螺旋矩阵？

- 声明vector；`vector<int> a(N);//存储输入数组`
- 声明二维vector；`vector<vector<int>> b(m,vector<int>(n));`这个声明很有用，声明了存储vector的vector，并在圆括号指明了为m大小，且用n大小的vector填充每一个元素。

3. 如何构造螺旋矩阵？

主要为四部分，例如：



1. 水平的-第一行：0~4（0~n列），应放在二维数组第一行；

```
int k = 0, i = 0;
int j = i;  //1. 水平的-第一行
while (j < n - i){
    b[i][j++] = a[k++];
}
```

2. 垂直的-最后一列：5~8（i+1~m-i-1行），应放在二维数组最后一列；

```
j = i + 1; //2.垂直的-最后一列
while (j < m - i){
    b[j++][n - i - 1] = a[k++];
}
```

3. 水平的-最后一行：9~11（n-i-2~i+1列），应放在二维数组最后一行；

```
j = n - i - 2;  //3.水平的-最后一行
while (j > i){
    b[m - i - 1][j--] = a[k++];
}
```

4. 竖直的-第一列：12~15（m-i-1~i+1行），应放在二维数组第一列。

```
j = m - i - 1;  //4.竖直的-第一列
while (j > i){
    b[j--][i] = a[k++];
}
```

四步完成了一个外层圈，加上相应的条件，即可构造整个二维数组，有两种方法：

法一：通过一维数组索引判断即可`k<N`，不需要多余的条件就可以完成。

> 且需注意内层for或while循环中还要控制k < N，因为如果螺旋矩阵中所有的元素已经都填充完毕，就不能再重复填充，例如N=5时，为5*1的矩阵，如果不加这个判断，则会在 `1.` `2.`中完成了二维数组，但是在`4.`时，`j = 5-0-1=4`，且满足`(j>i)--(4>0)`，所以又会进入赋值分支，造成数组越界问题，所以控制k < N是很必要的。

局部完整（使用while代码要简化些）：

```
//b是螺旋矩阵，a是输入数组
int k = 0, i = 0;
while (k < N){
    int j = i;  //1. 水平的-第一行
    while (j < n - i && k < N){
        b[i][j++] = a[k++];
    }
    j = i + 1;  //2.垂直的-最后一列
    while (j < m - i && k < N){
        b[j++][n - i - 1] = a[k++];
    }
    j = n - i - 2;  //3.水平的-最后一行
    while (j > i && k < N){
        b[m - i - 1][j--] = a[k++];
    }
    j = m - i - 1;  //4.竖直的-第一列
    while (j > i && k < N){
        b[j--][i] = a[k++];
    }
    i++;
}
```

法二：通过层数循环，通过m计算螺旋矩阵的层数level，如果m的值为偶数，层数为m/2，如果m为奇数，层数为m/2+1，所以level = m / 2 + m % 2; 。且同法一一样需要注意内层循环中还要控制k < N。

局部完整（这里使用for和上述比对）：

```
//b是螺旋矩阵，a是输入数组
int level = m / 2 + m % 2;
int k = 0;
for (int i = 0; i < level;i++){
    for (int j = i; j < n - i&&k < N; j++){
        b[i][j] = a[k++];//1. 水平的-第一行
    }
    for (int j = i + 1; j < m - i&&k < N; j++){
```

```
            b[j][n - i - 1] = a[k++];//2.垂直的-最后一列
        }
        for (int j = n - i - 2; j > i&&k < N; j--){
            b[m-i-1][j] = a[k++];// 3.水平的 - 最后一行
        }
        //4.垂直的-第一行
        for (int j = m - i - 1; j > i&&k < N; j--){
            b[j][i] = a[k++];
        }
    }
```

1068 万绿丛中一点红 （20 分）

对于计算机而言，颜色不过是像素点对应的一个 24 位的数值。现给定一幅分辨率为 M×N 的画，要求你找出万绿丛中的一点红，即有独一无二颜色的那个像素点，并且该点的颜色与其周围 8 个相邻像素的颜色差充分大。

输入格式：

输入第一行给出三个正整数，分别是 M 和N（≤ 1000），即图像的分辨率；以及 TOL，是所求像素点与相邻点的颜色差阈值，色差超过 TOL 的点才被考虑。随后 N 行，每行给出 M 个像素的颜色值，范围在 $[0, 2^{24})$ 内。所有同行数字间用空格或 TAB 分开。

输出格式：

在一行中按照 `(x, y): color` 的格式输出所求像素点的位置以及颜色值，其中位置 x 和 y 分别是该像素在图像矩阵中的列、行编号（从 1 开始编号）。如果这样的点不唯一，则输出 `Not Unique`；如果这样的点不存在，则输出 `Not Exist`。

输入样例 1：

```
8 6 200
0        0        0        0        0        0        0        0
65280    65280    65280    16711479 65280    65280    65280    65280
16711479 65280    65280    65280    16711680 65280    65280    65280
65280    65280    65280    65280    65280    65280    165280   165280
65280    65280    16777015 65280    65280    165280   65480    165280
16777215 16777215 16777215 16777215 16777215 16777215 16777215 16777215
```

输出样例 1：

```
(5, 3): 16711680
```

输入样例 2：

```
4 5 2
0 0 0 0
0 0 3 0
0 0 0 0
```

```
0 5 0 0
0 0 0 0
```

输出样例 2：

```
Not Unique
```

输入样例 3：

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

输出样例 3：

```
Not Exist
```

关键点

- 这个题的难点在于查找周围八个方向，一开始我的思路是，先判断边界，再去比较，但是这样的逻辑太复杂，需要考虑各种边界情况，以至于陷入逻辑深水中。
- 更好的思路，应更换思路，**对于八个方向先不管边界，得出边界索引后，再去判断是否溢出边界，这样的好处，不用对于每一种边界都去判断，仅根据得到索引去判断。**同时，利用图处理类似方法所采用的二维数组策略，可以更方便。

局部代码

```
/*
八个方向的行列索引差值，使用二维数组更形象方便。
{ -1, -1 }    { -1, 0 }    { -1, 1 }
{ 0, -1 }     { i, j }     { 0, 1 }
{ 1, -1 }     { 1, 0 }     { 1, 1 }
*/
int dir[8][2] = { { -1, -1 }, { -1, 0 }, { -1, 1 }, { 0, -1 }, { 0, 1 },
 { 1, -1 }, { 1, 0 }, { 1, 1 } };
bool judge(int x,int y){
    for (int i = 0; i < 8;i++){
        int tx = x + dir[i][0];//得到每个方向的x值
        int ty = y + dir[i][1];//得到每个方向的y值
        //越界判断，先计算索引，再判断会更方便
        if (tx >= 0 && tx < N && ty >= 0 && ty < M){
            if (abs(v[x][y] - v[tx][ty]) <= TOL)
                return false;
        }
    }
```

```
    return true;
}
```

1088 三人行 （20 分）

子曰：“三人行，必有我师焉。择其善者而从之，其不善者而改之。”
本题给定甲、乙、丙三个人的能力值关系为：甲的能力值确定是 2 位**正整数**；把甲的能力值的 2 个数字调换位置就是乙的能力值；甲乙两人能力差是丙的能力值的 X 倍；乙的能力值是丙的 Y 倍。请你指出谁比你强应“从之”，谁比你弱应“改之”。
输入格式：
输入在一行中给出三个数，依次为：M（你自己的能力值）、X 和 Y。三个数字均为不超过 1000 的正整数。
输出格式：
在一行中首先输出甲的能力值，随后依次输出甲、乙、丙三人与你的关系：如果其比你强，输出`Cong`；平等则输出`Ping`；比你弱则输出`Gai`。其间以1个空格分隔，行首尾不得有多余空格。
注意：如果解不唯一，则以甲的最大解为准进行判断；如果解不存在，则输出`No Solution`。
输入样例 1：

```
48 3 7
```

输出样例 1：

```
48 Ping Cong Gai
```

输入样例 2：

```
48 11 6
```

输出样例 2：

```
No Solution
```

对题目有异议

- 甲的能力值确定是 2 位**正整数**，0不是正整数，那么甲应该是两位1~9的数字组成的，但是，在提交时，却有一个点不通过，当加上个位数可以为0后，通过了，则说明这个条件是有问题的。

注意点

- 甲乙可以看出一定是整数，但是乙却不一定，所以对于乙的判断，应该用`double`去判断。

1. 除法的类型转换是以分子类型为准

- `int/double`会先将`int`转化为`double`，在进行除法计算。
- 同理，`double/int`是将`double`转化为`int`型，在进行除法计算，这个一定要注意。

类型二：查找元素

题号	题目	说明
B1028	人口普查 （20 分）	YY-MM-HH形式比较

B1028人口普查 （20 分）

某城镇进行人口普查，得到了全体居民的生日。现请你写个程序，找出镇上最年长和最年轻的人。
这里确保每个输入的日期都是合法的，但不一定是合理的——假设已知镇上没有超过 200 岁的老人，而今天是 2014 年 9 月 6 日，所以超过 200 岁的生日和未出生的生日都是不合理的，应该被过滤掉。
输入格式：
输入在第一行给出正整数 N，取值在$(0, 10^5]$；随后 N 行，每行给出 1 个人的姓名（由不超过 5 个英文字母组成的字符串）、以及按 `yyyy/mm/dd`（即年/月/日）格式给出的生日。题目保证最年长和最年轻的人没有并列。
输出格式：
在一行中顺序输出有效生日的个数、最年长人和最年轻人的姓名，其间以空格分隔。
输入样例：

```
5
John 2001/05/12
Tom 1814/09/06
Ann 2121/01/30
James 1814/09/05
Steve 1967/11/20
```

