抓码机试每日一题合集

题目一：害死人不偿命的(3n+1)猜想

> 卡拉兹(Callatz)猜想：

对任何一个自然数n，如果它是偶数，那么把它砍掉一半；如果它是奇数，那么把(3n+1)砍掉一半。这样一直反复砍下去，最后一定在某一步得到n=1。卡拉兹在1950年的世界数学家大会上公布了这个猜想，传说当时耶鲁大学师生齐动员，拼命想证明这个貌似很傻很天真的命题，结果闹得学生们无心学业，一心只证(3n+1)，以至于有人说这是一个阴谋，卡拉兹是在蓄意延缓美国数学界教学与科研的进展……

我们今天的题目不是证明卡拉兹猜想，而是对给定的任一不超过1000的正整数n，简单地数一下，需要多少步（砍几下）才能得到n=1？

> 输入格式：

每个测试输入包含1个测试用例，即给出自然数n的值。

> 输出格式：

输出从n计算到1需要的步数。

> 输入样例：

```
3
```

> 输出样例：

```
5
```

> 分析：

count从0开始统计需要的步数，(n % 2 != 0) 表示n为奇数，当n为奇数，就令n = 3*n+1；之后将其砍掉一半，步数count+1，直到n == 1为止，最后输出count

```
#include<cstdio>
int main()
{
  int n, step = 0;
  scanf("%d", &n);    //输入题目所给的n
```

```
  while(n != 1)      //循环判断n是否为1
  {
    if(n % 2 == 0)    //如果是偶数
       n = n/2;
    else
       n = (3*n + 1) /2;  //如果是奇数
    step++;           //计数器+1
  }
  printf("%d\n", step);
}
```

题目二：写出这个数

读入一个自然数n，计算其各位数字之和，用汉语拼音写出和的每一位数字。

> **输入格式：**

每个测试输入包含1个测试用例，即给出自然数n的值。这里保证n小于10100。

> **输出格式：**

在一行内输出n的各位数字之和的每一位，拼音数字间有1 空格，但一行中最后一个拼音数字后没有空格。

> **输入样例：**

```
123456789098765432112345678 9
```

> **输出样例：**

```
yi san wu
```

> **分析：**

用string接收输入，string的每一位数字累加到sum里面，再将sum转化为string类型的num，对num的每一位输出对应中文拼音～

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
  string s;
  cin >> s;
  int sum = 0;
```

```
  string str[10] = {"ling", "yi", "er", "san", "si", "wu", "liu", "qi",
  "ba", "jiu"};
  for (int i = 0; i < s.length(); i++)
     sum += (s[i] - '0');
  string num = to_string(sum);
  for (int i = 0; i < num.length(); i++)
  {
     if (i != 0) cout << " ";
     cout << str[num[i] - '0'];
   }
 return 0;
}
```

题目三：我要通过

“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行给出一个自然数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
```

分析：

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；

那么正确的有这些：

PAT
APATA
AAPATAA
AAAPATAAA
.....

中间一个A左右加上等量的A（不加也行）都是正确的。

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。

拿上面的那几个正确的举例子，那么正确的有这些：

PAT —— 对于 aPbTc 来说ac是空，b是A。所以 PAAT 是正确的。同理PAAAAAT中间加多少个A都是正确哒~

APATA —— 对于aPbTc来说，abc都是A。所以 APAATAA 是正确的。再类推一下，那么 APAAATAAA 是正确的。

AAPATAA —— 对于aPbTc来说，a和c是AA，b是A。所以AAPAATAAAA是正确的，再类推一下，

AAPAAATAAAAAA 是正确的。

所以说规律就是，可以在P和T中间加A并且在T后面加A，要求必须是，中间加上一个A，末尾就得加上几倍的(P前面A的那个字符串)。换句话说就是，中间的A的个数如果是3，那么末尾的A的个数就得是开头A的个数的3倍。很巧，当中间A为一个的时候，末尾和开头A的个数必须相等正好是第二条的要求

所以一句话总结字符串的要求：只能有一个P一个T，中间末尾和开头可以随便插入A。但是必须满足开头的A的个数 * 中间的A的个数 = 结尾的A的个数，而且P和T中间不能没有A

```
#include <iostream>
#include <map>
using namespace std;
int main() {
    int n, p = 0, t = 0;
    string s;
    cin >> n;
    for(int i = 0; i < n; i++) {
        cin >> s;
        map<char, int> m;
        for(int j = 0; j < s.size(); j++) {
            m[s[j]]++;
            if (s[j] == 'P') p = j;
            if (s[j] == 'T') t = j;
         }
         if (m['P'] == 1 && m['A'] != 0 && m['T'] == 1 && m.size() == 3 &
& t-p
!= 1 && p * (t-p-1) == s.length()-t-1)
              printf("YES\n");
          else
              printf("NO\n");
```

```
        }
        return 0;
}
```

题目四：继续(3n+1)猜想

卡拉兹(Callatz)猜想已经在1001中给出了描述。在这个题目里，情况稍微有些复杂。

当我们验证卡拉兹猜想的时候，为了避免重复计算，可以记录下递推过程中遇到的每一个数。例如对n=3进行验证的时候，我们需要计算3、5、8、4、2、1，则当我们对n=5、8、4、2进行验证的时候，就可以直接判定卡拉兹猜想的真伪，而不需要重复计算，因为这4个数已经在验证3的时候遇到过了，我们称5、8、4、2是被3“覆盖”的数。我们称一个数列中的某个数n为“关键数”，如果n不能被数列中的其他数字所覆盖。

现在给定一系列待验证的数字，我们只需要验证其中的几个关键数，就可以不必再重复验证余下的数字。你的任务就是找出这些关键数字，并按从大到小的顺序输出它们。

> **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K(<100)，第2行给出K个互不相同的待验证的正整数n(1<n<=100)的值，数字间用空格隔开。

> **输出格式：**

每个测试用例的输出占一行，按从大到小的顺序输出关键数字。数字间用1个空格隔开，但一行中最后一个数字后没有空格。

> **输入样例：**

```
6
3 5 6 7 8 11
```

> **输出样例：**

```
7 6
```

> **分析：**

对每一个输入的数字n进行验证，把验证过的数字对应的arr标记为1，然后对这些输入的数字从大到小排序，输出所有arr=0的数字即为关键数字。

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int arr[10000];
bool cmp(int a, int b) {return a > b;}
int main() {
    int k, n, flag = 0;
    cin >> k;
    vector<int> v(k);
    for (int i = 0; i < k; i++) {
        cin >> n;
        v[i] = n;
        while (n != 1) {
            if (n % 2 != 0) n = 3 * n + 1;
            n = n / 2
            if (arr[n] == 1) break;
            arr[n] = 1;
         }
     }
     sort(v.begin(), v.end(), cmp);
     for (int i = 0; i < v.size(); i++) {
         if (arr[v[i]] == 0) {
               if (flag == 1) cout << " ";
               cout << v[i];
               flag = 1;
         }
     }
     return 0;
}
```

题目五：素数对猜想

让我们定义 dn 为：dn = pn+1 – pn，其中 pi 是第i个素数。显然有 d1=1 且对于n>1 有 dn 是偶数。 “素数对猜想”认为“存在无穷多对相邻且差为2的素数”。

现给定任意正整数N (< 105)，请计算不超过N的满足猜想的素数对的个数。

> **输入格式：**

每个测试输入包含1个测试用例，给出正整数N。

输出格式：

每个测试用例的输出占一行，不超过N的满足猜想的素数对的个数。

输入样例：

```
20
```

输出样例：

```
4
```

分析：

判断素数的函数isprime这样写：对于数字a，i从2到根号a，如果a能够被其中一个i整除，说明i不是素数，return false，否则说明a是素数return true；对于输入数据N，for循环中的i从5到N依次判断i-2和i是否是素数，如果都是素数，则统计个数的cnt++，最后输出cnt即可。

```
#include <iostream>
using namespace std;
bool isprime(int a) {
     for (int i = 2; i * i <= a; i++)
         if (a % i == 0) return false;
      return true;
}
int main() {
    int N, cnt = 0;
    cin >> N;
    for (int i = 5; i <= N; i++)
        if (isprime(i-2) && isprime(i)) cnt++;
    cout << cnt;
    return 0;
}
```

题目六：数组元素循环右移问题

一个数组A中存有N（N>0）个整数，在不允许使用另外数组的前提下，将每个整数循环向右移M（M>=0）个位置，即将A中的数据由（A0 A1……AN-1）变换为（AN-M …… AN-1 A0 A1……AN-M-1）（最后M个数循环移至最前面的M个位置）。如果需要考虑程序移动数据的次数尽量少，要如何设计移动的方法？

> **输入格式：**

每个输入包含一个测试用例，第1行输入
N(1<=N<=100)、M（M>=0）；

第2行输入N个整数，之间用空格分隔。

> **输出格式：**

在一行中输出循环右移M位以后的整数序列，之间用空格分隔，序列结尾不能有多余空格。

> **输入样例：**

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

> **输出样例：**

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

> **分析：**

数组长度为n，要想把数组循环右移m位，只需要先将整个数组a倒置，再将数组前m位倒置，最后将数组后n-m位倒置即可完成循环右移m位。

reverse函数可以实现将一个数组或者vector中元素倒置，这个函数在algorithm头文件中。

（如果m大于n，那么循环右移m位相当于循环右移m%n位，因为那些n倍数位的移动是多余的，所以在使用m之前，先将m = m%n）

```

#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main() {
    int n, m;
    cin >> n >> m;
    vector<int> a(n);
    for (int i = 0; i < n; i++)
        cin >> a[i];
    m %= n;
    if (m != 0) {
```

```
        reverse(begin(a), begin(a) + n);
        reverse(begin(a), begin(a) + m);
        reverse(begin(a) + m, begin(a) + n);
    }
    for (int i = 0; i < n - 1; i++)
        cout << a[i] << " ";
    cout << a[n - 1];
    return 0;
}
```

题目七：一元多项式求导

设计函数求一一元多项式的导数。(注:xn(n为整数)的一一阶导数为n*xn-1。)

输入格式：

以指数递降方方式输入入多项式非非零项系数和指数(绝对值均为不不超过1000的整数)。数字间以空格分 隔。

输出格式：

以与输入入相同的格式输出导数多项式非非零项的系数和指数。数字间以空格分隔，但结尾不不能有多余空格。注意“零多项式”的指数和系数都是0，但是表示为“0 0”。

输入样例：

```
3 4 -5 2 6 1 -2 0
```

输出样例：

```
12 3 -10 1 6 0
```

分析：

1. flag用用来判断是否已经有过输出。

2. 当b!=0时，因为给出的是所有非非零项系数，所以必定会有输出，先判断flag是否为1，如果为1表示已 经有过输出，那么在前面面要先输出一一个空格。

3. 输出 a * b 和 b – 1，然后将flag标记为1表示已经有过输出 4.最后判断当没有输出并且b==0的时候， 输出“0 0”

```

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

```
int main() {
    int a, b, flag = 0;
    while (cin >> a >> b) {
        if (b != 0) {
            if (flag == 1) cout << " ";
            cout << a * b << " " << b - 1;
            flag = 1;
    }
}
    if (flag == 0) cout << "0 0";
return 0; }
```

