写在前面的话

这次上机字符串处理的题较多，希望做题不顺利的同学能够在课后再思考思考。

主要可以关注的地方有：

- 字符串读入。包括scanf读入、fgets读入、循环+getchar读入等等。
- 字符串处理库函数。常用的有：strlen、strcmp、strcpy、strncmp、strncpy、strstr、strchr。
- char也是一个“整型”变量，只不过它能表示的值最大只有127，最小只有-128。每一个由单引号括起来的单字符都相当于一个常数，编译器知道这个常数。

此外，随着本课程进度的推进，编程的复杂程度也在提升。希望大家在空余时间可以尝试对代码风格和规范等进行了解，比如变量命名、缩进与空格的常用约定等等。好的代码风格和习惯将会大幅提高代码的可读性、可维护性和可扩展性，同时会降低出bug的概率和debug的难度。

学有余力的同学还可以尝试理解函数的封装解耦作用，领悟这种程序设计的思维。

最后祝大家学习顺利！

A. vector's dot product

题目描述

在这个问题中，您将获得多组数据。

对于每组数据，您将得到两个向量，比如$\vec{a}=(x_1,x_2，\ \ldots,x_n)$，$\vec{b}=(y_1,y_2，\ \ldots,y_n)$ ，您需要得到它们的点积。

$$\vec{a}\cdot\vec{b}=\sum_{i=1}^{n}x_iy_i$$

输入

第一行包含一个整数T，其表示待计算向量组数. $T\in(0,50)$

对于下面的T行，每一行第一个整数为n, $n\in(0,50)$。(n表示这个向量的元素的个数)，每一行包含其他$2n$个整数，每一个整数范围为（-200，200），前n个数表示$\vec{a}$后n个数表示$\vec{b}$。每个整数都用空格隔开。

输出

对于每组数据，您需要输出一行一个整数，该整数表示$\vec{a}\cdot\vec{b}$的值。

输入样例

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

输出样例

```
5
3
```

英文版

Description

In this question, you will get multiple sets of data。

To every set of data ,you will get two vectors, such as $\vec{a}=(x_1,x_2,\ldots,x_n),\vec{b}=(y_1,y_2,\ldots,y_n)$, and you need to get their dot product.

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} x_i y_i$$

Input

The first line contains one integer which is named after T. $T \in (0,50)$

To the following n lines ,every line contains one integers which is named after n $n \in (0,50)$. (n stands for the number of elements of this vector.), and every line contains other $2n$ integers, which stand for $\vec{a}$ and $\vec{b}$. Every integer is separated by space.

Output

To every set of data, you need to output a line and print one integer which stands for $\vec{a} \cdot \vec{b}$

Sample of Input

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

Sample of Output

```
5
3
```

解题思路

对于向量点积的运算，只需要计算各个分量的和，这里需要注意的是先读完所有的x再读y，因此不能每读两个数就直接乘积相加，要存进数组里，待读取完后再运算。

另外，每组输出之后记得换行。

AC代码

```
#include <stdio.h>

int main()
{
        int x[55], y[55];
        int T, n, i, ans;

        scanf("%d", &T);
        while(T--)
        {
                scanf("%d", &n);
```

```
                for(i = 0; i < n; i++) scanf("%d", &x[i]);
                for(i = 0; i < n; i++) scanf("%d", &y[i]);
                ans = 0;
                for(i = 0; i < n; i++) ans += x[i] * y[i];
                printf("%d\n", ans);
        }
}
```

B. 统 计 成 绩

题目描述

猪脚收集了全年级的成绩表，ta想请你帮忙统计一下每门课的最高、最低、平均成绩和每个同学的平均成绩。

输入

共n+1行。

第一行：两个整数n($1 \leq n \leq 1000$)，m($1 \leq m \leq 20$)，表示共有n名同学，m门课程。

接下来n行，每行m个整数。这n行中，第i行的第j个数表示第i名同学第j门课的成绩Cij($0 \leq c \leq 100$)。

输出

共m+1行。

前m行，每行三个数，依次表示每门课的最高、最低和平均成绩（最高、最低成绩为整数，平均成绩保留小数点后两位）。

第m+1行，共n个数，分别表示第1-n名同学的平均成绩（保留小数点后两位）。

输入样例

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

输出样例

```
1 1 1.00
2 2 2.00
3 3 3.00
4 2 3.60
5 1 4.20
3.00 3.00 3.00 3.00 1.80
```