输出样例：

```
3 Tom John
```

思路

方法一. 年月日比较法，按顺序依次比较年月日，封装成函数。注意：结果为0时，只需要输出0，而不需要其他任何，**包括空格**。

```
struct person{
    char name[6];
    int year;
    int month;
    int day;
}tempP,maxP,minP,max,min;
//年份，月份，天均小于等于 --- 年龄等于大于
bool lessEqu(person p1,person p2){
    if (p1.year != p2.year)
```

```
        return p1.year <= p2.year;
    if (p1.month != p2.month)
        return p1.month <= p2.month;
    else
        return p1.day <= p2.day;
}
//年份，月份，天均大于等于 --- 年龄等于小于
bool moreEqu(person p1, person p2){
    if (p1.year != p2.year)
        return p1.year >= p2.year;
    if (p1.month != p2.month)
        return p1.month >= p2.month;
    else
        return p1.day >= p2.day;
}
```

方法二. 利用字符串的顺序比较，借助STL的string直接的结果，不需要封装函数。

完整代码

```
#include <cstdio>
#include <string>
#include <iostream>
using namespace std;

int main(){
    int n;
    scanf("%d",&n);
    int count = 0;
    string s1, s2, left = "1814/09/06", right = "2014/09/06";
    string max = right, min = left, maxName, minName;
    for (int i = 0; i < n; i++){
        cin >> s1 >> s2;
        if (s2 >= left && s2 <= right){
            count++;
            if (s2 < max){
                max = s2;
                maxName = s1;
            }
            if (s2 >min){
                min = s2;
                minName = s1;
            }
        }
    }
    cout << count;
    if (count != 0)
        cout << " " << maxName << " " << minName << endl;
```

```
    return 0;
}
```

类型三：图形输出

题号	题目	说明
B1027	打印沙漏（20 分）	逻辑，等差数列

1027 打印沙漏 （20 分）

本题要求你写个程序把给定的符号打印成沙漏的形状。例如给定17个“*”，要求按下列格式打印

```
*****
 ***
  *
 ***
*****
```

所谓“沙漏形状”，是指每行输出奇数个符号；各行符号中心对齐；相邻两行符号数差2；符号数先从大到小顺序递减到1，再从小到大顺序递增；首尾符号数相等。
给定任意N个符号，不一定能正好组成一个沙漏。要求打印出的沙漏能用掉尽可能多的符号。
输入格式：
输入在一行给出1个正整数N（≤1000）和一个符号，中间以空格分隔。
输出格式：
首先打印出由给定符号组成的最大的沙漏形状，最后在一行中输出剩下没用掉的符号数。
输入样例：

```
19 *
```

输出样例：

```
*****
 ***
  *
 ***
*****
2
```

关键点

- 首先这道题是一道数学题，需要知道这个是一个等差数列。

$$a_n = (2 * n + 1)$$

$$S_n = \frac{1}{2} n(a_1 + a_2)$$

本题中，先不考虑第一行，从第2行开始，所以为3+5+7+…(2*x+1)，其中我们设x总行数，所以：

$$S_x = \frac{x(3+2*x+1)}{2} = x*(x+2)$$

可以通过这个公式得到一个三角打印的最大行数x。

局部代码如下(设N为总个数)：

```
int x,m,n;//x:一个三角的层数，不算第一行；m*n数组
for (x = 1; x <= N;x++){
    if (2 * x*(x + 2) + 1 > N){
        x--;
        break;
    }
}
```

得到一个倒三角的最大行数后，剩下就是输出问题。

- 如何输出？

  按照一般思想，二维数组双循环+索引关系也是可以解决的，但是逻辑上是有重复的，而且容易乱。

  考虑，另一种关系，每一个循环变量不再是索引，而是**数量关系**，那么这个问题逻辑上就会简化。

局部代码如下

```
for (int i = x; i >= 0;i--){                    //打印倒三角，包含倒三角最后一行
    for (int j = 0; j < x - i;j++){               //每一行有行数-i个空格
        printf(" ");                            //最后一行 行数个 空格
    }
    for (int j = 0; j < 2 * i + 1;j++){         //每一行有2 * i + 1个符号（等差数列公式）
        printf("%c",C);
    }
    printf("\n");

}
for (int i = 1; i <= x;i++){                    //打印正三角，不含第一行
    for (int j = 0; j < x-i;j++){               //每一行有x-i个空格
        printf(" ");
    }
    for (int j = 0; j < 2 * i + 1;j++){         //每一行有 2 * i + 1个符号
        printf("%c", C);
    }
    printf("\n");
}
```

类型四：进制转换

两种思路：

1. 小时分秒，年月日等转换问题，化为最小，再进制转换。
2. 按照逻辑顺序依次比较，直到得到结果。

典型题目（不难）

题号	题目	说明
B1022	D进制的A+B （20 分）	do{}while()处理
B1037	在霍格沃茨找零钱 （20 分）	边界检测

类型五：字符串处理

- 对于简单字符操作，可以直接使用`scanf("%c",&C)`逐字读取。
- 配合`while (scanf("%c",&C)!= EOF)`循环读取字符串，EOF是文件结束标志，在windows平台，一般模拟eof的输入是在一个新行的开头输入`ctrl + z`就行了；在unix环境下，是在一个新行的开始出输入`ctrl + D`就可以了。同时要注意，**字符读取会读到`\n`换行标记符，所以计算长度时应减1。**

scanf(“%s”,ch)和cin >> str比较

相同点：

- 都不读空格，都是读取字符串，遇到空格，回车或者制表符就会结束输入；
- 其后接`fgets(ch,1010,stdin);`或`getline(cin,s);`时，都需要加`getchar();`读取换行符以进行下一行读取；
- 读取字符相同。

不同点

- `scanf`读取到`char`数组，`cin`读取到`string`对象；
- 需要的库不同：
  - `scanf`不需要，可能需要对`char`数组的操作库`<cstring>`
  - `cin >> str;`需要`<iostream><string>`

fgets(char数组,大小,stdin)和getline(cin,str)

相同点：

- 均按行读取，可以按行读取**任何**字符，所以可以读空格。

不同点

- `fgets`读取到`char`数组的长度比**你预期输入的字符串长度大一**，因为包含了结尾字符`\n`，这一点和`cin.get()`一样，都将换行符保留在输入序列中；而`getline`读取的字符串长度正常。

include <ctype.h>库

包含了判断字母数字，大小写转换等函数。

`isalpha`：检查ch是否是字母。
`iscntrl`：检查ch是否控制字符(其ASCII码在0和0x1F之间,数值为 0-31)。
`isdigit`：检查ch是否是数字(0-9)。
`islower`：检查ch是否小写字母(a-z)。
`isupper`：检查ch是否是大写字母(A-Z)。
`tolower`：将ch字符转换为小写字母。
`toupper`：将ch字符转换成大写字母。
`isalnum`：检查ch是否是字母或数字。

sscanf()和sprintf()的使用

`sscanf()`和`scanf()`类比
`sscanf(ch,%d,&n)`，类比于`scanf()`是读取到n，`sscanf()`也是读取到n，只不过是从字符串数组`ch[]`读入的。有一些注意：

- **返回值：**如果成功，该函数返回成功匹配和赋值的个数。如果到达文件末尾或发生读错误，则返回`EOF`。

`sprintf()`和`printf()`类比
`sprintf(ch,%d,n)`，类比于`printf()`是输出到命令行，`sprintf()`也是输出，只不过是输出到字符串数组`ch[]`。有一些注意：

- **返回值：**如果成功，则返回写入的字符总数，不包括字符串追加在字符串末尾的空字符。如果失败，则返回一个负数。

使用建议

三种读取方式，一般字符串较复杂的情况，`getline`和`fgets`和`cin`方式比较好，相对难用的是`while(scanf("%c",&c)!=EOF)`方法。

> `fgets`面向char数组，`cin`面向`string`对象，这两个均有相关函数`<cstring><string>`配合操作字符串，而`while`面向的确实`char`类型，其对于单个字符处理可能还不错，但如果涉及一些字符串操作，则会显得低效，但仍有一些方法不得不用`while`。

较难题目

题号	题目	说明
B1024	科学计数法 （20 分）	逻辑较为复杂，要谨慎
B1052	卖个萌 （20 分）	数组越界访问
B1054	求平均值 （20 分）	sscanf()和sprintf()

1024 科学计数法 （20 分）

科学计数法是科学家用来表示很大或很小的数字的一种方便的方法，其满足正则表达式 [+-][1-9].[0-9]+E[+-][0-9]+，即数字的整数部分只有 1 位，小数部分至少有 1 位，该数字及其指数部分的正负号即使对正数也必定明确给出。
现以科学计数法的格式给出实数 A，请编写程序按普通数字表示法输出 A，并保证所有有效位都被保留。
输入格式：
每个输入包含 1 个测试用例，即一个以科学计数法表示的实数 A。该数字的存储长度不超过 9999 字节，且其指数的绝对值不超过 9999。
输出格式：
对每个测试用例，在一行中按普通数字表示法输出 A，并保证所有有效位都被保留，包括末尾的 0。
输入样例 1：

> +1.23400E-03

输出样例 1：

```
0.00123400
```

输入样例 2：

```
-1.2E+10
```

输出样例 2：

```
-12000000000
```

切入点

正则表达式`[+-][1-9].[0-9]+E[+-][0-9]+`给出本题关键。**整数部分为（1-9）**限制了0的情况。

- 不论是什么样的字符串读取逻辑，其目的都是需要固定的几个数组或数字：
  - 实数符号，为’+’号不输出，’-‘号应输出。
  - 实数部分，存入char数组，应不含小数点，因为固定为第一位后为小数点。
  - 指数正负号，存为char，用于后面移位的判断。
  - 指数部分，存为int，后面可直接作为判断边界。