题目八：数字分类

给定一系列正整数，请按要求对数字进行分类，并输出以下5个数字：

A1 = 能被5整除的数字中所有偶数的和；

A2 = 将被5除后余1的数字按给出顺序进行交错求和，即计算n1-n2+n3-n4...；

A3 = 被5除后余2的数字的个数；

A4 = 被5除后余3的数字的平均数，精确到小数点后1位；

A5 = 被5除后余4的数字中最大数字。

> **输入格式：**

每个输入包含1个测试用例。每个测试用例先给出一个不超过1000的正整数N，随后给出N个不超过1000的待分类的正整数。数字间以空格分隔。

> **输出格式：**

对给定的N个正整数，按题目要求计算A1~A5并在一行中顺序输出。
数字间以空格分隔，但行末不得有多余空格。
若其中某一类数字不存在，则在相应位置输出“N”。

> **输入样例1：**

```
13 1 2 3 4 5 6 7 8 9 10 20 16 18
```

> **输出样例1：**

```
30 11 2 9.7 9
```

> **输入样例2：**

```
8 1 2 4 5 6 7 9 16
```

输出样例2：

```
N 11 2 N 9
```

分析：

将每一个数字按照取余后的结果i保存在v[i]数组中，然后对v[i]中的每一个元素按照不同i分类计算出A1 A2...A5～

```

#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, num, A1 = 0, A2 = 0, A5 = 0;
    double A4 = 0.0;
    cin >> n;
    vector<int> v[5];
    for (int i = 0; i < n; i++) {
        cin >> num;
        v[num%5].push_back(num);
    }
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < v[i].size(); j++) {
            if (i == 0 && v[i][j] % 2 == 0) A1 += v[i][j];
            if (i == 1 && j % 2 == 0) A2 += v[i][j];
            if (i == 1 && j % 2 == 1) A2 -= v[i][j];
            if (i == 3) A4 += v[i][j];
            if (i == 4 && v[i][j] > A5) A5 = v[i][j];
        }
    }
    for (int i = 0; i < 5; i++) {
        if (i != 0) printf(" ");
        if (i == 0 && A1 == 0 || i != 0 && v[i].size() == 0) {
            printf("N"); continue;
        }
        if (i == 0) printf("%d", A1);
        if (i == 1) printf("%d", A2);
        if (i == 2) printf("%d", v[2].size());
        if (i == 3) printf("%.1f", A4 / v[3].size());
        if (i == 4) printf("%d", A5);
     }
     return 0;
}
```

题目九：福尔摩斯的约会

大侦探福尔摩斯接到一张奇怪的字条：

> 我们约会吧！
>
> 3485djDkxh4hhGE
>
> 2984akDfkkkkggEdsb
>
> s&hgsfdk d&Hyscvnm

大侦探很快就明白了，字条上奇怪的乱码实际上就是约会的时间“星期四 14:04”，因为前面两字符串中第1对相同的大写英文字母（大小写有区分）是第4个字母‘D’，代表星期四；第2对相同的字符是‘E’，那是第5个英文字母，代表一天里的第14个钟头（于是一天的0点到23点由数字0到9、以及大写字母A到N表示）；后面两字符串第1对相同的英文字母‘s’出现在第4个位置（从0开始计数）上，代表第4分钟。现给定两对字符串，请帮助福尔摩斯解码得到约会的时间。

输入格式：

输入在4行中分别给出4个非空、不包含空格、且长度不超过60的字符串。

输出格式：

在一行中输出约会的时间，格式为“DAY HH:MM”，其中“DAY”是某星期的3字符缩写,即MON表示星期一，TUE表示星期二，WED表示星期三，THU表示星期四，FRI表示星期五，SAT表示星期六，SUN表示星期日。题目输入保证每个测试存在唯一解。

输入样例：

```
3485djDkxh4hhGE
2984akDfkkkkggEdsb
s&hgsfdk
d&Hyscvnm
```

输出样例：

```
30 11 2 9.7 9
```

输入样例2：

```
THU 14:04
```

分析：

按照题目所给的方法找到相等的字符后判断即可，如果输出的时间不足2位数要在前面添0，即用%02d输出。

```
#include <iostream>
#include <cctype>
using namespace std;
int main() {
    string a, b, c, d;
    cin >> a >> b >> c >> d;
    char t[2];
    int pos, i = 0, j = 0;
    while(i < a.length() && i < b.length()) {
        if (a[i] == b[i] && (a[i] >= 'A' && a[i] <= 'G')) {
            t[0] = a[i];
            break;
        }
        i++;
    }
    i = i + 1;
    while (i < a.length() && i < b.length()) {
        if (a[i] == b[i] && ((a[i] >= 'A' && a[i] <= 'N') || isdigit(a
[i]))) {
            t[1] = a[i];
            break;
        }
        i++
    }
    while (j < c.length() && j < d.length()) {
        if (c[j] == d[j] && isalpha(c[j])) {
            pos = j;
            break;
        }
        j++;
    }
    string week[7] = {"MON ", "TUE ", "WED ", "THU ", "FRI ", "SAT ", "SUN
"};
    int m = isdigit(t[1]) ? t[1] - '0' : t[1] - 'A' + 10;
    cout << week[t[0]-'A'];
    printf("%02d:%02d", m, pos);
```

```
    return 0;
}
```

题目十：部分A+B

正整数A的“DA（为1位整数）部分”定义为由A中所有DA组成的新整数PA。例如：给定A = 3862767，DA= 6，则A的“6部分”PA是66，因为A中有2个6。

现给定A、DA、B、DB，请编写程序计算PA + PB。

输入格式：

输入在一行中依次给出A、DA、B、DB，中间以空格分隔，其中0 < A, B < 10^10。

输出格式：

在一行中输出PA + PB的值。

输入样例1：

```
3862767 6 13530293 3
```

输出样例2：

```
399
```

输入样例2：

```
3862767 1 13530293 8
```

输出样例2：

```
0
```

分析：

将A和B保存在string a 和 b中，将DA和DB保存在da和db中，因为A为字符串，所以对于它的每一位a[i]，当da == (a[i]- ‘0’)时候表示da和a[i]相等，将相等的次数保存在cnta中，当cnta不为0时，说明A中有位数等于da，先令pa = da，然后根据cnta的次数，将cnta个da转化为pa的值（转化方法为for循环从1到cnta-1，每次pa

乘以10再加上da），B同理，相等的次数保存在cntb中，求出pb的值，最后输出pa+pb的值。

```
#include <iostream>
using namespace std;
int main() {
    string a, b;
    int da, db, cnta = 0, cntb = 0, pa = 0, pb = 0;
    cin >> a >> da >> b >> db;
    for (int i = 0; i < a.length(); i++)
        if (da == (a[i] - '0')) cnta++;
    for (int i = 0; i < b.length(); i++)
        if (db == (b[i] - '0')) cntb++;
    if (cnta != 0) pa = da;
    if (cntb != 0) pb = db;
    for (int i = 1; i < cnta; i++)
        pa = 10 * pa + da;
    for (int i = 1; i < cntb; i++)
        pb = 10 * pb + db;
    cout << pa + pb;
    return 0;
}
```

题目十一：锤子剪刀布

大家应该都会玩“锤子剪刀布”的游戏吧，现给出两人的交锋记录，请统计双方的胜、平、负次数，并且给出双方分别出什么手势的胜算最大。

> **输入格式：**

输入第1行给出正整数N（<=105），即双方交锋的次数。随后N行，每行给出一次交锋的信息，即甲、乙双方同时给出的的手势。C代表“锤子”、J代表“剪刀”、B代表“布”，第1个字母代表甲方，第2个代表乙方，中间有1个空格。

> **输出格式：**

输出第1、2行分别给出甲、乙的胜、平、负次数，数字间以1个空格分隔。第3行给出两个字母，分别代表甲、乙获胜次数最多的手势，中间有1个空格。如果解不唯一，则输出按字母序最小的解。

> **输入样例：**

```
10
C J
J B
C B
B B
B C
C C
C B
J B
B C
J J
```

> **输出样例：**

```
5 3 2
2 3 5
B B
```

> **分析：**

jiawin、yiwin分别表示甲乙赢的次数，s和t分别表示每一次甲乙给出的手势，maxjia和maxyi分别表示甲乙获胜次数最多的手势所对应的下标（012分别表示BCJ），枚举每一次甲乙手势的胜负结果并累加到jiawin和yiwin中，最后根据题目要求输出结果。

```

#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    int jiawin = 0, yiwin = 0;
    int jia[3] = {0}, yi[3] = {0};
    for (int i = 0; i < n; i++) {
        char s, t;
        cin >> s >> t;
        if (s == 'B' && t == 'C')
            jiawin++;
            jia[0]++;
        } else if (s == 'B' && t == 'J') {
            yiwin++;
            yi[2]++;
        } else if (s == 'C' && t == 'B') {
            yiwin++;
            yi[0]++;
        } else if (s == 'C' && t == 'J') {
            jiawin++;
            jia[1]++;
```

```
        } else if (s == 'J' && t == 'B') {
            jiawin++;
            jia[2]++;
        } else if (s == 'J' && t == 'C') {
            yiwin++;
            yi[1]++;
        }
    }
    cout << jiawin << " " << n - jiawin - yiwin << " " << yiwin << endl <<
yiwin << " " << n - jiawin - yiwin << " " << jiawin << endl;
    int maxjia = jia[0] >= jia[1] ? 0 : 1;
    maxjia = jia[maxjia] >= jia[2] ? maxjia : 2;
    int maxyi = yi[0] >= yi[1] ? 0 : 1;
    maxyi = yi[maxyi] >= yi[2] ? maxyi : 2;
    char str[4] = {"BCJ"};
    cout << str[maxjia] << " " << str[maxyi];
    return 0;
}
```

题目十二：个位数统计

给定一个k位整数N = dk-1*10k-1 + ... + d1*101 + d0 (0<=di<=9, i=0,...,k-1, dk-1>0)，请编写程序统计每种不同的个位数字出现的次数。例如：

给定N = 100311，则有2个0，3个1，和1个3。

> **输入格式：**

每个输入包含1个测试用例，即一个不超过1000位的正整数N。

> **输出格式：**

对N中每一种不同的个位数字，以D:M的格式在一行中输出该位数字D及其在N中出现的次数M。
要求按D的升序输出。

> **输入样例：**

```
100311
```

> **输出样例：**

```
0:2
1:3
```

```
3:1
```

> **分析：**

因为N为不超过1000位的正整数，所以用字符串s来接收N，遍历字符串中的每个字符，将每个数字出现的次数保存在数组a中，a[i]表示数字i出现的次数，最后将数组a的下标0-9中所有a[i]不为0的输出即可

```
#include <iostream>
using namespace std;
int main() {
    string s;
    cin >> s;
    int a[10] = {0};
    for (int i = 0; i < s.length(); i++)
        a[s[i] - '0']++;
    for (int i = 0; i < 10; i++) {
        if (a[i] != 0)
            printf("%d:%d\n", i, a[i]);
    }
    return 0;
}
```

题目十三：组个最小数

给定数字0-9各若干个。你可以以任意顺序排列这些数字，但必须全部使用。目标是使得最后得到的数尽可能小（注意0不能做首位）。例如：给定两个0，两个1，三个5，一个8，我们得到的最小的数就是10015558。

现给定数字，请编写程序输出能够组成的最小的数。

> **输入格式：**

每个输入包含1个测试用例。每个测试用例在一行中给出10个非负整数，顺序表示我们拥有数字0、数字1、……数字9的个数。整数间用一个空格分隔。10个数字的总个数不超过50，且至少拥有1个非0的数字。

> **输出格式：**

在一行中输出能够组成的最小的数。

> **输入样例：**

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

> **输出样例：**

```
10015558
```

> **分析：**

将数字0、数字1、……数字9的个数分别保存在数组a[i]中，因为0不能做首位，所以首先将i从1到9输出第一个a[i]不为0的数字 i ，并将这个 i 保存在变量 t 中，接着输出a[0]个0，最后输出a[t]-1个t（因为一个 t 已经被放在首位输出过一次了～），最后 i 从 t+1 到 9 输出 a[i] 个 i 。

```

#include <iostream>
using namespace std;
int main() {
    int a[10], t;
    for (int i = 0; i < 10; i++)
        cin >> a[i];
    for (int i = 1; i < 10; i++) {
        if (a[i] != 0) {
            cout << i;
            t = i;
            break;
        }
    }
    for (int i = 0; i < a[0]; i++) cout << 0;
    for (int i = 0; i < a[t] - 1; i++) cout << t;
    for (int i = t + 1; i < 10; i++)
        for (int k = 0; k < a[i]; k++)
            cout << i;
    return 0;
}
```