解题思路

这道题需要我们统计每节课成绩的最大值、最小值、平均值，和每名同学的平均值。朴素的想法是将所有数值读入二维数组中，在按行按列进行统计。这里给出一种不需要二维数组的方法，需要在每次读入新数据时，对

当前每门课的最值、总分进行更新。

对于二维的情况，在运算过程中一定要注意**不要把横纵搞错**。

AC代码

```
#include<stdio.h>
int main(){
        int n, m;
        scanf("%d%d", &n, &m);
        int i, j, max[25], min[25], sum[25] = {}, sums[1005] = {};
        for(j = 0; j < m; j++)max[j] = 0, min[j] = 100;
        for(i = 0; i < n; i++){
                for(j = 0; j < m; j ++){
                        int a;
                        scanf("%d", &a);
                        sums[i] += a;
                        sum[j] += a;
                        max[j] = max[j] > a ? max[j] : a;
                        min[j] = min[j] < a ? min[j] : a;
                }
        }
        for(j = 0; j < m; j++) printf("%d %d %.2f\n", max[j], min[j],
(double)sum[j] / n);
        for(i = 0; i < n; i++) printf("%.2f ", (double)sums[i] / m);
}
```

C. 最大值

题目描述

你得到了一个长度为n的正整数序列$a_1, a_2, \ldots, a_n$ 。对于每个整数$i \in [1, n]$，你需要回答下列问题：找出序列中除了a_i以外其他$n-1$个元素的最大值。

输入

输入多组数据，每组数据输入两行。

第一行，一个正整数$n(2 \leq n \leq 1000000)$，表示序列的长度。

第二行，n个正整数$a_1, a_2, \ldots, a_n$ ，之间用一个空格隔开。保证a_i在int范围内。

输出

每组数据输出一行。输出n个整数，每个整数之间用一个空格隔开，其中第i个整数$(1 \leq i \leq n)$表示序列中除了a_i以外其他$n-1$个元素的最大值。

输入样例

```
3
1 4 3
```

输出样例

```
4 3 4
```

Hint

如果你的做法需要n^2次计算，请考虑更快的做法。

Hint2

很多答案看起来是相同的？

解题思路

通过观察可以发现，若数组最大值唯一，则最大值的答案是次大值，其余值答案均为最大值； 若数组最大值不唯一，则所有答案均为最大值。

因此只需找出数组中前二大的元素即可。

（扩展：如果找前3大如何做？前k大呢？有兴趣的同学可以进一步探究）

AC代码

```
#include<stdio.h>
int a[1000010];
int main()
{
        int n, x, y, cnt;
        while (~scanf("%d",&n))
        {
                cnt = 0;
                for(int i=0; i<n; i++)
                        scanf("%d",&a[i]);
        x = y = a[0];
                for(int i=1; i<n; i++)
                {
                        if(a[i]>x)
                        {
                                y=x;
                                x=a[i];
                        }
            else if(a[i]<x&&a[i]>y)
                y=a[i];
                }

                for(int i=0; i<n; i++)
                        if(a[i]==x) cnt++;

                for(int i=0; i<n; i++)
                {
                        if(a[i]==x&&cnt==1) printf("%d ",y);
                        else printf("%d ",x);
                }
                printf("\n");
        }
        return 0;
}
```

D. Anagrams

题目描述

对两个字符串而言，如果一个字符串是另一个字符串的一个排列，则称两个字符串互为Anagrams。

小K有两个字符串P和Q（仅包含英文字母字符），现在要判断他们是不是互为Anagrams。

举个例子：有且仅有aBc，acB, Bac, Bca, caB, cBa是字符串aBc的排列。

输入

多组数据输入，每组数据两行，保证数据组数$T <= 100$。

第一行，一个字符串P

第二行，一个字符串Q

输出

对于每组数据。

输出一行，表示是否互为Anagrams

是，则输出“TAK”

否，则输出“NIE”