分上述a-d四部分读取并存入相应类型中。

- 逻辑部分
  - 不考虑指数为`0`（包含+0和-0）情况，因为无意义。
  - 指数为正数时，进位后仍需要小数点，说明指数小于（实数位数-1）（小数点在第一位后，所以要减一），并将小数点放置实数char数组中指数+1位置。
  - 指数为正数时，进位后不需要小数点，说明指数大于等于（实数位数-1），此时，可能需要补零，根据指数大于（实数位数-1）判断。
  - 指数为负数时，一定输出0.xxx…类型，所以先输出`0.`，再判断是否继续输出0，根据指数-1是否大于0判断，然后输出实数char数组。

指数为正

```
if (operE == '+'){//指数为正数
    if (expAbs < lenA - 1){ //需要小数点
        for (int i = 0; i < lenA; i++){
            if (i == expAbs + 1){
                printf(".");
            }
            printf("%c", A[i]);
        }
    }
    else{//不需要小数点，但是要添0
        for (int i = 0; i < lenA; i++){
            printf("%c", A[i]);
```

```
            }
            for (int j = 0; j < expAbs - lenA + 1; j++){
                printf("0");
            }
        }
    }
```

指数为负

```
    else{//指数为负数，不考虑 -0 情况，因为没意义
        printf("0.");
        for (int j = 0; j < expAbs - 1; j++){
            printf("0");
        }
        for (int i = 0; i < lenA; i++){
            printf("%c", A[i]);
        }
    }
```

1052 卖个萌 （20 分）

萌

萌哒表情符号通常由“手”、“眼”、“口”三个主要部分组成。简单起见，我们假设一个表情符号是按下列格式输出的：

[左手]([左眼][口][右眼])[右手]

现给出可选用的符号集合，请你按用户的要求输出表情。

输入格式：

输入首先在前三行顺序对应给出手、眼、口的可选符号集。每个符号括在一对方括号 []内。题目保证每个集合都至少有一个符号，并不超过 10 个符号；每个符号包含 1 到 4 个非空字符。

之后一行给出一个正整数 K，为用户请求的个数。随后 K 行，每行给出一个用户的符号选择，顺序为左手、左眼、口、右眼、右手——这里只给出符号在相应集合中的序号（从 1 开始），数字间以空格分隔。

输出格式：

对每个用户请求，在一行中输出生成的表情。若用户选择的序号不存在，则输出 `Are you kidding me? @\/@`。

输入样例：

```
[╮ ][ ╭][o][˜][/˜] [<][>]
[ノ ][ ╰][ˆ][-][=][>][<][@][⊙]
[Д][▽][_][ ε ][ˆ] …
4
1 1 2 2 2
6 8 1 5 5
3 3 4 3 3
2 10 3 9 3
```

输出样例：

```
ㄟ(ノ▽ヽ)ㄏ
<(@Д=)/~
o(^ε^)o
Are you kidding me? @\/@
```

注意点

我想正常的按行读取字符串应该不会在逻辑上有问题，仅注意考虑读取空格。

- 读入处理，需要根据括号`[ ]`来获取字符，这个逻辑也不难。

```
int getChars(string s,string strs[11]){
    int cnt = 0;
    int len = s.length();
    int i = 0;
    while (i < len){
        if (s[i] == '['){                                   //找到字符 '['
            i++;
            while (s[i] != ']'){
                strs[cnt] += s[i++];                        //将 [ ]中字符存入h
and[cnt]中，其他同理
            }
            cnt++;
        }
        i++;
    }
    return cnt;                                             //返回表情符号个
数
}
```

- 另一个考察点：**数组越界判断**，需要注意给出的数字既不能大于最长长度，也不能小于1。我没有注意下界导致很长时间过不去，测试数据里有两个测试用例是含有小于1的数字。
- 最后一个注意的，特殊符号如果想输出在c语言中是双写的。例如：`printf("%%\\"); //输出 : %\`

1054 求平均值 （20 分）

本题的基本要求非常简单：给定N个实数，计算它们的平均值。但复杂的是有些输入数据可能是非法的。一个“合法”的输入是 [-1000,1000] 区间内的实数，并且最多精确到小数点后 2 位。当你计算平均值的时候，不能把那些非法的数据算在内。

输入格式：

输入第一行给出正整数 $N(\leq 100)$。随后一行给出 N 个实数，数字间以一个空格分隔。

输出格式：

对每个非法输入，在一行中输出 `ERROR: X is not a legal number`，其中 X 是输入。最后在一行中输出

结果：`The average of K numbers is Y`，其中 K 是合法输入的个数，Y 是它们的平均值，精确到小数点后 2 位。如果平均值无法计算，则用 `Undefined` 替换 Y。如果 K 为 1，则输出 `The average of 1 number is Y`。

输入样例 1：

```
7
5 -3.2 aaa 9999 2.3.4 7.123 2.35
```

输出样例 1：

```
ERROR: aaa is not a legal number
ERROR: 9999 is not a legal number
ERROR: 2.3.4 is not a legal number
ERROR: 7.123 is not a legal number
The average of 3 numbers is 1.38
```

输入样例 2：

```
2
aaa -9999
```

输出样例 2：

```
ERROR: aaa is not a legal number
ERROR: -9999 is not a legal number
The average of 0 numbers is Undefined
```

关键点

本题主要考查，一个知识点：`sscanf()`和`sprintf()`的使用。

`sprintf(ch,%d,n)`，类比于`printf()`是输出到命令行，`sprintf()`也是输出，只不过是输出到字符串数组`ch[]`。有一些注意：

- **返回值**：如果成功，则返回写入的字符总数，不包括字符串追加在字符串末尾的空字符。如果失败，则返回一个负数。

这样可以利用这个性质，分别将原始读入的字符串和经过格式转化过的字符串比较，如果不等，则得到数据不合格。

部分代码

```
scanf("%s", a);
int all = sscanf(a, "%lf", &temp);
if (all == 0){ //用于不是数字的判断
    printf("ERROR: %s is not a legal number\n", a);
    continue;
```

```
    }
    sprintf(b,"%.2f",temp);
    bool flag = true;//数据合法
    int len = strlen(a);
    for (int j = 0; j < len; j++){
        if (a[j] != b[j]){
            flag = false;
            break;
        }
    }
```

注意：这里对sscanf加了一个返回判断，因为对于不是数字的读取，实际操作时返回了一个错误，所以在这里直接规避掉。

类型六：排序

- 考察`<algorithm>`库中`sort()`的使用。
- `unordered_map`效率优于`map`

题号	题目	说明
B1055	集体照 （25 分）	左右交替==循环加2
B1095	解码PAT准考证 （25 分）	有时候后处理数据会简化逻辑
B1035	插入与归并 （25 分））	插入排序和归并排序

1035 插入与归并 （25 分）

根据维基百科的定义：
插入排序是迭代算法，逐一获得输入数据，逐步产生有序的输出序列。每步迭代中，算法从输入序列中取出一元素，将之插入有序序列中正确的位置。如此迭代直到全部元素有序。
归并排序进行如下迭代操作：首先将原始序列看成 N 个只包含 1 个元素的有序子序列，然后每次迭代归并两个相邻的有序子序列，直到最后只剩下 1 个有序的序列。
现给定原始序列和由某排序算法产生的中间序列，请你判断该算法究竟是哪种排序算法？
输入格式：
输入在第一行给出正整数 N（≤100）；随后一行给出原始序列的 N 个整数；最后一行给出由某排序算法产生的中间序列。这里假设排序的目标序列是升序。数字间以空格分隔。
输出格式：
首先在第 1 行中输出`Insertion Sort`表示插入排序、或`Merge Sort`表示归并排序；然后在第 2 行中输出用该排序算法再迭代一轮的结果序列。题目保证每组测试的结果是唯一的。数字间以空格分隔，且行首尾不得有多余空格。
输入样例 1：

```
10
3 1 2 8 7 5 9 4 6 0
1 2 3 7 8 5 9 4 6 0
```

输出样例 1：

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

输入样例 2：

```
10
3 1 2 8 7 5 9 4 0 6
1 3 2 8 5 7 4 9 0 6
```

输出样例 2：

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

关键点

- 因为只有两种情况，那么根据从简单的情况入手。插入排序的判断比较容易，插入排序具有：**前面有序，而后面和排序前一样的特点。**

部分实现

```
int k = 1;
while (k < N-1 && b[k - 1] <= b[k]) k++;
int len = k;
while (k < N){
    if (a[k] != b[k]) break;
    k++;
}
if (k == N){ //根据是否按照到达结尾判断插入排序
    printf("Insertion Sort\n");
}
```

- 归并排序的下一轮排序，需要直到上一轮排序的步长。如何得到步长，提供一个资料：https://www.bilibili.com/video/av10948002/?p=141，利用了本题的一个特点，归并排序下最小步长为2，如果步长为1，那么也即是原数列，会出现一种：

  ```
  10
  3 1 2 8 7 5 9 4 0 6
  3 1 2 8 7 5 9 4 0 6
  ```

  那么就没办法判断是归并还是插入。所以，归并的步长从2开始找就可以了。

```
int getStep(){
    int l = 2;
    for (; l*2 < N; l *= 2){
        for (int j = l; j+2*l < N; j += 2 * l){
            if (b[j - 1] > b[j]) return l;
        }
    }
    return l;
}
```