题目十四：反转链表

给定一一个常数K以及一一个单链表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，即最后不不到K个元素不不反转。

输入格式：

每个输入入包含1个测试用用例例。每个测试用用例例第1行行行给出第1个结点的地址、 结点总个数正整数N(<= 10^5)、以及正整数K(<=N)， 即要求反转的子子链结点的个数。结点的地址是5位非非负整数，NULL地址用用-1表示。接下来有N行行行，每行行行格式为:
Address Data Next 其中Address是结点地址，Data是该结点保存的整数数据，Next是下一一结点的地址。

输出格式：

对每个测试用用例例，顺序输出反转后的链表，其上每个结点占一一行行行，格式与输入入相同。

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
```

分析：

输入入样例例正确连接顺序应该是:

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

还应该考虑输入入样例例中有不不在链表中的结点的情况。所以用用个sum计数。而且，algorithm头文文件里里里面面有reverse函数可以直接调用用。

```
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    int first, k, n, temp;
    cin >> first >> n >> k;
    int data[100005], next[100005], list[100005];
    for (int i = 0; i < n; i++) {
        cin >> temp;
        cin >> data[temp] >> next[temp];
    }
    int sum = 0;//不不一一定所有的输入入的结点都是有用用的，加个计数器器
    while (first != -1) {
        list[sum++] = first;
        first = next[first];
    }
    for (int i = 0; i < (sum - sum % k); i += k)
        reverse(begin(list) + i, begin(list) + i + k);
    for (int i = 0; i < sum - 1; i++)
        printf("%05d %d %05d\n", list[i], data[list[i]], list[i + 1]);
    printf("%05d %d -1", list[sum - 1], data[list[sum - 1]]);
    return 0;
 }
```

题目十五：打印沙漏

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
```

分析：

每个沙漏都是从最中间一行行向上下分别扩展一行，每次扩展行都要比之前一层多2个符号，最中间一行只有 1 个符号，假设扩展的层数为 i，则扩展出去的上边需要的所有符号个数为3 + 5 + 7 + ...+ (2i+1) = (3 + 2i + 1) * i / 2 = i * (i + 2)，扩展出去的下边与上边同样多所以乘以2，加上最重要那一行1个符号，所以 总共需要2 * i * (i + 2) + 1个符号，所以i从0到N，找满足(2 * i * (i + 2) + 1)> N的最小的 i，因为符号不能超过N，所以只能扩展出去 i-1 行，用变量row表示从最中间一行需要扩展出去的行数，row= i – 1，接下来开始输出，上面的每一行，对于扩展出去的第 i 层需要输出row – i 个空格，接着输出i *2 + 1个符号c和换行符；对于最中间一行，需要输出row – 1个空格、符号c和换行符；对于下面的每一行，对于扩展出去的第i层，需要输出row-i个空格，接着输出i * 2 + 1个符号c和换行符，因为用掉的符号数为2 * row * (row + 2) + 1，所以最后输出剩下没用掉的符号数为N – (2 * row * (row + 2) + 1)。

```

#include <iostream>
using namespace std;
int main() {
    int N, row = 0;
    char c;
    cin >> N >> c;
```

```
    for (int i = 0; i < N; i++) {
        if ((2 * i * (i + 2) + 1) > N) {
            row = i - 1;
            break;
        }
    }
    for (int i = row; i >= 1; i--) {
        for (int k = row - i; k >= 1; k--) cout << " ";
        for (int j = i * 2 + 1; j >= 1; j--) cout << c;
        cout << endl;
    }
    for (int i = 0; i < row; i++) cout << " ";
    cout << c << endl;
    for (int i = 1; i <= row; i++) {
        for (int k = row - i; k >= 1; k--) cout << " ";
        for (int j = i * 2 + 1; j >= 1; j--) cout << c;
        cout << endl;
    }
    cout << (N - (2 * row * (row + 2) + 1));
    return 0;
}
```

题目十六：旧键盘

旧键盘上坏了几个键，于是在敲一段文字的时候，对应的字符就不会出现。现在给出应该输入的一段文字、以及实际被输入的文字，请你列出肯定坏掉的那些键。

> 输入格式：

输入在2行中分别给出应该输入的文字、以及实际被输入的文字。每段文字是不超过80个字符的串，由字母A-Z（包括大、小写）、数字0-9、以及下划线“_”（代表空格）组成。题目保证2个字符串均非空。

> 输出格式：

按照发现顺序，在一行中输出坏掉的键。其中英文字母只输出大写，每个坏键只输出一次。题目保证至少有1个坏键。

> 输入样例：

```
7_This_is_a_test
_hs_s_a_es
```

> 输出样例：

```
7TI
```

> 分析：

用string的find函数～遍历字符串s1，当当前字符s1[i]不在s2中，它的大写也不在ans中时，将当前字符的大写放入ans中，最后输出ans字符串即可。

ps：感谢github上的@xiaorong61给我发的pull request中strchr()函数带来的灵感～让我想到了曾经用过的string的find函数。

```
#include <iostream>
#include <cctype>
using namespace std;
int main() {
    string s1, s2, ans;
    cin >> s1 >> s2;
    for (int i = 0; i < s1.length(); i++)
        if (s2.find(s1[i]) == string::npos && ans.find(toupper(s1[i])) ==
string::npos)
            ans += toupper(s1[i]);
    cout << ans;
    return 0;
}
```

题目十七：查验身份证

一个合法的身份证号码由17位地区、日期编号和顺序编号加1位校验码组成。校验码的计算规则如下：

首先对前17位数字加权求和，权重分配为：{7，9，10，5，8，4，2，1，6，3，7，9，10，5，8，4，2}；

然后将计算的和对11取模得到值Z；最后按照以下关系对应Z值与校验码M的值：

Z：0 1 2 3 4 5 6 7 8 9 10

M：1 0 X 9 8 7 6 5 4 3 2

现在给定一些身份证号码，请你验证校验码的有效性，并输出有问题的号码。

> 输入格式：

输入第一行给出正整数N（<= 100）是输入的身份证号码的个数。随后N行，每行给出1个18位身份证号码。

输出格式：

按照输入的顺序每行输出1个有问题的身份证号码。这里并不检验前17位是否合理，只检查前17位是否全为数字且最后1位校验码计算准确。如果所有号码都正常，则输出“All passed”。

输入样例1：

```
4
320124198808240056
12010X198901011234
110108196711301866
37070419881216001X
```

输出样例1：

```
12010X198901011234
110108196711301866
37070419881216001X
```

输入样例2：

```
2
320124198808240056
110108196711301862
```

输出样例2：

```
All passed
```

分析：

isTrue函数判断身份证号是否正常，如果不正常返回false，判断每一个给出的身份证号，如果不正常，就输出这个身份证号，并且置flag=1表示有不正常的号码，如果所有的号码都是正常，即flag依旧等于0，则输出All passed。在isTrue函数中，先判断前17位是否是数字，如果不是，直接returnfalse，如果是，就将当前字符转化为数字并与a[i]相乘，累加在sum中，对于第18位，如果是X要转化为10。比较b[sum%11]和第18位是否相等，如果相等就返回true，不相等就返回false。

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

```
int a[17] = {7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2};
int b[11] = {1, 0, 10, 9, 8, 7, 6, 5, 4, 3, 2};
string s;
bool isTrue() {
    int sum = 0;
    for (int i = 0; i < 17; i++) {
        if (s[i] < '0' || s[i] > '9') return false;
        sum += (s[i] - '0') * a[i];
    }
    int temp = (s[17] == 'X') ? 10 : (s[17] - '0');
    return b[sum%11] == temp;
}
int main() {
    int n, flag = 0;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> s;
        if (!isTrue()) {
            cout << s << endl;
            flag = 1
        }
    }
    if (flag == 0) cout << "All passed";
    return 0;
}
```

题目十八：有理数四则运算

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
```

分析：

func(m, n)的作用是对m/n的分数进行化简，gcd(t1, t2)的作用是计算t1和t2的最大公约数。在func函数中，先看m和n里面是否有0（即m*n是否等于0），如果分母n=0，输出Inf，如果分子m=0，输出”0”。flag表示m和n是否异号，flag=true表示后面要添加负号”(-“和括号”)”，再将m和n都转为abs(m)和abs(n)，即取他们的正数部分方便计算。

x = m/n为m和n的可提取的整数部分，先根据flag的结果判断是否要在前面追加”(-“，然后根据x是否等于0判断要不要输出这个整数位，接着根据m%n是否等于0的结果判断后面还有没有小分数，如果m能被n整除，表示没有后面的小分数，那么就根据flag的结果判断要不要加”)”，然后直接return。

如果有整数位，且后面有小分数，则要先输出一个空格，接着处理剩下的小分数，先把m分子减去已经提取出的整数部分，然后求m和n的最大公约数t，让m和n都除以t进行化简～最后输出“m/n”，如果flag==true还要在末尾输出”)”

注意：判断m和n是否异号，千万不要写成判断m*n是否小于0，因为m*n的结果可能超过了long long int的长度，导致溢出大于0，如果这样写的话会有一个测试点无法通

过。

```
#include <iostream>
#include <cmath>
using namespace std;
long long a, b, c, d;
long long gcd(long long t1, long long t2) {
    return t2 == 0 ? t1 : gcd(t2, t1 % t2);
}
void func(long long m, long long n) {
    if (m * n == 0) {
        printf("%s", n == 0 ? "Inf" : "0");
        return ;
    }
    bool flag = ((m < 0 && n > 0) || (m > 0 && n < 0));
    m = abs(m); n = abs(n);
    long long x = m / n;
    printf("%s", flag ? "(-" : "");
    if (x != 0) printf("%lld", x);
    if (m % n == 0) {
        if(flag) printf(")");
        return ;
    }
    if (x != 0) printf(" ");
    m = m - x * n;
    long long t = gcd(m, n);
    m = m / t; n = n / t;
    printf("%lld/%lld%s", m, n, flag ? ")" : "");
}
int main() {
    scanf("%lld/%lld %lld/%lld", &a, &b, &c, &d);
    func(a, b);
    printf(" + ");
    func(c, d);
    printf(" = ");
    func(a * d + b * c, b * d);
    printf("\n");
    func(a, b);
    printf(" - ");
    func(c, d);
    printf(" = ");
    func(a * d - b * c, b * d);
    printf("\n");
    func(a, b);
    printf(" * ");
    func(c, d);
    printf(" = ");
    func(a * c, b * d);
```

```
    printf("\n");
    func(a, b);
    printf(" / ");
    func(c, d);
    printf(" = ");
    func(a * d, b * c);
    return 0;
}
```

题目十九：统计同成绩学生

本题要求读入N名学生的成绩，将获得某一给定分数的学生人数输出。

输入格式：

输入在第1行给出不超过 10^5 的正整数N，即学生总人数。随后1行给出N名学生的百分制整数成绩，中间以空格分隔。最后1行给出要查询的分数个数K（不超过N的正整数），随后是K个分数，中间以空格分隔。

输出格式：

在一行中按查询顺序给出得分等于指定分数的学生人数，中间以空格分隔，但行末不得有多余空格。

输入样例：

```
10
60 75 90 55 75 99 82 90 75 50
3 75 90 88
```

输出样例：

```
3 2 0
```

分析：

用b数组保存每个分数对应的学生人数，在输入的时候，对于每一个成绩temp，b[temp]++表示将数组b中对应分数的人数+1。

对于m个查询，每一次都输出需要查询的temp所对应的人数b[temp]，注意i不等于0的时候要在输出人数之前输出一个空格。

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, m, temp;
    scanf("%d", &n);
    vector<int> b(101);
    for (int i = 0; i < n; i++) {
        scanf("%d", &temp);
        b[temp]++;
    }
    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d", &temp);
        if (i != 0) printf(" ");
        printf("%d", b[temp]);
    }
    return 0;
}
```