输出忽略引号。

输入样例

```
anagram
ganaram
a
b
```

输出样例

```
TAK
NIE
```

数据范围

$|P|,|Q| \in [1,100]$

Hint

可以考虑排序或者使用数组帮忙计数

多组数据请注意初始化

解题思路

解法一：

首先，如果两个字符串长度不相等，那么他们一定不是$Anagrams$。

我们把字符串P和字符串Q分别从小到大排序，即得到他们字典序最小的排列进行比较，如果相等说明他们互为$Anagrams$，否则不是。

解法二：

我们对于每个出现的字符计数，如果所有的英文字母在这两个串中出现的次数都相同，那么他们互为$Anagrams$，否则不是。

在计数的时候我们用到了类型强制转换，直接在它的$ASCII$码对应的位置加1。这个方法需要注意的是我们题中所说的英文字母，不仅有小写字母，还有大写字母。

AC代码

解法一：

```
#include<stdio.h>
#include<string.h>
int main()
{
        int i,j,lena,lenb;
        char a[105],b[105],c;
        while (~scanf("%s", a)) {
                scanf("%s",b);
                lena=strlen(a);
                lenb=strlen(b);
                for(i=0;i<lena;i++)
                        for(j=i+1;j<lena;++j)
                                if(a[i]>a[j]) {
                                        c=a[i];
                                        a[i]=a[j];
                                        a[j]=c;
                                }
                for(i=0;i<lenb;i++)
                        for(j=i+1;j<lenb;++j)
                                if(b[i]>b[j]) {
                                        c=b[i];
                                        b[i]=b[j];
                                        b[j]=c;
                                }
```

```
                int diff=0;
                if (lena!=lenb) diff=1;
                for(i=0;i<lena&&!diff;++i)
                        if(a[i]!=b[i])
                                diff=1;
                puts(diff?"NIE":"TAK");
        }
        return 0;
}
```

解法二：

```
#include<stdio.h>
#include<string.h>
int main()
{
        int i,lena,lenb,k[130],p[130];
        char a[105],b[105];
        while (~scanf("%s", a)) {
                scanf("%s",b);
                for(i=0;i<130;i++)
                        k[i]=p[i]=0;
                lena=strlen(a);
                lenb=strlen(b);
                for(i=0;i<lena;i++)
                        k[(int)a[i]]++;
                for(i=0;i<lenb;i++)
                        p[(int)b[i]]++;
                int diff=0;
                for(i=0;i<130&&!diff;i++)
                        if(k[i]!=p[i])
                                diff=1;
                puts(diff?"NIE":"TAK");
        }
        return 0;
}
```

E. 李总管所言极是

题目描述

班群内同学们正在讨论一道练习赛的题目，请你在其中找到最长的一句话（不含姓名及英文冒号：的字符个数最多的一句话）。

输入

多行输入。

每行一个字符串，格式为“姓名:所说的内容”，仅包含空格、字母、数字、可输入的英文符号。

输出

共两行。

第一行一个字符串，格式为 最长的一句话的姓名 + Suo Yan Ji Shi。姓名和Suo之间有一个空格。

第二行一个字符串，格式为 最长的一句话。

输入样例

```
WSY:Wo Jue De Wo Shuo De You Dao Li
Xiao Hong:+1
This is a name:HA HA HA HA HA HA HA HA HA
```

输出样例

```
WSY Suo Yan Ji Shi
Wo Jue De Wo Shuo De You Dao Li
```

数据限制

每行字符串长度小于1200。

最多输入1000行。

保证最长的一句话唯一。

保证：前后没有空格。

解题思路

这道题需要我们找出最长的字符串，不同的是字符串长度是冒号之后的长度。 在输出字符串时，根据字符串在\0位置截止的特性，我们可以在冒号位置插入\0截断字符串，然后输出。

AC代码

```
#include<stdio.h>
#include<string.h>
//定义函数为在str寻找c第一次出现的下标
int strchri(char [], char);
int strchri(char str[], char c){
        int a = 0;
        while(str[a] != c) a++;
        return a;
}
int main(){
        char str[1005][1205] = {};
        int pstr[1005];//冒号位置
        int i = 0, n = 0, maxl = 0, j =0 ;
        //读完文件
        while(gets(str[i]) != NULL) i++;
        n = i;
        //寻找长度最大的
        for(i = 0; i < n; i++){
                pstr[i] = strchri(str[i], ':');
                int l= strlen(str[i]) - (pstr[i]);
                if(maxl < l)maxl = l, j = i;
        }
        str[j][pstr[j]] = '\0';
        printf("%s Suo Yan Ji Shi\n", str[j]);
        printf("%s", &str[j][pstr[j] + 1]);
}
```

注意上面的strchri函数可以使用string库中的函数代替，需要使用指针。

```
#include<stdio.h>
#include<string.h>
int main(){
        char str[1005][1205] = {};
        char *pstr[1005];//指向冒号
        int i = 0, n = 0, maxl = 0, j =0 ;
        while(gets(str[i]) != NULL) i++;
        n = i;
        for(i = 0; i < n; i++){
                pstr[i] = strchr(str[i], ':');
                int l= strlen(str[i]) - (pstr[i] - str[i]);
                if(maxl < l)maxl = l, j = i;
        }
        *pstr[j] = '\0';
        printf("%s Suo Yan Ji Shi\n", str[j]);
        printf("%s", pstr[j] + 1);
}
```