完整代码

```
#include <cstdio>
```

```cpp
#include <algorithm>
using namespace std;
const int maxn = 110;
int N;
int a[maxn] = {};
int b[maxn] = {};
int getStep(){
    int l = 2;
    for (; l*2 < N; l *= 2)
        for (int j = l; j+2*l < N; j += 2 * l)//判断当前步长是否使得局部有序，如果有序，步长应该乘2继续判断
            if (b[j - 1] > b[j]) return l;
    return l;
}
int main()
{
    scanf("%d", &N);
    for (int i = 0; i < N; i++){
        scanf("%d",&a[i]);
    }
    for (int i = 0; i < N; i++){
        scanf("%d", &b[i]);
    }
    /*参照插入排序的判别*/
    int k = 1;
    while (k < N-1 && b[k - 1] <= b[k]) k++;//前k项有序
    int len = k;
    while (k < N && a[k] == b[k]) k++;//后几项相同
    if (k == N){
        printf("Insertion Sort\n");
        sort(b,b+len+1);//下一轮排序
        for (int i = 0; i < N; i++){
            if (i != 0) printf(" ");
            printf("%d", b[i]);
        }
    }
    else{
        printf("Merge Sort\n");
        /*得到下一次归并的步长*/
        int step = getStep() * 2;
        for (int i = 0; i < N; i += step){
            sort(b + i, b + min(i + step, N));//这里用个min来得到是否边界
        }
        for (int i = 0; i < N; i++){
            if (i != 0) printf(" ");
            printf("%d", b[i]);
        }
    }
    return 0;
```

```
}
```

1055 集体照 （25 分）

拍集体照时队形很重要，这里对给定的 N 个人 K 排的队形设计排队规则如下：

- 每排人数为 N/K（向下取整），多出来的人全部站在最后一排；
- 后排所有人的个子都不比前排任何人矮；
- 每排中最高者站中间（中间位置为 m/2+1，其中 m 为该排人数，除法向下取整）；
- 每排其他人以中间人为轴，按身高非增序，先右后左交替入队站在中间人的两侧（例如5人身高为190、188、186、175、170，则队形为175、188、190、186、170。这里假设你面对拍照者，所以你的左边是中间人的右边）；
- 若多人身高相同，则按名字的字典序升序排列。这里保证无重名。

现给定一组拍照人，请编写程序输出他们的队形。

输入格式：
每个输入包含 1 个测试用例。每个测试用例第 1 行给出两个正整数 $N(\le 10^4$，总人数）和 K（≤10，总排数）。随后 N 行，每行给出一个人的名字（不包含空格、长度不超过 8 个英文字母）和身高（[30, 300]区间内的整数）。
输出格式：
输出拍照的队形。即K排人名，其间以空格分隔，行末不得有多余空格。注意：假设你面对拍照者，后排的人输出在上方，前排输出在下方。
输入样例：

```
10 3
Tom 188
Mike 170
Eva 168
Tim 160
Joe 190
Ann 168
Bob 175
Nick 186
Amy 160
John 159
```

输出样例：

```
Bob Tom Joe Nick
Ann Mike Eva
Tim Amy John
```

疑难点

- 排数以及每行人数不是难点，难点在于每一行的人怎么站，如果被题目逻辑所引导，那么问题很难解决，左右交替站位直接实现很麻烦，但是如果跳出它的逻辑，发现，按大到小排序，每行从中间开始，向左依次是循环加2，向右也是循环加2，就挑出了它的左右交替排序了。

1095 解码PAT准考证 （25 分）

PAT 准考证号由 4 部分组成：

- 第 1 位是级别，即 T 代表顶级；A 代表甲级；B 代表乙级；
- 第 2~4 位是考场编号，范围从 101 到 999；
- 第 5~10 位是考试日期，格式为年、月、日顺次各占 2 位；
- 最后 11~13 位是考生编号，范围从 000 到 999。

现给定一系列考生的准考证号和他们的成绩，请你按照要求输出各种统计信息。

输入格式：

输入首先在一行中给出两个正整数 N（$\le 10^4$）和 M（≤100），分别为考生人数和统计要求的个数。

接下来 N 行，每行给出一个考生的准考证号和其分数（在区间 [0,100] 内的整数），其间以空格分隔。

考生信息之后，再给出 M 行，每行给出一个统计要求，格式为：`类型 指令`，其中

- `类型` 为 1 表示要求按分数非升序输出某个指定级别的考生的成绩，对应的 `指令` 则给出代表指定级别的字母；
- `类型` 为 2 表示要求将某指定考场的考生人数和总分统计输出，对应的 `指令` 则给出指定考场的编号；
- `类型` 为 3 表示要求将某指定日期的考生人数分考场统计输出，对应的 `指令` 则给出指定日期，格式与准考证上日期相同。

输出格式：

对每项统计要求，首先在一行中输出 `Case #: 要求`，其中 `#` 是该项要求的编号，从 1 开始；`要求` 即复制输入给出的要求。随后输出相应的统计结果：

- `类型` 为 1 的指令，输出格式与输入的考生信息格式相同，即 `准考证号 成绩`。对于分数并列的考生，按其准考证号的字典序递增输出（题目保证无重复准考证号）；
- `类型` 为 2 的指令，按 `人数 总分` 的格式输出；
- `类型` 为 3 的指令，输出按人数非递增顺序，格式为 `考场编号 总人数`。若人数并列则按考场编号递增顺序输出。

  如果查询结果为空，则输出 `NA`。

输入样例：

```
8 4
B123180908127 99
B102180908003 86
A112180318002 98
T107150310127 62
A107180908108 100
T123180908010 78
B112160918035 88
A107180908021 98
1 A
```

```
2 107
3 180908
2 999
```

输出样例：

```
Case 1: 1 A
A107180908108 100
A107180908021 98
A112180318002 98
Case 2: 2 107
3 260
Case 3: 3 180908
107 2
123 2
102 1
Case 4: 2 999
NA
```

关键点

本题应该着重于数据处理，但是到底是边读边处理还是，全部读完后处理。对于类型一和二，其实无所谓，但是对于类型三，实际操作下发现，全部读完后处理数据要方便。如果，边读边处理，那么需要给出所有关于日期的分类，这需要很多数据结构去存储，但是如果读完后处理，那么只需要去找需要的日期，并组成相应的数据结构就可以了。

-综上，本题不应该为了效率去在读数据时处理数据，这样增加了逻辑上的复杂度，我第一次想法就是这样，我觉得这样做，如果第一次做这个题，给定时间应该做不出来的。

- 如何处理日期关系，由于类型三所需要的数据结构和学生学号和成绩结构相同，可以借助学生的结构，只是对于每个数据解读不同，例如：学生的id在此时应被解读为考场号，学生的成绩被解读为考场的人数。而且比较的逻辑也一样，这样减少了一个结构和一个比较，减少了重复的逻辑关系。
- 使用哪几种数据结构，`map`在此时不是好的，`unordered_map`为一种更高效的`map`，应该用它来解决超时问题。

本题用了一些不常见的赋值方法，应该学习使用。

- 循环获取vector中的元素：

```
for (node n : v)
```

- 对vector存储

```
vector<node> ans;
ans.push_back({s,n});//可以这么使用，是因为vector的类型是定义过的，已知的
```

- 对map的读取并转换为vector

```
//方法一
for (unordered_map<string, int>::iterator it = m.begin(); it != m.end();
 it++){
    ans.push_back({it->first,it->second});
}
//方法二
for (auto it : m) ans.push_back({ it.first, it.second });
```

类型七：散列

主要考察hash。

题号	题目	说明
B1005	继续(3n+1)猜想 （25 分）	3*n+1的范围需要考虑
1059	C语言竞赛 （20 分）	逻辑关系
B1065	单身狗 （25 分）	注意输出%05d

类型八：二分查找

二分查找的基本思路不用再说了，注意几个点：

- 虽然查找条件`while(low<=high)`也可以写成`while(low<high)`，但是有区别，前者未找到时，low和high处于第一次`low>high`的状态；而后者处于`low==high`的状态。这里统一下，用第一种方法，后面会说为什么这么做。
- 总是在low~mid-1和mid+1~high之间查找元素。对于mid判断完毕后，不用再包含mid。

二分查找标准代码（查找不到返回-1）

```
//A为递增数列，x为欲查询的数，函数返回查找到的索引，未查找到返回-1
int binarySearch(int a[],int low,int high,int x){
    while (low <= high){
        int mid = low + (high - low) / 2;
        if (a[mid] < x){
            low = mid + 1;
        }
        else if (a[mid] > x){
            high = mid - 1;
        }
        else
            return mid;
    }
    return -1;
}
```

二分查找扩展

基于二分查找，可以进一步扩展两个方法。

- 查找第一个大于或等于x的元素位置
- 查找第一个大于x的元素位置

查找第一个大于或等于x的元素位置

原理比较简单，只需要对分支判断中的等于做相应处理即可。

```
//A为递增数列，x为欲查询的数，函数返回查找到的索引，未查找到返回-1
int binarySearch(int a[],int low,int high,int x){
    while (low <= high){
        int mid = low + (high - low) / 2;
        if (a[mid] < x){
            low = mid + 1;
        }
        else if (a[mid] >= x){
```

```
                high = mid - 1;
            }
        }
        return low;
    }
```

这里修改了三处：第一，修改了`return -1`为`return low`；第二，修改条件`if (a[mid] > x)`为`if (a[mid] >= x)`；第三，删除条件`return mid`。

分析：
查找第一个大于或等于x的元素位置，将条件`if (a[mid] > x)`改为`if (a[mid] >= x)`，对于只要大于等于x的位置，都在其左半部分查找（降低high）。该条件会导致高位high不断向左靠近，直到最后一个小于x的位置。
最终，high和low均指向最后一个小于x的位置。这里要解释下上面为什么while条件中使用`(low<=high)`，当`while (low == high)`成立，条件满足`if (a[mid] < mp) low = mid + 1;`，所以最终能通过low返回第一个大于等于x的索引位置。其目的就是为了保证low在等于high（指向最后一个小于x的位置）时，仍可以多一步运算而指向第一个大于等于的元素。

查找第一个大于x的元素位置

同上。只不过等于号加在另一个条件中。

```
//A为递增数列，x为欲查询的数，函数返回查找到的索引，未查找到返回-1
int binarySearch(int a[],int low,int high,int x){
    while (low <= high){
        int mid = low + (high - low) / 2;
        if (a[mid] <= x){
            low = mid + 1;
        }
        else if (a[mid] > x){
            high = mid - 1;
        }
    }
    return low;
}
```