题目二十：跟奥巴马一起编程

美国总统奥巴马不仅呼吁所有人都学习编程，甚至以身作则编写代码，成为美国历史上首位编写计算机代码的总统。

2014年底，为庆祝“计算机科学教育周”正式启动，奥巴马编写了很简单的计算机代码：在屏幕上画一个正方形。
现在你也跟他一起画吧！

> **输入格式：**

输入在一行中给出正方形边长N（3<=N<=20）和组成正方形边的某种字符C，间隔一个空格。

> **输出格式：**

输出由给定字符C画出的正方形。但是注意到行间距比列间距大，所以为了让结果看上去更像正方形，我们输出的行数实际上是列数的50%（四舍五入取整）。

> **输入样例：**

```
10 a
```

> **输出样例：**

```
aaaaaaaaaa
a a
a a
a a
aaaaaaaaaa
```

> 分析：

为了让结果看上去更像正方形，输出的行数实际上是列数的50%（四舍五入取整），列数是N，则行数t = N / 2 + N % 2，表示偶数等于除以2，奇数要除以2加1的意思，这样才能满足四舍五入。

首先输出第一行，N个c，然后输出中间t-2行，最左边和最右边为一个c，中间为N-2个空格，最后输出最后一行，N个c。

```
#include <iostream>
using namespace std;
int main() {
    int N;
    char c;
    cin >> N >> c;
    int t = N / 2 + N % 2;
    for (int i = 0; i < N; i++)
        cout << c;
    cout << endl;
    for (int i = 0; i < t - 2; i++) {
        cout << c;
        for (int k = 0; k < N - 2; k++)
            cout << " ";
        cout << c << endl;
    }
    for (int i = 0; i < N; i++)
        cout << c;
    return 0;
}
```

题目二十一：有几个PAT

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105，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
```

> **分析：**

要想知道构成多少个PAT，那么遍历字符串后对于每一A，它前面的P的个数和它后面的T的个数的乘积就是能构成的PAT的个数。然后把对于每一个A的结果相加即可。

只需要先遍历字符串数一数有多少个T，然后每遇到一个T，countt–;每遇到一个P，countp++;然后一遇到字母A就countt * countp。把这个结果累加到result中，最后输出结果就好了，别忘了要对10000000007进行取余。

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s;
    cin >> s;
    int len = s.length(), result = 0, countp = 0, countt = 0;
    for (int i = 0; i < len; i++) {
        if (s[i] == 'T')
            countt++;
    }
    for (int i = 0; i < len; i++) {
        if (s[i] == 'P') countp++;
        if (s[i] == 'T') countt--;
```

```
        if (s[i] == 'A') result = (result + (countp * countt) % 1000000007) %
1000000007;
    }
    cout << result;
    return 0;
}
```

题目二十二：火星数字

火星人是以13进制计数的：

地球人的0被火星人称为tret。
地球人数字1到12的火星文分别为：jan, feb, mar, apr, may, jun, jly, aug, sep, oct, nov, dec。火星人将进位以后的12个高位数字分别称为：tam, hel, maa, huh, tou, kes, hei, elo, syy, lok, mer, jou。例如地球人的数字“29”翻译成火星文就是“hel mar”；而火星文
“elo nov”对应地球数字“115”。为了方便交流，请你编写程序实现地球和火星数字之间的互译。

输入格式：

输入第一行给出一个正整数N（<100），随后N行，每行给出一个[0, 169)区间内的数字 —— 或者是地球文，或者是火星文。

输出格式：

对应输入的每一行，在一行中输出翻译后的另一种语言的数字。

输入样例：

```
4
29
5
elo nov
tam
```

输出样例：

```
hel mar
may
115
13
```

分析：

因为给出的可能是数字（地球文）也有可能是字母（火星文），所以用字符串s保存每一次的输入，因为如果是火星文则会出现空格，所以用getline接收一行的输入。计算string s的长度len，判断s[0]是否是数字，如果是数字，表示是地球文，则需要转为火星文，执行func1()；如果不是数字，则说明是火星文，需要转为地球文，执行func2()；

func1(int t)中，传入的值是string转int后的结果stoi(s)，因为数字最大不超过168，所以最多只会输出两位火星文，如果t / 13不等于0，说明有高位，所以输出b[t/13]；如果输出了高位（t/13不等于0）并且t% 13不等于0，说明有高位且有低位，所以此时输出空格；如果t % 13不等于0，说明有低位，此时输出a[t % 13]；注意，还有个数字0没有考虑，因为数字0取余13等于0，但是要特别输出tret，所以在func1的最后一句判断中加一句t == 0，并将a[0]位赋值成tret即可解决0的问题。

func2()中，t1和t2一开始都赋值0，s1和s2用来分离火星文单词，因为火星文字符串只可能一个单词或者两个单词，而且一个单词不会超过4，所以先将一个单词的赋值给s1，即s1 = s.substr(0, 3)；如果len> 4，就将剩下的一个单词赋值给s2，即s2 = s.substr(4, 3)；然后下标j从1到12遍历a和b两个数组，如果a数组中有和s1或者s2相等的，说明低位等于j，则将j赋值给t2；如果b数组中有和s1相等的（b数组不会和s2相等，因为如果有两个单词，s2只可能是低位），说明高位有值，将j赋值给t1，最后输出t1 *13 + t2即可。

```
#include <iostream>
#include <string>
using namespace std;
string a[13] = {"tret", "jan", "feb", "mar", "apr", "may", "jun", "jly",
 "aug",
"sep", "oct", "nov", "dec"};
string b[13] = {"####", "tam", "hel", "maa", "huh", "tou", "kes", "hei",
 "elo",
"syy", "lok", "mer", "jou"};
string s;
int len;
void func1(int t) {
    if (t / 13) cout << b[t / 13];
    if ((t / 13) && (t % 13)) cout << " ";
    if (t % 13 || t == 0) cout << a[t % 13];
}
void func2() {
    int t1 = 0, t2 = 0;
    string s1 = s.substr(0, 3), s2;
    if (len > 4) s2 = s.substr(4, 3);
    for (int j = 1; j <= 12; j++) {
        if (s1 == a[j] || s2 == a[j]) t2 = j;
        if (s1 == b[j]) t1 = j;
    }
```

```
    cout << t1 * 13 + t2;
}
int main() {
    int n;
    cin >> n;
    getchar();
    for (int i = 0; i < n; i++) {
        getline(cin, s);
        len = s.length();
        if (s[0] >= '0' && s[0] <= '9')
            func1(stoi(s));
        else
            func2();
        cout << endl;
    }
    return 0;
}
```

题目二十三：划 拳

划拳是古老中国酒文化的一个有趣的组成部分。酒桌上两人划拳的方法为：每人口中喊出一个数字，同时用手比划出一个数字。如果谁比划出的数字正好等于两人喊出的数字之和，谁就赢了，输家罚一杯酒。两人同赢或两人同输则继续下一轮，直到唯一的赢家出现。

下面给出甲、乙两人的划拳记录，请你统计他们最后分别喝了多少杯酒。

> **输入格式：**

输入第一行先给出一个正整数N（<=100），随后N行，每行给出一轮划拳的记录，格式为：
甲喊 甲划 乙喊 乙划
其中“喊”是喊出的数字，“划”是划出的数字，均为不超过100的正整数（两只手一起划）。

> **输出格式：**

在一行中先后输出甲、乙两人喝酒的杯数，其间以一个空格分隔。

> **输入样例：**

```
5
8 10 9 12
5 10 5 10
3 8 5 12
```

```
12 18 1 13
4 16 12 15
```

> **输出样例：**

```
1 2
```

> **分析：**

jia和yi分别代表甲乙两人喝酒的杯数，sum表示甲喊和乙喊之和，如果sum == 甲划，并且sum!= 乙划，表示乙输了，乙喝酒杯数yi++，反之jia++，最后输出jia和yi的值。

```
#include <iostream>
using namespace std;
int main() {
    int n, jia = 0, yi = 0, jiahan, jiahua, yihan, yihua, sum;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> jiahan >> jiahua >> yihan >> yihua;
        sum = jiahan + yihan;
        if (sum == jiahua && sum != yihua)
            yi++;
        if (sum != jiahua && sum == yihua)
            jia++;
    }
    cout << jia << " " << yi;
    return 0;
}
```

题目二十四：数字加密

本题要求实现一种数字加密方法。首先固定一个加密用正整数A，对任一正整数B，将其每1位数字与A的对应位置上的数字进行以下运算：对奇数位，对应位的数字相加后对13取余——这里用J代表10、Q代表11、K代表12；对偶数位，用B的数字减去A的数字，若结果为负数，则再加10。这里令个位为第1位。

> **输入格式：**

输入在一行中依次给出A和B，均为不超过100位的正整数，其间以空格分隔。

> **输出格式：**

在一行中输出加密后的结果。

> **输入样例：**

```
1234567 368782971
```

> **输出样例：**

```
3695Q8118
```

> **分析：**

首先将a和b倒置，将字符串a和b中较短的那个末尾添加0直到两个字符串长度相等，然后从0开始依次处理每一位，如果当前位是奇数位（i % 2 == 0）则将a[i]的数字加上b[i]的数字再对13取余，结果添加在字符串c的末尾；如果是偶数位，计算b[i]和a[i]的差值，如果小于0就加10，然后将结果添加在字符串c的末尾，最后倒序输出字符串c。

```
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    string a, b, c;
    cin >> a >> b;
    int lena = a.length(), lenb = b.length();
    reverse(a.begin(), a.end());
    reverse(b.begin(), b.end());
    if (lena > lenb)
        b.append(lena - lenb, '0');
    else
        a.append(lenb - lena, '0');
    char str[14] = {"0123456789JQK"};
    for (int i = 0; i < a.length(); i++) {
        if (i % 2 == 0) {
            c += str[(a[i] - '0' + b[i] - '0') % 13];
        } else {
            int temp = b[i] - a[i];
            if (temp < 0) temp = temp + 10;
            c += str[temp];
    }
    }
    for (int i = c.length() - 1; i >= 0; i--)
        cout << c[i];
    return 0;
}
```

题目二十五：组合数的和

给定N个非0的个位数字，用用其中任意2个数字都可以组合成1个2位的数字。要求所有可能组合出来的2位数字的和。例如给定2、5、8，则可以组合出:25、28、52、58、82、85，它们的和为330。

输入格式：

输入入在一一行中先给出N(1<N<10)，随后是N个不同的非0个位数字。数字间以空格分隔。

输出格式：

输出所有可能组合出来的2位数字的和。

输入样例：

```
3285
```

输出样例：

```
330
```

分析：

用用sum统计所有可能组合出来的两位数字之和，在sum累加的过程中，对于每一一个输入入的数字 temp，都能和其他N-1个数字组合出新的数字，temp能够放在个位也能够放在十十位，所以每个数字 temp都能在个位出现(N-1)次，十十位出现(N-1)次，在个位产生生的累加效果为temp * (N-1)，而而在十十位产生生 的累加效果为temp * (N-1) * 10，所以所有数字的累加结果sum即是所有可能组合出来的2位数字的和。

```
#include <cstdio>
int main() {
    int N, sum = 0, temp;
    scanf("%d", &N);
    for (int i = 0; i < N; i++) {
        scanf("%d", &temp);
        sum += temp * 10 * (N - 1) + temp * (N - 1);
    }
```

```
    printf("%d", sum);
   return 0;
}
```

题目二十六：多选题批改

批改多选题是比较麻烦的事情，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。注意判题时只有选择全部正确才能得到该题的分数。最后一行输出错得最多的题目的错误次数和编号（题目按照输入的顺序从1开始编号）。
如果有并列，则按编号递增顺序输出。数字间用空格分隔，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：

```
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) (2 b d) (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) (2 b c) (4 a b c d)
```