F. 今天你"剁手"了吗

题目描述

又到了一年一度的双十一，syb 打算在某宝买一堆好东西，但是无奈商家太多，各商家的价格、质量参差不齐。syb 非常头疼，本着质量第一的原则，他决定对于每件商品，按店铺评分由高到低排序，如果评分相同则按价格从低到高排序，最后选出排名第一的物品加购。这件事做起来比较麻烦，syb 决定把这项重大的任务交给你，请你根据他列出的清单，帮他加购。

输入

第 1 行，一个正整数 n，表示输入数据组数，即 syb 要买 n 种物品。

对于每组数据，第 1 行一个正整数 m，表示这种物品共 m 家店有货。接下来 m 行，每行两个浮点数s，v，以空格分开，分别表示每家店的店铺评分和这种物品的价格。

数据范围：$1 \leq n \leq 20,\ 1 \leq m \leq 50,\ 4.0 \leq s \leq 5.0,\ 50.0 \leq v \leq 2000.0$ 。

输出

共 $n+1$ 组输出，每两组输出之间加一行空行，浮点数全部保留一位小数。

前 n 组，对应 n 种物品，每组数据 m 行，每行两个浮点数 s，v，用一个空格分开，表示排序后的商品对应的店铺评分和价格。

第 $n+1$ 组，共 n 行，每行两个浮点数 s，v，用一个空格分开，表示按输入顺序加购的物品的店铺评分和价格。

输入样例

```
2
4
4.5 123.6
4.9 144.8
4.8 115.2
4.9 133.4
5
4.2 1845.4
4.1 1863.7
4.2 1838.3
4.9 1879.9
4.9 1875.2
```

输出样例

```
4.9 133.4
4.9 144.8
4.8 115.2
4.5 123.6

4.9 1875.2
4.9 1879.9
4.2 1838.3
4.2 1845.4
4.1 1863.7

4.9 133.4
4.9 1875.2
```

Hint

请勿使用$qsort$。

如果你使用冒泡排序，可以考虑自己重新定义一下小于关系，比如在什么时候我们认为店铺a < 店铺b。

解题思路

这道题是一个简单的排序问题，但是要注意需要根据两个条件进行排序，由于输出要求，需要建立一个单独的数组来储存需要购买的商品。

有很多种排序方法都可以满足这次这种时间、内存都要求不大的题目，但建议大家自己手写，不要用现成的库函数。对于时间、内存有限制的题目，届时大家应选择相应的排序方法。对于某些不稳定的排序，大家可以有意识地减少使用，或者打乱数据防止被卡。

下附各种排序的时间、空间复杂度及稳定性表格，有兴趣的同学可以稍作了解。

类别	排序方法	时间复杂度			空间复杂度	稳定性
		平均情况	最好情况	最坏情况	辅助存储	
插入排序	直接插入	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
	Shell排序	$O(n^{1.3})$	$O(n)$	$O(n^2)$	$O(1)$	不稳定
选择排序	直接选择	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
	堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(1)$	不稳定
交换排序	冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
	快速排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$	$O(n\log_2 n)$	不稳定
归并排序		$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$	稳定
基数排序		$O(d(r+n))$	$O(d(n+rd))$	$O(d(r+n))$	$O(rd+n)$	稳定

图片来自CSDN

AC代码

在这里只给出冒泡排序和选择排序两种最基础方法的代码。

冒泡排序代码

```
#include <stdio.h>
double s[50], v[50];
/*为了简化交换函数，避免使用指针，故将这两个数组设置为全局变量*/

swap(int a, int b)
/*部分功能封装成函数，可提高代码可读性*/
/*某些复用性很高的代码，写出来可以简化代码的书写*/
/*大家要养成写函数的好习惯啊*/
{
        double t;
        t = s[a];
        s[a] = s[b];
        s[b] = t;
        t = v[a];
        v[a] = v[b];
        v[b] = t;
}

int main()
{
        int n, m, i, j, mark = 0;
        double bought[20][2];

        scanf("%d", &n);
        while(n--)
        {
                scanf("%d", &m);
                for(i = 0; i < m; i++) scanf("%lf%lf", &s[i], &v[i]);
                for(i = 0; i < m - 1; i++)
                {
                        for(j = m - 2; j >= i; j--)
                        if(s[j] < s[j + 1] || (s[j] == s[j + 1] && v[j] >
v[j + 1]))
            /*同时判断两个条件，也可以分开判断，见下一代码*/
                        {
                swap(j, j + 1);
                        }
            printf("%.1f %.1f\n", s[i], v[i]);
                }
                printf("%.1f %.1f\n", s[i], v[i]);
                bought[mark][0] = s[0];
                bought[mark++][1] = v[0];
                puts("");//注意格式不要错，中间有一个换行
        }
        for(i = 0; i < mark; i++) printf("%.1f %.1f\n", bought[i][0],
```

```
bought[i][1]);
}
```