与上面唯一的不同在于将等号放在了条件`if (a[mid] <= x)`中，但是却将最终结果变成了查找第一个大于x的元素位置。

分析：
此时，对于小于等于x的情况，都是在右半部分查找（提高low），该条件会导致低位low不断向右靠近，直到最后一个小于或等于x的位置。
当（low==high）时，将`low = mid+1`，最终将返回第一个大于x的位置索引。

需注意的题目

题号	题目	说明
B1030	完美数列 （25 分）	查找第一个大于x的元素位置

1030 完美数列 （25 分）

给定一个正整数数列，和正整数 p，设这个数列中的最大值是 M，最小值是 m，如果 M≤mp，则称这个数列是完美数列。

现在给定参数 p 和一些正整数，请你从中选择尽可能多的数构成一个完美数列。

输入格式：
输入第一行给出两个正整数 N 和 p，其中 N（$\le 10^5$）是输入的正整数的个数，p（$\le 10^9$）是给定的参数。第二行给出 N 个正整数，每个数不超过 10^9。
输出格式：
在一行中输出最多可以选择多少个数可以用它们组成一个完美数列。
输入样例：

```
10 8
2 3 20 4 5 1 6 7 8 9
```

输出样例：

```
8
```

注意点

- 是在这个数列中选择最长的完美数列，所以应对每一个数进行完美数列的判断。
- long long类型不自动转换

```
long long mp;
int a = 1 << 20;
mp = a*a;
printf("%lld",mp);//结果为：0
```

想要得到a*a的正确结果，要加类型转换：

```
long long mp;
int a = 1 << 20;
mp = a*a;//要在a前加(long long)
printf("%lld",mp);//结果为：1099511627776
```

同理，知道，c语言不像java那样可以根据声明类型自动转换类型，所以，以后类型转换一定要手动添加。

此题有两个解法：二分查找和双指针法

类型九：贪心问题

简单地讲，就一句话，**局部最优解将作为整体最优解就是贪心。**

题号	题目	说明
B1070	结绳 （25 分）	弄清逻辑再动手

1070 结绳 （25 分）

给定一段一段的绳子，你需要把它们串成一条绳。每次串连的时候，是把两段绳子对折，再如下图所示套接在一起。这样得到的绳子又被当成是另一段绳子，可以再次对折去跟另一段绳子串连。每次串连后，原来两段绳子的长度就会减半。

给定 N 段绳子的长度，你需要找出它们能串成的绳子的最大长度。

输入格式：

每个输入包含 1 个测试用例。每个测试用例第 1 行给出正整数 $N(2 \le N \le 10^4)$；第 2 行给出 N 个正整数，即原始绳段的长度，数字间以空格分隔。所有整数都不超过10^4 。

输出格式：

在一行中输出能够串成的绳子的最大长度。结果向下取整，即取为不超过最大长度的最近整数。

输入样例：

```
8
10 15 12 3 4 13 1 15
```

输出样例：

```
14
```

注意点

- 搞清楚逻辑，14是怎么得到的，**为了得到最长的绳子，较长的绳子则应该被较少次数的对折。**需要将所有的绳子从短到长排序，从短的开始，每两个对折连接。输入样例排序：

```
1 3 4 10 12 13 15 15
```

$$\frac{1}{2}+\frac{3}{2}=2$$

$$\frac{2}{2}+\frac{4}{2}=3$$

$$\frac{3}{2} + \frac{10}{2} = 6.5$$

$$\frac{6.5}{2} + \frac{12}{2} = 9.25$$

$$\frac{9.25}{2} + \frac{13}{2} = 11.125$$

$$\frac{11.125}{2} + \frac{15}{2} = 13.0625$$

$$\frac{13.0625}{2} + \frac{15}{2} = 14.03125$$

最后取整，得到14。

- 只有两个绳子的情况，循环首先应将开始置为第一个数，从第二个数开始循环。否则，会导致逻辑问题。

类型十：双指针

一般针对有序序列，双指针有着很大的作用。举一道LeetCode的题目：1.TwoSum，就利用了排序+双指针来处理。

题号	题目	说明
B1030	完美数列（25分）	逻辑，二维vector

B1030双指针代码

```
#include <cstdio>
#include <algorithm>
using namespace std;

const int maxn = 100010;
int a[maxn] = {};
int main()
{
    /*双指针法*/
    int N, p;
    long long mp;
    scanf("%d%d",&N,&p);
    for (int i = 0; i < N;i++){
        scanf("%d", &a[i]);
    }
    sort(a, a + N);
    int max = 0,high = 0;//high赋值一次即可
    for (int i = 0; i < N;i++){
        mp = (long long)a[i] * p;
        while (high < N){
            if (a[high] <= mp) high++;//对于后面的数来说，a[high]要么是第一个大于mp的数，要么是小于等于mp的数。
            else break;
        }
        max = max > high - i? max : high - i;
    }
    printf("%d",max);
    return 0;
}
```

类型十一：数学问题

涉及一些数学规律的发现和分析，要仔细，不急躁的处理这种问题。

分数处理

两个分数间的大小比较，可以利用交换律，改为乘法比较，这样不会出现小数double的精度问题。虽然乘法有溢出问题，但是只要注意溢出，他是比实际除法小数要好。

$$\frac{n1}{m1} > \frac{n2}{m2} \Rightarrow n1m2 > m1n2$$

最小公约数

欧几里得算法
计算两个非负数a和b的最大公约数：若b是0，则最大公约数为a。否则，将a除以b得到的余数r，a和b的最大公约数即为b和r的最大公约数。

```
int gcd(int a,int b){
    return b == 0 ? a : gcd(b, a%b);
}
```

题号	题目	说明
B1003	我要通过！（20 分）	数学规律分析
B1019	数字黑洞 （20 分）	0补位
B1034	有理数四则运算 （20 分）	struct分数的四则运算
B1040	有几个PAT （25 分）	动态规划
B1045	快速排序 （25 分）	动态规划
B1049	数列的片段和 （20 分）	数学规律分析
B1051	复数乘法 （15 分）	sin,cos,边界检查
B1060	爱丁顿数 （25 分）	数学规律分析
B1062	最简分数 （20 分）	小数精度有限，利用乘法

1003 我要通过！ （20 分）

“答案正确”是自动判题系统给出的最令人欢喜的回复。本题属于 PAT 的“答案正确”大派送 —— 只要读入的字符串满足下列条件，系统就输出“答案正确”，否则输出“答案错误”。
得到“答案正确”的条件是：

1. 字符串中必须仅有 `P`、 `A`、 `T`这三种字符，不可以包含其它字符；

2. 任意形如 `xPATx` 的字符串都可以获得“答案正确”，其中 `x` 或者是空字符串，或者是仅由字母 A 组成的字符串；
3. 如果 `aPbTc` 是正确的，那么 `aPbATca` 也是正确的，其中 `a`、 `b`、 `c` 均或者是空字符串，或者是仅由字母 A 组成的字符串。

现在就请你为 PAT 写一个自动裁判程序，判定哪些字符串是可以获得“答案正确”的。

输入格式：

每个测试输入包含 1 个测试用例。第 1 行给出一个正整数 $n(<10)$，是需要检测的字符串个数。接下来每个字符串占一行，字符串长度不超过 100，且不包含空格。

输出格式：

每个字符串的检测结果占一行，如果该字符串可以获得“答案正确”，则输出 `YES`，否则输出 `NO`。

输入样例：

```
8
PAT
PAAT
AAPATAA
AAPAATAAAA
xPATx
PT
Whatever
APAAATAA
```

输出样例：

```
YES
YES
YES
YES
NO
NO
NO
NO
```

关键点

如何理解3个条件：

1. 字符串中必须仅有 `P`、 `A`、 `T` 这三种字符，不可以包含其它字符；

- 应该保证只含有这三个字符，map可以解决。

2. 任意形如 `xPATx` 的字符串都可以获得“答案正确”，其中 `x` 或者是空字符串，或者是仅由字母 A 组成的字符串；

- 也就是 `PAT` 一定是满足的，在此基础上，`APATA`，`AAPATAA`，`AAAPATAAA`，……等均满足。

3. 如果 `aPbTc` 是正确的，那么 `aPbATca` 也是正确的，其中 `a`、 `b`、 `c` 均或者是空字符串，或者是仅由字母 A 组成的字符串。

- 由于条件1、2限制，最少的符合是`PAT`形式，那么：

```
PAT      a:空    b:A     c:空
PAAT     a:空    b:AA    c:空
PAAAT    a:空    b:AA    c:空
……
```

- 满足条件3，正确。随后当a和c不为空，首先应该满足条件2，然后运用条件3，以`AAPATAA`为例：

```
AAPATAA       a:AA    b:A     c:AA      条件2
AAPAATAAAA    a:AA    b:AA    c:AAAA    条件3
```

- 接下来列出整个集合表示，来找规律。

```
APATA      AAPATAA      AAAPATAAA       ……      条件2
APAATAA    AAPAATAAAA   AAAPAATAAAAAA   ……      条件3
APAAATAAA  AAPAAATAAAAAA  AAAPAAATAAAAAAAAA  ……  条件3
……
```

  我们发现，`PT`之间每次加一个`A`，`T`后面要加一个`a`，所以可以总结一个规律：P前A的个数*PT之间A的个数 = T后A的个数，进而转化成数学乘法问题，而不需要每次去判断是不是PAT。