输出样例：

```
3
6
5
2 2 3 4
```

> 分析：

n个学生，m道题目。对于每一道题目，将题目的总分存储在total[i]数组里面，将题目的选项插入存储在right[i]（为集合类型）里。

wrongCnt[i]存储第i道题错误的人数，对于n个学生，每一个学生的答案插入一个集合st里面，比较st与right[i]是否相等，如果相等说明该生答对了，他的score +=total[i]（加上当前题目的总分），如果该生答错了，wrongCnt[i]++，表示第i道题新增一个错误的人。输出每一个学生的得分；遍历wrongCnt数组，求wrongCnt的最大值maxWrongCnt。

如果maxWrongCnt== 0说明没有一个人做错题目，则输出“Too simple”，否则输出maxWrongCnt的值，和wrongCnt数组中wrongCnt[i] == maxWrongCnt的那些题号。

注意：scanf中的%d和%c之间一定要有分隔符的主动scanf输入，否则可能接收成空格或者空值。

```
#include <cstdio>
#include <vector>
#include <set>
using namespace std;
int main() {
    int n, m, temp, k;
    scanf("%d%d", &n, &m);
    vector<set<char>> right(m);
    vector<int> total(m), wrongCnt(m);
    for(int i = 0; i < m; i++) {
        scanf("%d%d%d", &total[i], &temp, &k);
        for(int j = 0; j < k; j++) {
            char c;
            scanf(" %c", &c);
            right[i].insert(c);
        }
    }
    for(int i = 0; i < n; i++) {
        int score = 0;
        scanf("\n");
        for(int j = 0; j < m; j++) {
            if(j != 0) scanf(" ");
            scanf("(%d", &k);
            set<char> st;
```

```
            char c;
            for(int l = 0; l < k; l++) {
                scanf(" %c", &c);
                st.insert(c);
            }
            scanf(")");
            if(st == right[j]) {
                score += total[j];
            } else {
                wrongCnt[j]++;
            }
        }
        printf("%d\n", score);
    }
```

题目二十七：爱丁顿数

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（<=N）。

输入格式：

输入第一行给出一个正整数N（<=105），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
```

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。

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
int a[1000000];
bool cmp1(int a, int b) {
    return a > b;
}
int main() {
    int n;
    scanf("%d", &n);
    for(int i = 1; i <= n; i++) scanf("%d", &a[i]);
    sort(a+1, a+n+1, cmp1);
    int ans = 0, p = 1;
    while(ans <= n && a[p] > p) {
        ans++;
        p++;
    }
    printf("%d", ans);
    return 0;
}
```

题目二十八：最简分数

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 N1/M1 和 N2/M2，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
```

> **分析：**

使用辗转相除法gcd计算a和b的最大公约数，因为要列出n1/m1和n2/m2之间的最简分数，但是n1/m1不一定小于n2/m2，所以如果n1 * m2 > n2 * m1，说明n1/m1比n2/m2大，则调换n1和n2、m1和m2的位置～假设所求的分数分母为k、分子num，先令num=1，当n1 * k >= m1 * num时，num不断++，直到num符合n1/m1 < num/k为止～然后在n1/m1和n2/m2之间找符合条件的num的值，用gcd(num, k)是否等于1判断num和k是否有最大公约数，如果等于1表示没有最大公约数，就输出num/k，然后num不断++直到退出循环

```
#include <iostream>
using namespace std;
int gcd(int a, int b){
    return b == 0 ? a : gcd(b, a % b);
}
int main() {
    int n1, m1, n2, m2, k;
    scanf("%d/%d %d/%d %d", &n1, &m1, &n2, &m2, &k);
    if(n1 * m2 > n2 * m1) {
        swap(n1, n2);
        swap(m1, m2);
    }
    int num = 1;
    bool flag = false;
    while(n1 * k >= m1 * num) num++;
    while(n1 * k < m1 * num && m2 * num < n2 * k) {
        if(gcd(num, k) == 1) {
            printf("%s%d/%d", flag == true ? " " : "", num, k);
            flag = true;
        }
        num++;
    }
    return 0;
}
```

题目二十九：朋友数

如果两个整数各位数字的和是一样的，则被称为是“朋友数”，而那个公共的和就是它们的“朋友证号”。例如123和51就是朋友数，因为1+2+3 = 5+1 = 6，而6就是它们

的朋友证号。给定一些整数，要求你统计一下它们中有多少个不同的朋友证号。

注意：默认一个整数自己是自己的朋友。

输入格式：

输入第一行给出正整数N。随后一行给出N个正整数，数字间以空格分隔。题目保证所有数字小于104。

输出格式：

首先第一行输出给定数字中不同的朋友证号的个数；随后一行按递增顺序输出这些朋友证号，数字间隔一个空格，且行末不得有多余空格。

输入样例：

```
8
123 899 51 998 27 33 36 12
```

输出样例：

```
4
3 6 9 26
```

分析：

在接收输入数据的时候就把该数字的每一位相加，并把结果插入一个set集合中。因为set是有序的、不重复的，所以set的size值就是输出的个数，set中的每一个数字即所有答案的数字序列。

```
#include <iostream>
#include <set>
using namespace std;
int getFriendNum(int num) {
    int sum = 0;
    while(num != 0) {
        sum += num % 10;
        num /= 10;
    }
    return sum;
}
int main() {
    set<int> s;
    int n, num;
    scanf("%d", &n);
```

```
    for(int i = 0; i < n; i++) {
        scanf("%d", &num);
        s.insert(getFriendNum(num));
    }
    printf("%d\n", s.size());
    for(auto it = s.begin(); it != s.end(); it++) {
        if(it != s.begin()) printf(" ");
        printf("%d", *it);
    }
    return 0;
}
```

题目三十：万绿丛中一点红

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MxN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[0,224)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

> **分析：**

首先这个点必须是唯一的，所以用map标记如果不是唯一的点就不用考虑了，接着对于每个点，判断它的周围八个点与它的差值是否大于阈值，如果有一个点没有满足大于阈值就return false。最后记得输入的时候是列、行——m、n，输出的时候也是列、行坐标。

```
#include <iostream>
#include <vector>
#include <map>
using namespace std
int m, n, tol;
vector<vector<int>> v;
int dir[8][2] = {{-1, -1}, {-1, 0}, {-1, 1}, {0, 1}, {1, 1}, {1, 0}, {1, -1},
                 {0, -1}};
bool judge(int i, int j) {
    for (int k = 0; k < 8; k++) {
        int tx = i + dir[k][0];
        int ty = j + dir[k][1];
```

```
        if (tx >= 0 && tx < n && ty >= 0 && ty < m && v[i][j] - v[tx][ty]
          >= 0 - tol && v[i][j] - v[tx][ty] <= tol) return false;
                }
            return true;
        }
    int main() {
        int cnt = 0, x = 0, y = 0;
        scanf("%d%d%d", &m, &n, &tol);
        v.resize(n, vector<int>(m));
        map<int, int> mapp;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                scanf("%d", &v[i][j]);
                mapp[v[i][j]]++;
            }
        }
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
    if (mapp[v[i][j]] == 1 && judge(i, j) == true) {
                    cnt++;
                    x = i + 1;
                    y = j + 1;
                }
            }
        }
        if (cnt == 1)
    printf("(%d, %d): %d", y, x, v[x-1][y-1]);
        else if (cnt == 0)
            printf("Not Exist");
        else
            printf("Not Unique");
        return 0;
    }
```

题目三十一：结绳

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 <= N <= 104)；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104。

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
```

分析：

因为所有长度都要串在一起，每次都等于(旧的绳子长度+新的绳子长度)/2，所以越是早加入绳子长度中的段，越要对折的次数多，所以既然希望绳子长度是最长的，就必须让长的段对折次数尽可能的短。所以将所有段从小到大排序，然后从头到尾从小到大分别将每一段依次加入结绳的绳子中，最后得到的结果才会是最长的结果。

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main() {
    int n;
    scanf("%d", &n);
    vector<int> v(n);
    for (int i = 0; i < n; i++)
        scanf("%d", &v[i]);
    sort(v.begin(), v.end());
```

```
    int result = v[0];
    for (int i = 1; i < n; i++)
        result = (result + v[i]) / 2;
    printf("%d", result);
    return 0;
}
```

题目三十二：开学寄语

图像过滤是把图像中不重要的像素都染成背景色，使得重要部分被凸显出来。现给定一幅黑白图像，要求你将灰度值位于某指定区间内的所有像素颜色都用一种指定的颜色替换。

本题要求你写个程序帮助这所学校的老师检查所有学生的物品，以助其成大器。

输入格式：

输入第一行给出两个正整数N（<= 1000）和M（<= 6），分别是学生人数和需要被查缴的物品种类数。第二行给出M个需要被查缴的物品编号，其中编号为4位数字。随后N行，每行给出一位学生的姓名缩写（由1-4个大写英文字母组成）、个人物品数量K（0 <= K <= 10）、以及K个物品的编号。

输出格式：

顺次检查每个学生携带的物品，如果有需要被查缴的物品存在，则按以下格式输出该生的信息和其需要被查缴的物品的信息（注意行末不得有多余空格）：
姓名缩写: 物品编号1 物品编号2 ……
最后一行输出存在问题的学生的总人数和被查缴物品的总数。

输入样例：

```
4 2
2333 6666
CYLL 3 1234 2345 3456
U 4 9966 6666 8888 6666
GG 2 2333 7777
JJ 3 0012 6666 2333
```

输出样例：

```
U: 6666 6666
GG: 2333
JJ: 6666 2333
3 5
```

> **分析：**

bool类型的forbid存储禁止携带的物品，如果需要被查缴则赋值为true；flag变量表示当前学生是否已经输出过姓名，一开始flag=false，当前学生如果有需要被查缴的物品且还未输出过他的姓名，则输出name，并令flag=true；如果有需要被查缴的物品且已经输出过姓名，则输出该物品的编号，因为编号为4位数字，不满4位要在前面补0，所以用%04d输出，并将被查缴物品的总数fnum++，最后如果当前学生已经输出过姓名，则输出一个空行，并将学生的总人数snum++，最后输出snum和fnum的值。

```
#include <iostream>
#include <algorithm>
using namespace std;
bool forbid[10000];
int main() {
    int n, m, temp, k, snum = 0, fnum = 0;
    scanf("%d %d", &n, &m);
    for (int i = 0; i < m; i++) {
        scanf("%d", &temp);
        forbid[temp] = true;
    }
    for (int i = 0; i < n; i++) {
        char name[10];
        bool flag = false;
        scanf("%s %d", name, &k);
        for (int j = 0; j < k; j++) {
            scanf("%d", &temp);
            if (forbid[temp]) {
                if (!flag) {
                    printf("%s:", name);
                    flag = true;
                }
                printf(" %04d", temp);
                fnum++;
             }
        }
        if (flag) {
            printf("\n");
            snum++;
        }
    }
    printf("%d %d\n", snum, fnum);
    return 0;
}
```

题目三十三：宇宙无敌加法器

地球人习惯使用十进制数，并且默认一个数字的每一位都是十进制的。而在PAT星人开挂的世界里，每个数字的每一位都是不同进制的，这种神奇的数字称为“PAT数”。每个PAT星人都必须熟记各位数字的进制表，例如“……0527”就表示最低位是7进制数、第2位是2进制数、第3位是5进制数、第4位是10进制数，等等。每一位的进制d或者是0（表示十进制）、或者是[2，9]区间内的整数。理论上这个进制表应该包含无穷多位数字，但从实际应用出发，PAT星人通常只需要记住前20位就够用了，以后各位默认为10进制。

在这样的数字系统中，即使是简单的加法运算也变得不简单。例如对应进制表“0527”，该如何计算“6203+415”呢？

我们得首先计算最低位：3+5=8；因为最低位是7进制的，所以我们得到1和1个进位。第2位是：0+1+1（进位）=2；因为此位是2进制的，所以我们得到0和1个进位。第3位是：2+4+1（进位）=7；因为此位是5进制的，所以我们得到2和1个进位。第4位是：6+1（进位）=7；因为此位是10进制的，所以我们就得到7。最后我们得到：6203+415=7201。

输入格式：

输入首先在第一行给出一个N位的进制表（0 < N <=20），以回车结束。随后两行，每行给出一个不超过N位的正的PAT数。

输出格式：

在一行中输出两个PAT数之和。

输入样例：