选择排序代码

这一代码是使用了选择排序的思路，但并不完全按照选择排序的方法进行。

```
#include <stdio.h>

int main()
{
        double s[50], v[50];
        int n, m, i, j, mark = 0, best, marking;
        double bought[20][2];

        scanf("%d", &n);
        while(n--)
        {
                scanf("%d", &m);
                for(i = 0; i < m; i++) scanf("%lf%lf", &s[i], &v[i]);
                marking = 0;
                for(j = 0; j < m; j++)
                {
                        best = 0;
                        for(i = 1; i < m; i++)
                        {
                                if(s[i] > s[best]) best = i;
                else if(s[i] == s[best])
                        if(v[i] < v[best]) best = i;
                /*
                这两行也可以写成
                else if((s[i] == s[best] && v[i] < v[best]) best = i;
                */
                        }
                        printf("%.1f %.1f\n", s[best], v[best]);
            /*这里选择直接输出，以避免交换*/
                        if(marking == 0)
                        {
                                bought[mark][0] = s[best];
                                bought[mark++][1] = v[best];
                        }
                        marking = 1;
                        s[best] = 0;//也可以选择其他的可行方法将用过的数去除
            //在正常的选择排序中，应将最大or小的数与低一个or最后一个数字交换
                }

                puts("");
        }
        for(i = 0; i < mark; i++) printf("%.1f %.1f\n", bought[i][0],
bought[i][1]);
}
```

关于关键词

这个题目包含两个关键词，大家可以用上述的判断方法判断两个关键词，也可以通过 p = s * 10000 - v类似的办法将两个关键词合并成一个进行判断。

G. 糖糖背单词（简单版）

题目描述

糖糖每天都在坚持背单词，每天她都会随机地从词库里选出一些单词来背。选择的方式有三种，第1种是选择相同**首**字母的单词；第2种是选择**最后**一个字母相同的单词；第3种是选择**包含**特定字母的单词。

但是光背单词是不够的，糖糖每天还会定期地阅读一些英文文章来实际检验她背单词的成果。今天，糖糖想知道她看的文章里面有多少是她今天刚刚背过的单词，你能帮帮她吗？

单词的定义：只包括英文字母'a'-'z'和'A'-'Z'的字符串我们称之为单词。例如"aBc"是1个单词,而"a-bc"是2个单词"a"和"bc","abc123"不是单词(但"abc"是)。

输入

输入共两行。

第一行为一个小写字母c和一个整数t，代表糖糖今天选择的字母和选择方式，以一个空格隔开。($t \in \{1,2,3\}$)

第二行为一个字符串，保证字符串中只含有可见字符，且长度不超过1024。

Note：糖糖选择单词的时候，是不考虑大小写的。

输出

请输出字符串中所有糖糖今天背的单词出现的次数和。

样例输入1

```
a 1
hello,aworld!how are you? today is my birthdAy,,,,Ae12ae12ae12you ok?
```

样例输出1

```
5
```

样例输入2

```
m 2
emmmm,well,it's hard to saym cause im trapped in dmandim mainly by my mum
and sam and hammer.
```

样例输出2

```
6
```

样例输入3

```
o 3
ooop,ohw couldo'oit be such a pity stuff as nOthing can always stOp.
```

样例输出3

```
6
```

解题思路

这里单词中断符为非单词字符，因此需要单独处理。处理字符串时，我们可以存储当前为读到单词/空格的状态state，是否已经记录此单词背过的状态f，根据这两个状态和当前、上一个字符的值判断新读入的字符会不会使所求数值增加。

AC代码

```
#include<stdio.h>
#include<ctype.h>
int match(char a, char c){
        return c == a || c == toupper(a);
}
int main(){
        char a, c, pre = 0;
        int i, op;
        int state = 0, cnt = 0, f = 0;
        scanf("%c%d", &a, &op);
        while((c = getchar()) != EOF){
                if(islower(c) || isupper(c)) {
                        if(state == 0){
                            //第一个字符匹配
                                if(match(a, c) && op == 1) cnt++;
                                //读入新单词，更新记录
                                state = 1, f = 0;
                        }
                }
                else if(state != 0 ){
                    //最后一个字符（上一个字符）匹配
                        if(op == 2 && match(a, pre)) cnt++;
                        state = 0;
                }
                //单词中任意字符匹配且未被记录
```

```
                if((c == a || c == toupper(a)) && f == 0 && op == 3) f =
1, cnt++;
                pre = c;
        }
        printf("%d", cnt);
}
```