1034 有理数四则运算 （20 分）

本题要求编写程序，计算 2 个有理数的和、差、积、商。

输入格式：

输入在一行中按照 `a1/b1 a2/b2` 的格式给出两个分数形式的有理数，其中分子和分母全是整型范围内的整数，负号只可能出现在分子前，分母不为 0。

输出格式：

分别在 4 行中按照 `有理数1 运算符 有理数2 = 结果` 的格式顺序输出 2 个有理数的和、差、积、商。注意输出的每个有理数必须是该有理数的最简形式 `k a/b`，其中 `k` 是整数部分，`a/b` 是最简分数部分；若为负数，则须加括号；若除法分母为 0，则输出 `Inf`。题目保证正确的输出中没有超过整型范围的整数。

输入样例 1：

```
2/3 -4/2
```

输出样例 1：

```
2/3 + (-2) = (-1 1/3)
2/3 - (-2) = 2 2/3
2/3 * (-2) = (-1 1/3)
2/3 / (-2) = (-1/3)
```

输入样例 2：

```
5/3 0/6
```

输出样例 2：

```
1 2/3 + 0 = 1 2/3
1 2/3 - 0 = 1 2/3
1 2/3 * 0 = 0
1 2/3 / 0 = Inf
```

注意点

- 使用分数的固定加减乘除的形式，问题会变得很简单。但是代码会相对较长，不能着急。

分数计算核心代码

```
#include <cstdio>
#include <algorithm>
using namespace std;
typedef long long ll;
/*最小公约数*/
int gcd(ll a,ll b){
    return b == 0 ? a : gcd(b, a%b);
}
struct Fraction{
    ll up;
    ll down;
};
/*分数化简*/
Fraction reduction(Fraction result){
    if (result.down < 0){
        result.up = -result.up;
        result.down = -result.down;
    }
    if (result.up == 0){
        result.down = 1;
    }
    else{
        int d = gcd(abs(result.up), abs(result.down));
        result.up /= d;
        result.down /= d;
    }
    return result;
}
/*分数加法*/
Fraction add(Fraction f1, Fraction f2){
    Fraction result;
    result.up = f1.up*f2.down + f2.up*f1.down;
```

```
    result.down = f1.down*f2.down;
    return reduction(result);//结果别忘了化简
}
/*分数减法*/
Fraction sub(Fraction f1, Fraction f2){
    Fraction result;
    result.up = f1.up*f2.down - f2.up*f1.down;
    result.down = f1.down*f2.down;
    return reduction(result);
}
/*分数乘法*/
Fraction mult(Fraction f1, Fraction f2){
    Fraction result;
    result.up = f1.up*f2.up;
    result.down = f1.down*f2.down;
    return reduction(result);
}
/*分数除法*/
Fraction divd(Fraction f1, Fraction f2){
    Fraction result;
    result.up = f1.up*f2.down;
    result.down = f1.down*f2.up;
    return reduction(result);
}
void show(Fraction f){
    f = reduction(f);
    if (f.up < 0) printf("(");
    if (f.down == 1){//f.down == 1: 既判断了0, 也判断了正数
        printf("%lld",f.up);
    }
    else if (abs(f.up) > abs(f.down)){
        printf("%lld", f.up / f.down);
        printf(" %lld/%lld", abs(f.up%f.down), f.down);
    }
    else{
        printf("%lld/%lld", f.up, f.down);
    }
    if (f.up < 0) printf(")");
}
```

1040 有几个PAT （25 分）

字符串 `APPAPT` 中包含了两个单词 `PAT`，其中第一个 `PAT` 是第 2 位(`P`)，第 4 位(`A`)，第 6 位(`T`)；第二个 PAT 是第 3 位(`P`)，第 4 位(`A`)，第 6 位(`T`)。

现给定字符串，问一共可以形成多少个 `PAT`？

输入格式：

输入只有一行，包含一个字符串，长度不超过10^5，只包含 `P`、`A`、`T` 三种字母。

输出格式：

在一行中输出给定字符串中包含多少个 `PAT`。由于结果可能比较大，只输出对 1000000007 取余数的结果。

输入样例：

```
APPAPT
```

输出样例：

```
2
```

关键点

- 暴力破解是不可能暴力的，这辈子都不会暴力的。还是分析下他的数学规律吧。
- 首先，应该看到它的有序型，PAT是分先后顺序的，有点动态规划的意思，也就是，没遇到A前，P个数是被统计的，A以后的P是遇到下一个A才会被统计。考虑`APAPAT`返回应该是3。所以有每次遇到A前，统计P个数；遇到A时，应统计PA个数，也就是之前的PA和当前组成PA的个数；遇到T时，应统计PAT个数，也就是之前的PAT和当前PAT个数。

完整代码

```
#include <cstdio>
#include <cstring>
using namespace std;
const int maxn = 100010;
char ch[maxn];
int main()
{
    scanf("%s",ch);
    int len = strlen(ch);
    int res = 0, P = 0, PA = 0;
    for (int i = 0; i < len; i++){
        if (ch[i] == 'P') P++;
        if (ch[i] == 'A') PA = (PA + P) % 1000000007;//这里很巧妙地解决了PA的顺序出现。要慢慢理解
        if (ch[i] == 'T') res = (res + PA) % 1000000007;//这里很巧妙地解决了PAT的顺序出现。
    }
    printf("%d",res);
    return 0;
}
```

1045 快速排序 （25 分）

著名的快速排序算法里有一个经典的划分过程：我们通常采用某种方法取一个元素作为主元，通过交换，把比主元小的元素放到它的左边，比主元大的元素放到它的右边。 给定划分后的 N 个互不相同的正整数的排列，请问有多少个元素可能是划分前选取的主元？
例如给定 $N = 5$，排列是1、3、2、4、5。则：

- 1 的左边没有元素，右边的元素都比它大，所以它可能是主元；
- 尽管 3 的左边元素都比它小，但其右边的 2 比它小，所以它不能是主元；
- 尽管 2 的右边元素都比它大，但其左边的 3 比它大，所以它不能是主元；
- 类似原因，4 和 5 都可能是主元。

因此，有 3 个元素可能是主元。
输入格式：
输入在第 1 行中给出一个正整数 N ($\leq 10^5$)； 第 2 行是空格分隔的 N 个不同的正整数，每个数不超过 10^9。
输出格式：
在第 1 行中输出有可能是主元的元素个数；在第 2 行中按递增顺序输出这些元素，其间以 1 个空格分隔，行首尾不得有多余空格。
输入样例：

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

输出样例：

```
3
1 4 5
```

关键点

- 这个和B1040都是动态规划类似问题，需要想清楚所需前置结果。
- 这里再建立两个数组，分别存储每一个数字的前i-1项的最大值和后n-i项的最小值，这样可以利用动态规划的思想，分别建立这两个数组。

部分实现

```
leftMax[0] = a[0];
for (int i = 1; i < N; i++){//用于存储每一个数字左边的最大值
leftMax[i] = a[i] > leftMax[i - 1] ? a[i] : leftMax[i - 1];
}
rightMin[N - 1] = a[N - 1];
for (int i = N - 2; i >= 0; i--){//用于存储每一个数字右边的最小值
rightMin[i] = a[i] < rightMin[i + 1] ? a[i] : rightMin[i + 1];
}
```

- 然后即可直接比较得出结果，这里题目虽然要求有序，但是，仔细想想，这样得到的结果就是有序的，因为，如果无序，那么他肯定不满足左小右大这个条件的。

1049 数列的片段和 （20 分）

给定一个正数数列，我们可以从中截取任意的连续的几个数，称为片段。例如，给定数列 `{ 0.1, 0.2, 0.3, 0.4 }`，我们有 `(0.1) (0.1, 0.2) (0.1, 0.2, 0.3) (0.1, 0.2, 0.3, 0.4) (0.2) (0.2, 0.3) (0.2, 0.3, 0.4) (0.3) (0.3, 0.4) (0.4)` 这 10 个片段。

给定正整数数列，求出全部片段包含的所有的数之和。如本例中 10 个片段总和是 `0.1 + 0.3 + 0.6 + 1.0 + 0.2 + 0.5 + 0.9 + 0.3 + 0.7 + 0.4 = 5.0`。

输入格式：

输入第一行给出一个不超过 10^5 的正整数 N，表示数列中数的个数，第二行给出 N 个不超过 1.0 的正数，是数列中的数，其间以空格分隔。

输出格式：

在一行中输出该序列所有片段包含的数之和，精确到小数点后 2 位。

输入样例：

```
4
0.1 0.2 0.3 0.4
```

输出样例：

```
5.00
```

关键点

- 需要发现规律，如果全部找出在计算，三层循环肯定超时。动态规划我没有考虑。分析如下，假设5个数，（用整数代替小数便于观看）：

```
 1      2        3          4             5
(1)    (1 2)    (1 2 3)    (1 2 3 4)     (1 2 3 4 5)   1:5次(N*1)
(2)    (2 3)    (2 3 4)    (2 3 4 5)                   2:4+4=8次((N-1)*2)
(3)    (3 4)    (3 4 5)                                3:3+3+3=9次((N-2)*3)
(4)    (4 5)                                           4:2*4=8次((N-3)*4)
(5)                                                    5:1*5=5次((N-4)*5)
```

  根据这个枚举，很容易发现规律，对于每一个元素i，出现次数为(N-i+1)*i次。

注意点

- 一直有两个点不过，将所有的数据类型全部改为double类型，通过了。**说明是类型或者基本可以感觉到是溢出问题。**
  为什么会溢出呢？$10^{5^2}=10^{10}$ = 100亿超过了int范围。其实这个极限在i=N/2的时候取到的，此时(N-i+1)*i最大，当$N=10^5$，则$i=2.5^5$，则$(N-i+1)*i\approx 2.5^{10}$超过了int范围。

1051 复数乘法 （15 分）

复数可以写成（A+Bi）的常规形式，其中 A 是实部，B 是虚部，i 是虚数单位，满足$i^2=-1$；也可以写成极坐标下的指数形式 $(R\times e^{(Pi)})$，其中 R 是复数模，P 是辐角，i 是虚数单位，其等价于三角形式 $(R(cos(P)+isin(P))$。