```
30527
06203
415
```

输出样例：

```
7201
```

分析：

先将要相加的两个字符串S1和S2都扩展到和S等长，然后从后往前按照进制相加到ans中，注意进位carry，最后输出字符串ans，记得不要输出字符串ans前面的0。如果一次都没有输出，最后要输出一个0。

```
#include <iostream>
using namespace std;
int main() {
    string s, s1, s2, ans;
    int carry = 0, flag = 0;
    cin >> s >> s1 >> s2;
    ans = s;
    string ss1(s.length() - s1.length(), '0');
    s1 = ss1 + s1;
    string ss2(s.length() - s2.length(), '0');
    s2 = ss2 + s2;
    for(int i = s.length() - 1; i >= 0; i--) {
        int mod = s[i] == '0' ? 10 : (s[i] - '0');
        ans[i] = (s1[i] - '0' + s2[i] - '0' + carry) % mod + '0';
        carry = (s1[i] - '0' + s2[i] - '0' + carry) / mod;
    }
    if (carry != 0) ans = '1' + ans;
    for(int i = 0; i < ans.size(); i++) {
        if (ans[i] != '0' || flag == 1) {
            flag = 1;
            cout << ans[i];
        }
    }
    if (flag == 0) cout << 0;
    return 0;
}
```

题目三十四：互评成绩计算

在浙大的计算机专业课中，经常有互评分组报告这个环节。一个组上台介绍自己的工作，其他组在台下为其表现评分。最后这个组的互评成绩是这样计算的：所有其他组的评分中，去掉一个最高分和一个最低分，剩下的分数取平均分记为 G1；老师给这个组的评分记为 G2。该组得分为 (G1+G2)/2，最后结果四舍五入后保留整数分。本题就要求你写个程序帮助老师计算每个组的互评成绩。

> **输入格式：**

输入第一行给出两个正整数N（ > 3 ）和M，分别是分组数和满分，均不超过100。随后N行，每行给出该组得到的N个分数（均保证为整型范围内的整数），其中第1个是老师给出的评分，后面 N-1 个是其他组给的评分。合法的输入应该是[0, M]区间内的整数，若不在合法区间内，则该分数须被忽略。题目保证老师的评分都是合法的，并且每个组至少会有3个来自同学的合法评分。

输出格式：

为每个组输出其最终得分。每个得分占一行。

输入样例：

```
6 50
42 49 49 35 38 41
36 51 50 28 -1 30
40 36 41 33 47 49
30 250 -25 27 45 31
48 0 0 50 50 1234
43 41 36 29 42 29
```

输出样例：

```
42
33
41
31
37
39
```

分析：

maxn和minn分别保存最高分和最低分，将所有其他组评分中的有效分数累加到g1，最后减去minn和maxn并求平均分，最后求得最终得分。

```
#include <iostream>
using namespace std;
int main() {
    int n, m;
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        int g2, g1 = 0, cnt = -2, temp, maxn = -1, minn = m + 1;
        cin >> g2;
        for (int j = 0; j < n-1; j++) {
            cin >> temp;
            if (temp >= 0 && temp <= m) {
```

```
                if (temp > maxn) maxn = temp;
                if (temp < minn) minn = temp;
                g1 += temp;
                cnt++;
            }
        }
        cout << int((((g1 - minn - maxn) * 1.0 / cnt) + g2) / 2 + 0.5) <<
 endl;
    }
    return 0;
}
```

题目三十五：延迟的回文数

给定一个 k+1 位的正整数 N，写成 ak...a1a0 的形式，其中对所有 i 有 0 <= ai < 10 且 ak > 0。N 被称为一个回文数，当且仅当对所有 i 有 ai = ak-i。零也被定义为一个回文数。

非回文数也可以通过一系列操作变出回文数。首先将该数字逆转，再将逆转数与该数相加，如果和还不是一个回文数，就重复这个逆转再相加的操作，直到一个回文数出现。如果一个非回文数可以变出回文数，就称这个数为延迟的回文数。

给定任意一个正整数，本题要求你找到其变出的那个回文数。

输入格式：

输入在一行中给出一个不超过1000位的正整数。

输出格式：

对给定的整数，一行一行输出其变出回文数的过程。每行格式如下

A + B = C

其中A是原始的数字，B是A的逆转数，C是它们的和。A从输入的整数开始。重复操作直到C在10步以内变成回文数，这时在一行中输出“C is a palindromic number.”；或者如果10步都没能得到回文数，最后就在一行中输出“Not found in 10 iterations.”。

输入样例1：

```
97152
```

输出样例1：

```
97152 + 25179 = 122331
122331 + 133221 = 255552
255552 is a palindromic number.
```

输入样例2：

```
196
```

输出样例2：

```
196 + 691 = 887
887 + 788 = 1675
1675 + 5761 = 7436
7436 + 6347 = 13783
13783 + 38731 = 52514
52514 + 41525 = 94039
94039 + 93049 = 187088
187088 + 880781 = 1067869
1067869 + 9687601 = 10755470
10755470 + 07455701 = 18211171
Not found in 10 iterations.
```

分析：

1、将字符串倒置与原字符串比较看是否相等可知s是否为回文串

2、字符串s和它的倒置t相加，只需从头到尾相加然后再倒置（记得要处理最后一个进位carry，如果有进位要在末尾+‘1’）

3、倒置可采用algorithm头文件里面的函数reverse(s.begin(), s.end())直接对s进行倒置

```
#include <iostream>
#include <algorithm>
using namespace std;
string rev(string s) {
    reverse(s.begin(), s.end());
    return s;
}
string add(string s1, string s2) {
    string s = s1;
    int carry = 0;
    for (int i = s1.size() - 1; i >= 0; i--) {
        s[i] = (s1[i] - '0' + s2[i] - '0' + carry) % 10 + '0';
        carry = (s1[i] - '0' + s2[i] - '0' + carry) / 10;
    }
    if (carry > 0) s = "1" + s;
    return s;
}
int main() {
    string s, sum;
```

```
    int n = 10;
    cin >> s;
    if (s == rev(s)) {
        cout << s << " is a palindromic number.\n";
        return 0;
    }
    while (n--) {
        sum = add(s, rev(s));
        cout << s << " + " << rev(s) << " = " << sum << endl;
        if (sum == rev(sum)) {
            cout << sum << " is a palindromic number.\n";
            return 0;
        }
        s = sum;
    }
    cout << "Not found in 10 iterations.\n";
    return 0;
}
```

题目三十六：射击比赛

本题目给出的射击比赛的规则非常简单，谁打的弹洞距离靶心最近，谁就是冠军；谁差得最远，谁就是菜鸟。本题给出一系列弹洞的平面坐标(x,y)，请你编写程序找出冠军和菜鸟。我们假设靶心在原点 (0,0)。

> **输入格式：**

输入在第一行中给出一个正数 N(<= 10 000)。随后 N 行，每行按下列格式给出:
ID x y
其中 ID 是运动员的编号(由4位数字组成);x 和 y 是其打出的弹洞的平面坐标(x,y)，均为整数，且 0 <= |x|, |y| <= 100。题目保证每个运动员的编号不重复，且每人只打 1 枪。

> **输出格式：**

输出冠军和菜鸟的编号，中间空 1 格。题目保证他们是唯一的。

> **输入样例：**

```
3
0001 5 7
1020 -1 3
0233 0 -1
```

> **输出样例例:**

```
0233 0001
```

> **分析：**

1、注意n=1的情况，即冠军和菜鸟都是同一个人的情况(第二个测试点)

2、注意距离越大的越菜

```
#include <iostream>
using namespace std;
int main() {
    int n, id, x, y, maxid, maxdis = -1, minid, mindis = 99999;
    cin >> n;
    for (int i = 0; i < n; i++) {
     cin >> id >> x >> y;
        int dis = x * x + y * y;
        if (dis > maxdis) maxid = id;
        if (dis < mindis) minid = id;
        maxdis = max(maxdis, dis);
        mindis = min(mindis, dis);
    }
    printf("%04d %04d", minid, maxid);
    return 0;
```

题目三十七：成绩排序

> 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。

示例：

```
jack 70
peter 96
Tom 70
smith 67

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

```

从低到高
smith 67
jack 70
Tom 70
peter 96
```

> **输入描述：**

输入多行，先输入要排序的人的个数，然后输入排序方法0（降序）或者1（升序）再分别输入他们的名字和成绩，以一个空格隔开。

> **输出格式：**

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
```

> **分析：**

a 和 b的乘积转换成字符串，再将字符串反转，最后将反转过的字符串转换成数字。

```
/*成绩排序---清华大学*/
#include<iostream>
#include<cstdio>
#include<algorithm>
using namespace std;
const int maxn = 1000;
//定义学生信息的结构体
struct stu{
  char name[20];
  int score;
}buf[maxn];
```

```
//自定义排序规则
bool cmp1(stu a, stu b)
{
    if(a.score != b.score)
     return a.score < b.score;        //分数小的排在前面，升序排序
 }
 bool cmp2(stu a, stu b)
 {
     if(a.score != b.score)
       return a.score > b.score;       //分数高的排在前面， 降序排序
  }
int main()
{
  int n;
  int way;     //排序方式
  while(cin>>n)
  {
    cin>>way;        //输入排序方式
    for(int i = 0; i<n; i++)
    {
      cin>>buf[i].name>>buf[i].score;     //输入学生姓名和分数
     }
    if(way == 1)
    {
      //升序排序
      stable_sort(buf, buf + n, cmp1);
      for(int i = 0; i<n; i++)
      {
        cout<<buf[i].name<<" "<<buf[i].score<<endl;
       }
    }
    else
    {
      //降序排序
      stable_sort(buf, buf + n, cmp2);
      for(int i = 0; i<n; i++)
      {
        cout<<buf[i].name<<" "<<buf[i].score<<endl;
      }
     }
  }
  return 0;
 }
```

题目三十八：最大数和最小数

> 查找和排序

输入N个（N<=10000）数字，求出这N个数字中的最大值和最小值。每个数字的绝对值不大于1000000

> 输入描述：

输入包括多组测试用例，每组测试用例由一个整数N开头，接下去一行给出N个整数。

> 输出格式：

输出包括两个整数，为给定N个数中的最大值与最小值。

> 示例1

输入

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

输出

```
5 1
8 3
```

```
/*清华大学--求最大数最小数*/
#include<iostream>
#include<cstdio>
using namespace std;
const int maxn = 20000;
int main()
{
  int n;
  int a[maxn];
  int max = -1000000, min = 1000000;
  while(cin>>n)
  {
    for(int i = 0; i<n; i++)
    {
      scanf("%d",&a[i]);
    }
    for(int i = 0; i<n; i++)
    {
      if(a[i]>max)
```

```
      {
        max = a[i];
      }
      if(a[i]<min)
      {
        min = a[i];
      }
    }
    cout<<max<<" "<<min;
  //  cout<<min;
    return 0;

  }
}
```

题目三十九：今天的第几天？

> 题目描述

输入年、月、日，计算该天是本年的第几天。

> 输入描述：

包括三个整数年(1<=Y<=3000)、月(1<=M<=12)、日(1<=D<=31)。

> 输出格式：

输入可能有多组测试数据，对于每一组测试数据，输出一个整数，代表Input中的年、月、日对应本年的第几天。

> 示例1

输入