H. HugeGun学姐的方程

题目描述

HugeGun学姐又上完了西点课，给她的男神带了她做的巨难吃的三明治。

男神非常生气，扔给了她一个方程：$\frac{e^{(-\sqrt{\frac{x}{10}})}}{ln\frac{x}{10}}=y$

睿智的HugeGun学姐算不出来它的解，只好像你寻求帮助。

保证答案在区间$(10,30)$内且等式左边函数在此区间上单调递减

输入

一行一个小数$y(0.25\leq y\leq 5)$

输出

一行一个五位小数x_0表示$\frac{e^{-\sqrt{\frac{x}{10}}}}{ln\frac{x}{10}}=y$的解。

输入样例

```
1.2500000000
```

输出样例

```
12.92586
```

解题思路

这道题给出了确定的上下界的范围，因此可以直接使用二分。函数比较复杂，需要使用math.h库进行辅助。注意是y而非0，所以应当跟y做比较。

关于**math.h**库中的几个函数的使用： **1.exp（double a)**用来计算自然底数**e**的**a**次方 **2.log(double a)**用以计算以**e**为底数的对数，**log10(double a)**则是以**10**为底数的对数，其他对数请使用换底公式 **3.sqrt(double a)**用以开根运算 **4.pow(double x,double y**）求**x**的**y**次方

AC代码

```
#include <stdio.h>
#include <math.h>
#define eps 1e-8
```

```
double f(double x)
{
        return exp(-1 * sqrt(x / 10)) / log(x / 10);
}

int main()
{
        double y;
        double left = 10, right = 30, mid;
        double leftAns, rightAns, midAns = 0;

        scanf("%lf", &y);
        while(midAns - y > eps || y - midAns > eps)
        {
                leftAns = f(left);
                rightAns = f(right);
                mid = (left + right) / 2;
                midAns = f(mid);
                if(midAns > y)
                {
                        left = mid;
                }
                else if(midAns < y)
                {
                        right = mid;
                }
                else
                {
                        printf("%.5f", mid);
                        return 0;
                }
        }
        printf("%.5f", mid);
}
```

I. 数三角形

题目描述

$Erin$有n根长度互不相同的杆子，第i根杆的长度为L_i。

$Erin$想知道用这些杆子能组成多少种不同的三角形。

输入

第一行为一个正整数n, $3 \leq n \leq 2.5 * 10^3$

第二行为n个正整数L_i,$1 \leq L_i \leq 10^9$，用一个空格隔开，保证这个n个数互不相同。

对于60%的数据，保证$3 \leq n \leq 10^2$

输出

一行，为不同的三角形的个数。

输入样例1

```
4
3 4 2 1
```

输出样例2

```
1
```

输入样例2

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

输出样例2

```
3
```

样例2解释

[2, 3, 4], [2, 4, 5], [3, 4, 5]

解题思路

一个简单的思路是利用三重循环枚举所有的情况，但是本题 $n \leq 2.5 * 10^3$ ，对大多数在线C/C++评测系统来讲，一秒钟约能进行 10^7 至 10^8 次计算，因此本题如果使用三重循环的做法，面对取到上确界的 n 的情况时一定会超时。

我们可以从另一个角度考虑。假设数组是有序的，那么只需枚举两个值，然后通过二分查找尝试得到第三个值在数组中可行的下标范围，如果存在这个范围，那么直接累计上这个范围的长度即可。这相当于用二分查找代替了一层循环。大家都知道 $\lim\limits_{n\to\infty} \frac{log(n)}{n} = 0$，因此这大幅提高了程序效率。

有兴趣或者有基础的同学可以从时间复杂度的角度来分析。当然，你也可以尝试直接从数学分析的无穷大的阶的角度进行分析。k层循环耗时相当于幂函数 n^k，每一次二分查找的耗时则是对数函数 $log(n)$。冒泡/选择排序因为有两层满打满算的循环，所以耗时相当于二次函数 n^2，快速排序则是 $nlog(n)$ （有兴趣的同学可以自行学习）