现给定两个复数的 R 和 P，要求输出两数乘积的常规形式。

输入格式：

输入在一行中依次给出两个复数的 R1,P1,R2,P2，数字间以空格分隔。

输出格式：

在一行中按照 `A+Bi` 的格式输出两数乘积的常规形式，实部和虚部均保留 2 位小数。注意：如果 `B` 是负数，则应该写成 `A-|B|i` 的形式。

输入样例：

```
2.3 3.5 5.2 0.4
```

输出样例：

```
-8.68-8.23i
```

关键点

1. 题目的理解，需要你借助函数cos()和sin()来处理(R(cos(P)+isin(P))，由于计算乘法，那就使用乘法分配律。

$$(R1(cos(P1)+isin(P1))*(R2(cos(P2)+isin(P2))=$$

$$R1R2cos(P1)cos(P2)+$$

$$i^2*R1R2sin(P1)sin(P2)+$$

$$i*(R1R2sin(P1)cos(P2)+R1R2cos(P1)sin(P2))$$

为`(A+Bi)`形式，且$i^2=-1$，所以可以分别计算A和B的值。

2. 边界检测。因为保留2位小数，且double的舍入是符合四舍五入的。计算结果A和B可能为小于0.005的小数，因为0.005会舍入为0.01；-0.004会被舍入为-0.00，其实负号是多余的，所以应该对abs(A)判断是否小于0.005，来决定是否直接输出0.00。另外，对B的判断，要比A多了一个加号的判断，因为其前不会自动加`+`号。

1060 爱丁顿数 （25 分）

英国天文学家爱丁顿很喜欢骑车。据说他为了炫耀自己的骑车功力，还定义了一个“爱丁顿数” E ，即满足有 E 天骑车超过 E 英里的最大整数 E。据说爱丁顿自己的 E 等于87。

现给定某人 N 天的骑车距离，请你算出对应的爱丁顿数 E（$\le N$）。

输入格式：

输入第一行给出一个正整数 $N(\le 10^5)$，即连续骑车的天数；第二行给出 N 个非负整数，代表每天的骑车距离。

输出格式：

在一行中给出 N 天的爱丁顿数。
输入样例：

```
10
6 7 6 9 3 10 8 2 7 8
```

输出样例：

```
6
```

关键点

- 首先要知道这道题要排序，如果不知道排序，那么就在做不了。
- 排序后，其实就已经可以按照题目要求直接去找爱丁顿数了。
- 但这不是最优解，理论上你要发现它的规律，是去找从大到小排序`(1~N)`中`a[i]>i`的最后一次出现的i。

不用发现规律

- 虽然可以这么做，但是，需要优化一个地方，不容易去找，不过，好就好在如果看不到规律，这个方法还是不错的。
- 主要思路，是直接模拟，并用倒着判断的方法去优化。

发现规律

- 一旦看到了规律，这个题目就已经失去意义了。
  分析：从下标1开始存储n天的公里数在数组a中，对n个数据从大到小排序，i表示了骑车的天数，那么满足$a[i] > i$的最大值即为所求。

1062 最简分数 （20 分）

一个分数一般写成两个整数相除的形式：N/M，其中 M 不为0。最简分数是指分子和分母没有公约数的分数表示形式。
现给定两个不相等的正分数 N_1/M_1 和 N_2/M_2，要求你按从小到大的顺序列出它们之间分母为 K 的最简分数。
输入格式：
输入在一行中按 N/M 的格式给出两个正分数，随后是一个正整数分母 K，其间以空格分隔。题目保证给出的所有整数都不超过 1000。
输出格式：
在一行中按 N/M 的格式列出两个给定分数之间分母为 K 的所有最简分数，按从小到大的顺序，其间以 1 个空格分隔。行首尾不得有多余空格。题目保证至少有 1 个输出。

输入样例：

```
7/18 13/20 12
```

输出样例：

```
5/12 7/12
```

注意点

- 对于分数的判断，不能局限于计算小数比较，因为除不尽会出现精度问题，所以应该利用乘法来解决大小问题。
- 此处给出的分数大小不一定是正序，需要调整顺序。就可以利用乘法性质。
-

```
if (n1*m2 > n2*m1){
    swap(n1,n2);
    swap(m1, m2);
}
```

- 猜测：一直想不通的点，我一直用除法来解决边界问题，这样其实并不好，在一定程度上会出现小数精度不够，所以用乘法来解决分数的大小比较。

类型十二：素数

- 素数的判断
- 素数表的生成

素数的判断

- 不需要全部判断，只需要判断2~$\sqrt{x}$是否被整除，就可以判断是否为素数了。

```
bool isPrime(int x){
    if (x < 2) return false;
    for (int i = 2; i*i <= x; i++){
        if (x % i == 0) return false;
    }
    return true;
}
```

素数表的生成

- 埃拉托斯特尼筛法

```
/*生成前100100个素数，需要测试的前面10000000一千万个数！ */
const int maxn = 10000000;
bool isPrime[maxn] = {};
int prime[100100] = {};
int len = 0;
void getPrime(){
    for (int i = 2; i < maxn&&len < 100100; i++){
        if (!isPrime[i]){
            prime[len++] = i;
            for (int j = i+i; j < maxn; j+=i){
                isPrime[j] = true;
            }
        }
    }
}
```

典型题目（不难）

题号	题目	说明
B1007	素数对猜想 （20 分）	素数表的生成，注意取值范围
B1013	数素数 （20 分）	素数表的生成，注意取值范围

B1094	谷歌的招聘 （20 分）	边界检测(long long)，输出格式

类型十三：大整数运算

- 主要以数组形式的大整数的进位加减运算。

典型题目（不难）

题号	题目	说明
B1017	A除以B （20 分）	边界检测
B1079	延迟的回文数 （20 分）	`reverse()`反转函数需要`<algorithm>`库支持

类型十四：中级模拟

比简单模拟要难些，需要注意

题号	题目	说明
B1058	选择题 （20 分）	按照格式读取更简单
B1089	1089 狼人杀-简单版 （20 分）	逻辑分析
B1073	1073 多选题常见计分法 （20 分）	借助`xor`（异或）解决多选半对问题

1089 狼人杀-简单版 （20 分）

以下文字摘自《灵机一动·好玩的数学》：“狼人杀”游戏分为狼人、好人两大阵营。在一局“狼人杀”游戏中，1 号玩家说：“2 号是狼人”，2 号玩家说：“3 号是好人”，3 号玩家说：“4 号是狼人”，4 号玩家说：“5 号是好人”，5 号玩家说：“4 号是好人”。已知这 5 名玩家中有 2 人扮演狼人角色，有 2 人说的不是实话，有狼人撒谎但并不是所有狼人都在撒谎。扮演狼人角色的是哪两号玩家？
本题是这个问题的升级版：已知 N 名玩家中有 2 人扮演狼人角色，有 2 人说的不是实话，有狼人撒谎但并不是所有狼人都在撒谎。要求你找出扮演狼人角色的是哪几号玩家？
输入格式：
输入在第一行中给出一个正整数 N（5≤N≤100）。随后 N 行，第 i 行给出第 i 号玩家说的话（1≤i≤N），即一个玩家编号，用正号表示好人，负号表示狼人。
输出格式：
如果有解，在一行中按递增顺序输出 2 个狼人的编号，其间以空格分隔，行首尾不得有多余空格。如果解不唯一，则输出最小序列解 —— 即对于两个序列 A=a[1],⋯,a[M] 和 B=b[1],⋯,b[M]，若存在 0≤k<M 使得 a[i]=b[i] （i≤k），且 a[k+1]<b[k+1]，则称序列 A 小于序列 B。若无解则输出 No Solution。

输入样例 1：

```
5
-2
+3
-4
+5
+4
```

输出样例 1：

```
1 4
```

输入样例 2：

```
6
+6
```

```
+3
+1
-5
-2
+4
```

输出样例 2（解不唯一）：

```
1 5
```

输入样例 3：

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

输出样例 3：

```
No Solution
```

问题分析

- 分析：每个人说的数字保存在v数组中，i从1～n、j从i+1～n遍历，分别假设i和j是狼人，a数组表示该人是狼人还是好人，等于1表示是好人，等于-1表示是狼人。k从1～n分别判断k所说的话是真是假，k说的话和真实情况不同（即v[k] * a[abs(v[k])] < 0）则表示k在说谎，则将k放在lie数组中；遍历完成后判断lie数组，如果说谎人数等于2并且这两个说谎的人一个是好人一个是狼人（即a[lie[0]] + a[lie[1]] == 0）表示满足题意，此时输出i和j并return，否则最后的时候输出No Solution～

```cpp
#include <cstdio>
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;
int main() {
    int n;
    cin >> n;
    vector<int> v(n+1);
    for (int i = 1; i <= n; i++) cin >> v[i];
    for (int i = 1; i <= n; i++) {
        for (int j = i + 1; j <= n; j++) {
            vector<int> lie, a(n + 1, 1);
            a[i] = a[j] = -1;
            for (int k = 1; k <= n; k++)
```

```
                if (v[k] * a[abs(v[k])] < 0) lie.push_back(k);
            if (lie.size() == 2 && a[lie[0]] + a[lie[1]] == 0) {
                cout << i << " " << j;
                return 0;
            }
        }
    }
    cout << "No Solution";
    return 0;
}
```

1073 多选题常见计分法 （20 分）

批改多选题是比较麻烦的事情，有很多不同的计分方法。有一种最常见的计分方法是：如果考生选择了部分正确选项，并且没有选择任何错误选项，则得到 50% 分数；如果考生选择了任何一个错误的选项，则不能得分。本题就请你写个程序帮助老师批改多选题，并且指出哪道题的哪个选项错的人最多。