```
1990 9 20
2000 5 1
```

输出

```
263
122
```

```
/**今年的第几天*/
#include<cstdio>
```

```cpp
#include<iostream>
using namespace std;

const int n = 13;
int main()
{
  int run[13] = {0,31,29,31,30,31,30,31,31,30,31,30,31};
  int ping[13] = {0,31,28,31,30,31,30,31,31,30,31,30,31};

    int year ,month, day;
    while(cin>>year>>month>>day)
    {
        if(year%4==0&&year%100!= 0 || year%400 == 0)
        {
            for(int i = 0; i<month; i++)
            {
                day = day + run[i];
            }
        }
        else
        {
            for(int i = 0; i<month; i++)
            {
              day = day + ping[i];
            }
        }
        cout<<day<<endl;
    }
 }
```

题目四十：查找学生信息

> 题目描述

输入N个学生的信息，然后进行查询。

> 输入描述：

输入的第一行为N，即学生的个数(N<=1000)
接下来的N行包括N个学生的信息，信息格式如下：
01 李江 男 21
02 刘唐 男 23
03 张军 男 19
04 王娜 女 19

然后输入一个M(M<=10000),接下来会有M行，代表M次查询，每行输入一个学号，格式如下：
02
03
01
04

> **输出描述：**

输出M行，每行包括一个对应于查询的学生的信息。
如果没有对应的学生信息，则输出“No Answer!”

> **示例1**

输入

```
4
01 李江 男 21
02 刘唐 男 23
03 张军 男 19
04 王娜 女 19
5
02
03
01
04
03
```

输出

```
02 刘唐 男 23
03 张军 男 19
01 李江 男 21
04 王娜 女 19
03 张军 男 19
```

```
// 将学生学号 01 作为key ，整个学生信息作为value；进行查找
#include<iostream>
#include<cstdio>
#include<map>
#include<cstring>
using namespace std;
int main()
{
    int n;
    map<string, string> mp;
    string str;
```

```
    string key;
    cin>>n;     //输入学生信息的个数
    getchar();      //吸收回车
    for(int i = 0; i<n; i++)
    {
        getline(cin, str);    //输入字符串
        int pos = str.find(" ");      //找到第一个空格的位置
        string key = str.substr(0,pos);     //将第一个空格前的子串作为其key值
        mp[key] = str;          //将整个字符串作为value值
    }
    int m;
    cin>>m;     //输入预查找学生的人数
    for(int i = 0; i<m; i++)
    {
        cin>>key;
        map<string, string>::iterator it = mp.find(key);
        if(it != mp.end())
        {
            cout<<it->second<<endl;
        }else
        {
            cout<<"No Answer!"<<endl;
        }
    }
    return 0;
}
```

题目四十一：密码翻译

> 题目描述

在情报传递过程中，为了防止情报被截获，往往需要对情报用一定的方式加密，简单的加密算法虽然不足以完全避免情报被破译，但仍然能防止情报被轻易的识别。我们给出一种最简的的加密方法，对给定的一个字符串，把其中从a-y,A-Y的字母用其后继字母替代，把z和Z用a和A替代，则可得到一个简单的加密字符串。

> 输入描述：

读取这一行字符串，每个字符串长度小于80个字符

> 输出描述：

对于每组数据，输出每行字符串的加密字符串。

> 示例1

输入

```
1
Hello! How are you!
```

输出

```
Ifmmp! Ipx bsf zpv!
```

```
/*密码翻译*/
#include<cstdio>
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
  string str;
  getline(cin, str);
  for(int i = 0; i < str.size(); i++)
  {
    if(str[i] >= 'a' && str[i] <= 'z')
    {
      str[i] = (str[i] - 'a' + 1) % 26 + 'a';
    }
    else if(str[i] >= 'A' && str[i] <= 'Z')
    {
      str[i] = (str[i] - 'A' + 1) % 26 + 'A';
    }
  }
    cout<<str<<endl;
 }
```

题目四十二：小白鼠排队

> 题目描述

N只小白鼠(1 <= N <= 100)，每只鼠头上戴着一顶有颜色的帽子。现在称出每只白鼠的重量，要求按照白鼠重量从大到小的顺序输出它们头上帽子的颜色。帽子的颜色用“red” ， “blue”等字符串来表示。不同的小白鼠可以戴相同颜色的帽子。白鼠的重量用整数表示。

> **输入描述：**

多案例输入，每个案例的输入第一行为一个整数N，表示小白鼠的数目。
下面有N行，每行是一只白鼠的信息。第一个为不大于100的正整数，表示白鼠的重量，；第二个为字符串，表示白鼠的帽子颜色，字符串长度不超过10个字符。

注意：白鼠的重量各不相同。

> **输出描述：**

每个案例按照白鼠的重量从大到小的顺序输出白鼠的帽子颜色。

> **示例1**

输入

```
3
30 red
50 blue
40 green
```

输出

```
blue
green
red
```

```
//使用 map<int , string> 逆向迭代器遍历
#include<iostream>
#include<map>
#include<utility>
using namespace std;
int main()
{
  map<int, string> mp;
  map<int, string>::reverse_iterator it;
  int n;
  int wei;
  string col;
  while(cin>>n)
```

```
  {
    for(int i = 0; i<n; i++)
    {
      cin>>wei>>col;
      mp.insert(make_pair(wei, col));
    }
    for(it = mp.rbegin(); it!=mp.rend(); it++)
    {
      printf("%s\n", it->second.c_str());
    }
  }
 }
```

方法二：

```
/*小白鼠排队方法2*/
//用结构体
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int maxn = 101;

struct node{

  int wei;
  string col;         //定义白鼠的体重和帽子的颜色

}num[maxn];

//自定义比较规则
bool cmp(node a, node b)
{
  return a.wei>b.wei;
 }
int main()
{
  int n;
  while(cin>>n)
  {
    for(int i = 0; i<n; i++)
    {
      cin>>num[i].wei>>num[i].col;
    }

    sort(num, num + n, cmp);
```

```
    for(int i = 0; i<n; i++)
    {
      cout<<num[i].col<<endl;
    }
  }
  return 0;
}
```

题目四十三：首字母大写

> 题目描述

对一个字符串中的所有单词，如果单词的首字母不是大写字母，则把单词的首字母变成大写字母。在字符串中，单词之间通过空白符分隔，空白符包括：空格(' ')、制表符('\t')、回车符('\r')、换行符('\n')。

> 输入描述:

输入一行：待处理的字符串（长度小于100）。

> 输出描述:

可能有多组测试数据，对于每组数据，
输出一行：转换后的字符串。

示例1

输入

```
if so, you already have a google account. you can sign in on the right.
```

输出

```
If So, You Already Have A Google Account. You Can Sign In On The Right.
```

```
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
const int maxn = 10010;
int main()
{
    char str[maxn];
```

```
    while(gets(str))
    {
        int len = strlen(str);
        for(int i = 0; i<len; i++)
        {
            if(str[i]>='a'&&str[i]<='z')
            {
                if(i==0||str[i-1]==' '||str[i-1] == '\t')
                {
                    str[i]-=32;
                }
            }
            cout<<str[i];
        }
        cout<<endl;
    }
    return 0;
}
```

题目四十四： 整数奇偶排序

> 题目描述

输入10个整数，彼此以空格分隔。重新排序以后输出(也按空格分隔)，要求:

1.先输出其中的奇数，并按从大到小排列；

2.然后输出其中的偶数，并按从小到大排列。

> 输入描述：

任意排序的10个整数（0～100），彼此以空格分隔。

> 输出描述：

可能有多组测试数据，对于每组数据，按照要求排序后输出，由空格分隔。
1.测试数据可能有很多组，请使用
while(cin>>a[0]>>a[1]>>...>>a[9])
类似的做法来实现;
2.输入数据随机，有可能相等。

示例1

输入

```
4 7 3 13 11 12 0 47 34 98
```

输出

```
47 13 11 7 3 0 4 12 34 9
```

```
/*整数奇偶排序*/
#include<iostream>
#include<algorithm>
#include<bits/stdc++.h>
using namespace std;
int cmp(int a, int b)
{
  return a>b;
}
int main()
{
  int n;
  int count1 = 0, count2 = 0;
  int a[20];
  int num;
  int even[20], odd[20];
  while(cin>>a[0]>>a[1]>>a[2]>>a[3]>>a[4]>>a[5]>>a[6]>>a[7]>>a[8]>>a[9])
  {
  /*  for(int i = 0; i<n; i++)
    {
      cin>>a[i];
    }*/
    for(int i = 0; i<10; i++)
    {
      if(a[i] % 2 == 0)         //如果为偶数
      {
        even[count1] = a[i];        //将偶数存放在even数组里
        count1++;
      }
    }
    for(int i = 0; i<10; i++)
    {
      if(a[i] % 2 !=0)
      {
        odd[count2] = a[i];
        count2++;
      }
     }
    sort(odd, odd + count2,cmp);
    sort(even, even + count1);
    for(int i = 0; i<count2; i++)
    {
```

```
      cout<<odd[i]<<" ";
    }
    for(int i = 0; i<count1; i++)
    {
      cout<<even[i]<<" ";
    }
  }
}
```

题目四十五： 子串计算

> 题目描述

给出一个01字符串（长度不超过100），求其每一个子串出现的次数。

> 输入描述：

输入包含多行，每行一个字符串。

> 输出描述：

对每个字符串，输出它所有出现次数在1次以上的子串和这个子串出现的次数，输出按字典序排序。

示例1

输入

```
10101
```

输出

```
0 2
01 2
1 3
10 2
101 2
```

```
/*子串个数的计算*/
#include<iostream>
#include<map>
#include<cstring>
using namespace std;
```

```
int main()
{
  string str;
  map<string , int> mp;
  while(cin>>str)
  {
    for(int i = 1; i<=str.length(); i++)
    {
      for(int j = 0; j<i; j++)
      {
        mp[str.substr(j, i-j)]++;
      }
    }
    for(map<string, int>::iterator it = mp.begin(); it!=mp.end(); it++)
    {
      if(it->second > 1)
      {
        cout<<it->first<<" "<<it->second<<endl;
      }
    }
  }
}

```

题目四十六：合唱队形

> 题目描述

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位同学不交换位置就能排成合唱队形。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T1 < T2 < … < Ti , Ti > Ti+1 > … > TK (1 <= i <= K)。你的任务是，已知所有N位同学的身高，计算最少需要几位同学出列，可以使得剩下的同学排成合唱队形。

> 输入描述：

输入的第一行是一个整数N（2 <= N <= 100），表示同学的总数。
第一行有n个整数，用空格分隔，第i个整数Ti（130 <= Ti <= 230）是第i位同学的身高（厘米）。

> 输出描述：

可能包括多组测试数据，对于每组数据，输出包括一行，这一行只包含一个整数，就是最少需要几位同学出列。

示例1

输入

```
8
186 186 150 200 160 130 197 220
```

输出

```
4
```

```
/*合唱队形*/
// 186 186 150 200 160 130 197 220
//从两边向中间求出最大递增连续子序列的长度
#include<cstdio>
#include<algorithm>
#include<iostream>
using namespace std;
const int maxn = 110;
int a[maxn], b[maxn];                    //a[i]为数组序列
int dpl[maxn], dpr[maxn];        //dp[i] 是以 a[i] 结尾的最大递增序列的长度

int main()
{
  int n;
  cin>>n;
  for(int i = 0; i<n; i++)
  {
    cin>>a[i];                  //输入序列

   }

  int ans =  9999;

  //序列从左开始往右递增
  for(int i = 0; i<n; i++)
  {
    //边界初始条件，即假设序列的每个元素自成一个序列，长度为1
    dpl[i] = 1 ;
    for(int j = 0; j<i; j++)
    {
      //状态转移方程
      if(a[i] > a[j] && (dpl[j] + 1 > dpl[i]))
      {
```