再回到本题的分析。对于第一种求解思路（嵌套3层满满的循环），总耗时相当于幂函数 n^3。

对于第二种求解思路，总耗时是排序的耗时 + 进行二分查找的次数 (n^2) 和每次二分查找的耗时 ($log(n)$) 的乘积。如果使用冒泡/选择排序，总耗时则是 $n^2 + n^2log(n)$。

大家都知道 $\lim\limits_{n\to\infty} \frac{n^2+n^2log(n)}{n^3} = 0$，因此第二种求解思路确实大幅提高了程序效率。并且当 n 取到上确界时，$n^2 + n^2log(n)$ 的数量级也是可以接受的。因此本题可使用第二种解法通过。

AC代码

```
#include <stdio.h>

#define MN 5005

int binary_search(int a[], int left, int right, int lower, int upper)
{
    int mid = (left + right) / 2;
    int key = lower + a[mid] - upper;
    if (right - left <= 1)
        return right;
    else if (key > 0)
        return binary_search(a, left, mid, lower, upper);
    else
        return binary_search(a, mid, right, lower, upper);
}

void bubble_sort(int a[], int n)
{
    int i, j, temp;
    for (i = 0; i < n - 1; i++)
    {
        for (j = 0; j < n - 1 - i; j++)
        {
            if (a[j] > a[j + 1])
            {
```

```
                temp = a[j + 1];
                a[j + 1] = a[j];
                a[j] = temp;
            }
        }
    }
}

int main()
{
    static int a[MN];
    int i, j, n;
    long long ans = 0;

    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

    bubble_sort(a, n);

    for (i = 0; i < n; i++)
        for (j = n - 1; j > i; j--)
            ans += j - binary_search(a, i, j, a[i], a[j]);

    printf("%lld", ans);
    return 0;
}
```

J. 自动AC机

题目描述

hyy买到了一台仅售998的自动AC机感到十分开心（传言这台机器可以自动打出能AC的代码），仔细一看却发现是一件假冒伪劣产品。

这台机器有26个小写字母按键和代表退格($Backspace$)的B键以及代表打印($Print$)的P键。

- 键入一个小写字母，这个字母将被加在凹槽的末端。
- 键入B，凹槽末端的最尾一个字母将被除去。保证在按下B时凹槽不为空。
- 键入P，凹槽中的全部字母将被打印出，不过它们不会在凹槽中消失。

在报警之前，她决定利用这台老式打字机问你一些问题。请你帮她统计第x次打印的字符串在第y次打印的字符串中出现了几次。

输入

第1行一个字符串$s(1 \leq |s| \leq 10^2$，仅包含小写字母与B、P，$|s|$代表字符串长度$)$，表示键入按键的顺序。

第2行一个正整数$q(1 \leq q \leq 10^2)$，表示询问组数。

接下来q行，每行两个整数$x,y(1 \leq x,y \leq s$中$'P'$的出现次数$)$，以一个空格分隔，表示询问第x次打印的字符串在第y次打印的字符串中出现了几次。例：aa在$aaab$中出现了2次，从1开始计数的话这两次出现分别在$aaab$的位置1和位置2。

输出

q行，每行一个整数ans_i表示第i个询问的答案。

输入样例

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

输出样例

```
2
1
0
```

样例解释

第1次打印a，第2次打印aa，第3次打印ab

Hint

可以根据需要使用一些内置函数。

解题思路

题目中出现的打字机需要我们建立缓冲区来模拟。每次按P将缓冲区中的全部字符复制出来。注意删除字符需要将字符置0以标记缓冲区字符串结尾。

每次询问在第二个字符串中找第一个字符串的出现次数。注意每次找到以后需要更新下一次查找的起始位置。

AC代码

```
#include<stdio.h>
int main(){
        char buf[200] = {}, c;
        int pos = 0;
        char str[200][200] = {};
        int i = 1, q;
        while((c = getchar()) != '\n'){
                if(c == '\r') break;
                if(c == 'B') buf[--pos] = '\0';
                else if(c == 'P') strcpy(str[i++], buf);
                else buf[pos++] = c;
        }
        scanf("%d", &q);
        while(q--){
                int x, y;
                scanf("%d%d",&x, &y);
                int cnt = 0;
                char *s = str[y];
                //每次查找更新s
                while((s = strstr(s, str[x])) != NULL){
                        s++;
                        cnt++;
                        if(*s == '\0') break;
                }
                printf("%d\n", cnt);
        }
}
```

K. 连接运算符

题目描述

在$Verilog$语言中，存在着一种神奇的运算符，它的名字叫做连接运算符（$concatenation\ operator$），其运算结果为将大括号里所有数依次**转化为二进制**，并**依次连接**起来得到的新的二进制数。