输入格式：

输入在第一行给出两个正整数 N（≤1000）和 M（≤100），分别是学生人数和多选题的个数。随后 M 行，每行顺次给出一道题的满分值（不超过 5 的正整数）、选项个数（不少于 2 且不超过 5 的正整数）、正确选项个数（不超过选项个数的正整数）、所有正确选项。注意每题的选项从小写英文字母 a 开始顺次排列。各项间以 1 个空格分隔。最后 N 行，每行给出一个学生的答题情况，其每题答案格式为 `(选中的选项个数 选项1 ……)`，按题目顺序给出。注意：题目保证学生的答题情况是合法的，即不存在选中的选项数超过实际选项数的情况。

输出格式：

按照输入的顺序给出每个学生的得分，每个分数占一行，输出小数点后 1 位。最后输出错得最多的题目选项的信息，格式为：`错误次数 题目编号（题目按照输入的顺序从1开始编号）-选项号`。如果有并列，则每行一个选项，按题目编号递增顺序输出；再并列则按选项号递增顺序输出。行首尾不得有多余空格。如果所有题目都没有人错，则在最后一行输出 `Too simple`。

输入样例 1：

```
3 4
3 4 2 a c
2 5 1 b
5 3 2 b c
1 5 4 a b d e
(2 a c) (3 b d e) (2 a c) (3 a b e)
(2 a c) (1 b) (2 a b) (4 a b d e)
(2 b d) (1 e) (1 c) (4 a b c d)
```

输出样例 1：

```
3.5
6.0
2.5
2 2-e
```

```
2 3-a
2 3-b
```

输入样例 2：

```
2 2
3 4 2 a c
2 5 1 b
(2 a c) (1 b)
(2 a c) (1 b)
```

输出样例 2：

```
5.0
5.0
Too simple
```

关键点

- 用二进制来解决该问题，涉及异或，或等操作，EDCBA可以用二进制(11111)来表示，这样的好处是，直接解决了该问题最核心的一点，半对的判断，因为异或的结果为0，说明，是全部正确的；不为零，则分为有错或者半对，其中这个结果明确了哪些位（即选项）是错误的；另外，半对是不全导致的，那么半对的判断则需要借助`或`来判断 - `（异或结果 | 正确答案） == 正确答案`。

```cpp
#include <cstdio>
#include <cstring>
#include <set>
#include <vector>
#include <map>
#include <ctype.h>
using namespace std;
const int maxm = 110;
const int maxn = 1010;
struct question{
    int full;//总得分
    double half;//半分
    int cnt;//选项个数
    int right;//正确个数
    int answer;
}q[maxm];
int a[maxm][5] = {};
int main()
{
    int M, N;
    scanf("%d%d", &N, &M);
    char ch;
```

```
    for (int i = 0; i < M; i++){
        scanf("%d%d%d", &q[i].full, &q[i].cnt, &q[i].right);
        q[i].half = (double)q[i].full / 2.0;
        for (int j = 0; j < q[i].right; j++){
            scanf(" %c)", &ch);
            q[i].answer += 1 << (ch - 'a');
        }
    }

    int maxi = -1;
    for (int i = 0; i < N; i++){
        int num;
        double sum = 0;
        for (int j = 0; j < M; j++){
            getchar();
            scanf("(%d", &num);
            int answer = 0;
            for (int k = 0; k < num; k++){
                scanf(" %c)", &ch);
                answer += 1 << (ch - 'a');
            }
            int el = answer ^ q[j].answer;
            if (el){//异或不为0，说明答案不全或者错误
                if ((answer | q[j].answer) == q[j].answer) sum += q[j].ha
lf;//或为0，答案不全
                if (el){//记录错误选项
                    for (int k = 0; k < 5; k++){
                        a[j][k] += ((el >> k) & 1);
                        maxi = a[j][k] > maxi ? a[j][k] : maxi;
                    }
                }
            }
            else{
                sum += q[j].full;
            }
        }
        printf("%.1f\n", sum);
    }
    if (maxi == -1){
        printf("Too simple\n");
        return 0;
    }
    else{
        for (int i = 0; i < M; i++){
            for (int j = 0; j < 5; j++){
                if (a[i][j] == maxi){
                    printf("%d %d-%c\n", maxi,i+1,j+'a');
                }
            }
```

```
            }

        }
        return 0;
    }
```

类型十五：常用stl

PAT最常用的考点，需要注意。

题号	题目	说明
B1090	危险品装箱 （25 分））	PAT甲级题目，去年考试原题，当时错了1分

1090 危险品装箱 （25 分）

集装箱运输货物时，我们必须特别小心，不能把不相容的货物装在一只箱子里。比如氧化剂绝对不能跟易燃液体同箱，否则很容易造成爆炸。
本题给定一张不相容物品的清单，需要你检查每一张集装箱货品清单，判断它们是否能装在同一只箱子里。
输入格式：
输入第一行给出两个正整数：$N(\le 10^4)$ 是成对的不相容物品的对数；M（≤100）是集装箱货品清单的单数。随后数据分两大块给出。第一块有 N 行，每行给出一对不相容的物品。第二块有 M 行，每行给出一箱货物的清单，格式如下：

```
K G[1] G[2] ⋯ G[K]
```

其中 K（≤1000）是物品件数，G[i] 是物品的编号。简单起见，每件物品用一个 5 位数的编号代表。两个数字之间用空格分隔。
输出格式：
对每箱货物清单，判断是否可以安全运输。如果没有不相容物品，则在一行中输出 Yes，否则输出 No。
输入样例：

```
6 3
20001 20002
20003 20004
20005 20006
20003 20001
20005 20004
20004 20006
4 00001 20004 00002 20003
5 98823 20002 20003 20006 10010
3 12345 67890 23333
```

输出样例：

```
No
Yes
Yes
```

注意点

- 因为每一个物品不只有一个键值，所以应该声明为`map<int, set<int>> mp;`类型存储匹配对。
- 对于最后的判断，应该对每次找出来的set中的每一个元素判断，是否出现在清单中。

完整代码

```
#include <cstdio>
#include <set>
#include <vector>
#include <map>
using namespace std;
int main()
{
    int N, M, K;
    scanf("%d%d", &N, &M);
    map<int, set<int>> mp;
    int x, y;
    for (int i = 0; i < N; i++){
        scanf("%d%d", &x, &y);
        mp[x].insert(y);
        mp[y].insert(x);
    }
    for (int i = 0; i < M; i++){
        scanf("%d", &K);
        set<int> s;
        for (int j = 0; j < K; j++){
            scanf("%d", &x);
            s.insert(x);
        }
        bool secure = true;
        for (auto a : s){//物品清单中的元素
            for (auto b : mp[a]){//所对应的匹配对中的每一个元素
                if (s.find(b) != s.end()){//清单中不含匹配对中的元素
                    secure = false;
                    break;
                }
            }
        }
        if (secure) printf("Yes\n");
        else printf("No\n");
    }
    return 0;
}
```

类型十六：链表

- 伪链表（也叫**静态链表**），他和链表区别是，本质上是数组，维护了链表的链式关系，但是操作它可以按照数组的方式，那么有些问题就会变得简单。
- 链表结构，掌握链表的增删改查等基本操作，很重要。

题号	题目	说明
B1025	反转链表 （25 分）	静态链表
B1075	链表元素分类 （25 分）	静态链表

1025 反转链表 （25 分）

给定一个常数 K 以及一个单链表 L，请编写程序将 L 中每 K 个结点反转。例如：给定 L 为 1→2→3→4→5→6，K 为 3，则输出应该为 3→2→1→6→5→4；如果 K 为 4，则输出应该为 4→3→2→1→5→6，即最后不到 K 个元素不反转。

输入格式：
每个输入包含 1 个测试用例。每个测试用例第 1 行给出第 1 个结点的地址、结点总个数正整数 $N(\le 10^5)$、以及正整数 $K(\le N)$，即要求反转的子链结点的个数。结点的地址是 5 位非负整数，NULL 地址用 -1 表示。
接下来有 N 行，每行格式为：

```
Address Data Next
```

其中 Address 是结点地址，Data 是该结点保存的整数数据，Next 是下一结点的地址。
输出格式：
对每个测试用例，顺序输出反转后的链表，其上每个结点占一行，格式与输入相同。
输入样例：

```
00100 6 4
00000 4 99999
00100 1 12309
68237 6 -1
33218 3 00000
99999 5 68237
12309 2 33218
```

输出样例：

```
00000 4 33218
33218 3 12309
12309 2 00100
00100 1 99999
```

```
99999 5 68237
68237 6 -1
```

关键点

- 这种题存在技巧，如果按照链表的方式直接操作，那么可能会涉及链表的插入删除等操作，而且会很复杂，但是如果利用一种**伪链表（也叫静态链表）**，他和链表区别是，本质上是数组，维护了链表的链式关系，但是操作它可以按照数组的方式，那么有些问题就会变得简单。
- 利用静态链表反转，直接将每一个地址顺序存储到数组中，将其对应的数据按照散列的方式存储。
- 反转时，需要找到反转规律。例如k=4时：

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

对于每一行索引i与k的关系：

```
0交换k-1(索引3)
1交换k-2(索引2)

4交换2k-1(索引7)
5交换2k-2(索引6)
```

这是一个数组反转问题了，只需要折半反转就行了。
可以利用一个二次循环解决：

```
//设数组a[]为静态链表
for (int i = 0; i < len && (i + K - 1) < len;i+=K){//对每一组反转
    int x = i / K + 1;//得出k前系数
    for (int j = 0; j < K / 2;j++){//组内反转
        swap(a[i+j],a[x*K-1-j]);//反转
    }
}
```

- 输出，只需要a和data数组就可以解决输出问题了，注意最后一个节点的下一个地址为`-1`。