例如，`{1'b1,1'b0}=2'b10`，`{3'b101,1'b1}=4'b1011`。

形如`x'yz`的串代表一个整数，其中`x`为整数，代表数据位宽（转换为二进制后需要占有的位数）；`y`为一个表示进制的字符，**仅可能是`'b','o','d','h'`中的某一个**；`z`为`y`所代表的进制表示的整数。这种表示数字的方式称作“位宽-进制表示法”。

当`y`为'b'`(binary)`时，`z`为二进制数；当`y`为'o'`(octal)`时，`z`为八进制数；`y`为'd'`(decimal)`时，`z`为十进制数；当`y`为'h'`(hexadecimal)` 时，`z`为十六进制数，**`z`中的字母**可能为大写或小写。本题**仅考虑这四种情况**。

给定一个仅包含一个连接运算符构成的表达式`{a,b,c,...}`，请你求出它的运算结果，并**用“位宽-进制表示法”输出**。

输入

多组数据。

对于每组数据，一行一个字符串，保证第一个字符为'{'，最后一个字符为'}'，中间为若干个以**一个逗号和若干空格**隔开的、用“位宽-进制表示法”表示**且不含空格**的数。

输入的最后一行是整数-1，表示输入结束。

保证输入的每个数的位宽都是**合法**的，且每个数转化为10进制后都在$[0,2^{63}-1]$范围内。字符串长度不超过1000。输入数据可能存在前导零。

输出

对每组数据输出一行，一个用“位宽-进制表示法”表示的数（其中y要求为'b'），代表运算结果。

要求结果去掉前导零。

输入样例

```
{1'b0, 1'b1}
{3'b101   ,5'd9}
{5'o22,16'h5e59,4'b1}
{1'b0,1'b0,1'b0,1'b1,1'b1,1'b1 , 1'b0,1'b0,1'b0}
{1'b0}
{2'b0,2'b0}
-1
```

输出样例

```
2'b1
8'b10101001
25'b1001001011110010110010001
9'b111000
1'b0
4'b0
```

解题思路

本题需要根据题意，模拟整个过程，主要考察了大家对字符串的处理。

字符串处理题很考察大家对细节、边角情况的把握能力。一方面大家需要考虑周全、严谨，避免遇到没有考虑到的边角情况致使程序出错，另一方面，大家最好在读完题后、写代码前再仔细斟酌，尽量简化解题逻辑，以从根本上减少情况/分支的种数，降低犯错的概率。

AC代码

```
#include <stdio.h>
#include <string.h>

#define ML 1000007

int a[ML];

int main()
{
        char ch;
        int i, tot = 0, width, val, radix, has_leading_zero = 0, flag = 0;
        long long sum = 0;

        while (scanf("%c", &ch) != EOF)
        {
                if (ch == '-')
                        break;
                if (ch == '\r' || ch == '\n' || ch == '{' || ch == ' ' ||
ch == '\'')
                {
                        if (ch == '\'')
                                flag = 1;
                        continue;
                }

                if (flag == 0)
                        width = 10 * width + ch-'0';

                if (ch == '}' || ch == ',')
                {
```

```
                        for (i = tot + width; i > tot; i--, sum>>=1)
                                a[i] = sum & 1;
                        tot += width;

                        sum = flag = width = 0;
                        if (ch == '}')
                        {
                                printf("%d\'b", tot);
                                has_leading_zero = 0;
                                for (i = 1; i < tot; i++)
                                {
                                        if (a[i] == 0 && has_leading_zero
== 0)
                                                continue;
                                        has_leading_zero = 1;
                                        printf("%d", a[i]);
                                }

                                printf("%d\n", a[tot]);
                                tot = 0;
                        }
                }

                if (flag == 1)
                {
                        switch (ch)
                        {
                                case 'b':
                                        radix = 2;
                                        break;
                                case 'o':
                                        radix = 8;
                                        break;
                                case 'd':
                                        radix = 10;
                                        break;
                                case 'h':
                                        radix = 16;
                                        break;
                        }
                        flag = 2;
                        continue;
                }

                if (flag == 2)
                {
                        if (ch >= '0' && ch <= '9')
                                val = ch - '0';
                        else if (ch >= 'a' && ch <= 'z')
                                val = ch - 'a' + 10;
                        else if (ch >= 'A' && ch <= 'Z')
                                val = ch - 'A' + 10;

                        sum = radix * sum + val;
```

```
                }
            }

            return 0;
}
```