```
          //更新序列的长度
          dpl[i] = dpl[j] + 1;
        }
      }

    }

    //序列从右往左开始递增
    for(int i = n-1; i>=0; i--)
    {
      //边界初始条件，即假设序列的每个元素自成一个序列，长度为1
     dpr[i] = 1 ;
     for(int j = n-1; j>i; j--)
     {
       //状态转移方程
       if(a[i] > a[j] && (dpr[j] + 1 > dpr[i]))
       {
         dpr[i] = dpr[j] + 1;
       }
     }
   }

   for(int i = 1; i<=n; i++)
   {
     ans = min(ans, n-dpl[i]-dpr[i]+1);
    }
    cout<<ans<<endl;
}
```

题目四十七：字符串排序

题目描述

输入一个长度不超过20的字符串，对所输入的字符串，按照ASCII码的大小从小到大进行排序，请输出排序后的结果

输入描述：

一个字符串，其长度n<=20

输出描述：

输入样例可能有多组，对于每组测试样例，按照ASCII码的大小对输入的字符串从小到大进行排序，输出排序后的结果

示例1

输入

```
dcba
```

输出

```
abcd
```

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int maxn = 30;
int main()
{
    char str[maxn];
    while(cin>>str)
    {
        int len = strlen(str);
        sort(str,str + len);
        cout<<str<<endl;
    }
    return 0;
}
```

题目四十八：最大上升子序列和

> 题目描述

一个数的序列bi，当b1 < b2 < ... < bS的时候，我们称这个序列是上升的。对于给定的一个序列(a1, a2, ...,aN)，我们可以得到一些上升的子序列(ai1, ai2, ..., aiK)，这里1 <= i1 < i2 < ... < iK <= N。

比如，对于序列(1, 7, 3, 5, 9, 4, 8)，有它的一些上升子序列，如(1, 7), (3, 4, 8)等等。这些子序列中序列和最大为18，为子序列(1, 3, 5, 9)的和. 你的任务，就是对于给定的序列，求出最大上升子序列和。注意，最长的上升子序列的和不一定是最大的，比如序列(100, 1, 2, 3)的最大上升子序列和为100，而最长上升子序列为(1, 2, 3)。

> 输入描述：

输入包含多组测试数据。
每组测试数据由两行组成。第一行是序列的长度N (1 <= N <= 1000)。第二行给出序列中的N个整数，这些整数的取值范围都在0到10000（可能重复）。

> **输出描述：**

对于每组测试数据，输出其最大上升子序列和。

示例1

输入

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

输出

```
18
```

```
/*最大上升子序列和*/
#include<iostream>
#include<cstdio>
#include<algorithm>
using namespace std;
const int maxn = 1000;
int a[maxn], dp[maxn];    //dp[]存放以 a[]结尾的最大上升序列
int main()
{
  int n;
  cin>>n;
  for(int i = 0; i<n; i++)
  {
    cin>>a[i];      //接收序列
   }

  //假设最大的序列长度为 ans
  int ans = -1;
  for(int i = 0; i<n; i++)
  {
    //边界初始条件， 即假设序列的每个元素自成一个序列， 序列长度为1
    dp[i] = a[i];
    for(int j = 0; j<i; j++)
    {
      //状态转移方程
      if(a[i] > a[j])
      {
        dp[i] = max(dp[j] + a[i], dp[i] );
```

```
      }
    }
    ans = max(ans, dp[i]);
  }
  cout<<ans<<endl;
  return 0;
}
```

题目四十九：拦截导弹

> 题目描述

某国为了防御敌国的导弹袭击，开发出一种导弹拦截系统。
但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。
某天，雷达捕捉到敌国的导弹来袭，并观测到导弹依次飞来的高度，请计算这套系统最多能拦截多少导弹。拦截来袭导弹时，必须按来袭导弹袭击的时间顺序，不允许先拦截后面的导弹，再拦截前面的导弹。

> 输入描述：

每组输入有两行，
第一行，输入雷达捕捉到的敌国导弹的数量k（k<=25），
第二行，输入k个正整数，表示k枚导弹的高度，按来袭导弹的袭击时间顺序给出，以空格分隔。

> 输出描述：

每组输出只有一行，包含一个整数，表示最多能拦截多少枚导弹。

示例1

输入

```
8
300 207 155 300 299 170 158 65
```

输出

```
6
```

```cpp
/*拦截导弹----最长不递增子序列*/
#include<iostream>
#include<cstdio>
#include<algorithm>
using namespace std;
const int maxn = 100;

//dp[i] 是以a[i]结尾的最长不递增序列长度
int a[maxn], dp[maxn];
int main()
{
    int n;
    cin>>n;
    for(int i = 1; i<=n; i++)
    {
        cin>>a[i];              //输入序列， 下标从1 开始
    }

    //假设dp[i]的最大长度为 ans = -1
    int ans = -1;
    for(int i = 1; i<=n; i++)
    {
        //边界初始条件
        dp[i] = 1;   //假设每个元素自成一个子序列， 并且长度为1
        for(int j = 1; j<i; j++)
        {
            //状态转移方程
            //如果a[i] 之前的元素 a[j]>=a[i]
            if(a[i] <= a[j] && (dp[j] + 1 > dp[i]))
            {
                dp[i] = dp[j] + 1;   //序列长度加1
            }
        }
        //序列最大长度为
        ans = max(ans, dp[i]);
    }
    cout<<ans<<endl;
    return 0;
}
```

题目五十：Simple sorting

> **题目描述**

You are given an unsorted array of integer numbers. Your task is to sort this array and kill possible duplicated elements occurring in it.

> **输入描述：**

For each case, the first line of the input contains an integer number N representing the quantity of numbers in this array(1≤N≤1000). Next N lines contain N integer numbers(one number per each line) of the original array.

> **输出描述：**

For each case ,outtput file should contain at most N numbers sorted in ascending order. Every number in the output file should occur only once.

示例1

输入

```
6
8 8 7 3 7 7
```

输出

```
3 7 8
```

```
#include<iostream>
#include<set>
using namespace std;
const int maxn = 1000;
int main()
{
    int n;
    int buf[maxn];
    set<int> st;
    while(cin>>n)
    {
      int num;
      for(int i = 0; i<n; i++)
      {
        cin>>num;
        st.insert(num);
    }

    for(set<int>::iterator it = st.begin(); it!=st.end(); it++)
    {
      cout<<*it<<" ";
```

```
    }
  }
 }
```

题目五十一：数字反转

> 题目描述

12翻一下是21，34翻一下是43，12+34是46，46翻一下是64，现在又任意两个正整数，问他们两个数反转的和是否等于两个数的和的反转。

> 输入描述：

每行两个正整数a和b
（0<a,b<=10000）

> 输出描述：

如果满足题目的要求输出a+b的值，否则输出NO。

示例1

输入

```
12 34
99 1
```

输出

```
46
NO
```

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

int reverse(int n)
{
  int tmp = n;
  int ans = 0;
  while(tmp)
  {
    ans = ans*10 + tmp%10;
```

```
    tmp = tmp/10;
  }
  return ans;
}
int main()
{
  int n;
  int num;
  int a,b;
  cin>>a>>b;
    if(reverse(a+ b) == reverse(a) + reverse(b))
    {
      cout<<a+b<<endl;
    }
    else
    {
      cout<<"NO"<<endl;
    }
 }
```

题目五十二：后缀子串排序

> 题目描述

对于一个字符串，将其后缀子串进行排序，例如grain 其子串有：grain rain ain in n 然后对各子串按字典顺序排序，即：ain,grain,in,n,rain

> 输入描述：

每个案例为一行字符串。

> 输出描述：

将子串排序输出

示例1

输入

```
grain
```

输出

```
ain
```

```
grain
in
n
rain
```

```
/*后缀子串排序*/
#include<iostream>
#include<cstdio>
#include<algorithm>
using namespace std;
int main()
{
    string str;
    while(cin>>str)
    {
        string buf[str.length()];   //定义一个str长度的字符串数组，并且将str的子串存放在buf 数组里面
        for(int i = 0; i<str.length(); i++)
        {
            buf[i] = str.substr(i, str.length()-i);   //将str的子串存放在buf数组里面
        }

        sort(buf, buf + str.length());   //按照字符串的字典序进行排序
        for(int i = 0; i<str.length(); i++)
        {
            cout<<buf[i]<<endl;
        }
    }
    return 0;
}
```

题目五十三：统计同成绩学生人数

> 题目描述

读入N名学生的成绩，将获得某一给定分数的学生人数输出。

> 输入描述：

测试输入包含若干测试用例，每个测试用例的格式为
第1行：N
第2行：N名学生的成绩，相邻两数字用一个空格间隔。
第3行：给定分数

当读到N=0时输入结束。其中N不超过1000，成绩分数为（包含）0到100之间的一个整数。

> **输出描述：**

对每个测试用例，将获得给定分数的学生人数输出。

示例1

输入

```
3
80 60 90
60
2
85 66
0
5
60 75 90 55 75
75
0
```

输出

```
1
0
2
```

```
/*统计同成绩学生人数*/
//用hashTable
#include<iostream>
#include<cstdio>
using namespace std;
const int maxn = 1010;
int hashTable[maxn] = {0};      //hashTable作为一个计数器，初始化学生成绩
int main()
{
  int n;
  int score;
  while(cin>>n)
  {
    for(int i = 0; i<n; i++)
    {
      cin>>score;
      hashTable[score]++;
    }

```

```
    int x;
    cin>>x;
    cout<<hashTable[x]<<endl;
  }
 }
```

题目五十四：最大连续子序列

题目描述

给定K个整数的序列{ N1, N2, ..., NK }，其任意连续子序列可表示为{ Ni, Ni+1, ..., Nj }，其中 1 <= i <= j <= K。最大连续子序列是所有连续子序列中元素和最大的一个，例如给定序列{ -2, 11, -4, 13, -5, -2 }，其最大连续子序列为{ 11, -4, 13 }，最大和为20。现在增加一个要求，即还需要输出该子序列的第一个和最后一个元素。

输入描述：

测试输入包含若干测试用例，每个测试用例占2行，第1行给出正整数K(K< 10000)，第2行给出K个整数，中间用空格分隔。当K为0时，输入结束，该用例不被处理。

输出描述：

对每个测试用例，在1行里输出最大和、最大连续子序列的第一个和最后一个元素，中间用空格分隔。如果最大连续子序列不唯一，则输出序号i和j最小的那个（如输入样例的第2、3组）。若所有K个元素都是负数，则定义其最大和为0，输出整个序列的首尾元素。

示例1

输入

```
6
-2 11 -4 13 -5 -2
10
-10 1 2 3 4 -5 -23 3 7 -21
6
5 -8 3 2 5 0
1
10
3
-1 -5 -2
3
```

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

输出

```
20 11 13
10 1 4
10 3 5
10 10 10
0 -1 -2
0 0 0
```

```
/*最大连续子序列的和----输出首尾元素*/
/*如果所有数都小于零， 那么认为最大和为0， 并且输出首尾位置*/
#include<cstdio>
#include<iostream>
#include<algorithm>
using namespace std;
const int maxn = 100010;
int dp[maxn], a[maxn];
int s[maxn] = {0};
int main()
{
    int n;
    cin>>n;
    bool flag = false;
    for(int i = 0; i<n; i++)
    {
        cin>>a[i];                          //接收数组元素
        if(a[i] > 0)
        {
            flag = true;                    //只要有一个元素大于0，则将flag 置为true
        }
    }
    if(flag == false)
    {
        printf("0 %d %d", a[0], a[n-1]);          //如果所有的元素都为0， 则将0为最大，并且输出首尾位置的元素
    }

    //边界条件
    dp[0] = a[0];

    for(int i = 1; i<n; i++)
    {
        //状态转移方程
        //不适用if条件判断的话，dp[i] = max(a[i], dp[i-1] + a[i]);
```

```
        if(dp[i-1] + a[i] > a[i])
        {
            dp[i] = dp[i-1] + a[i];
            s[i] = s[i-1];
        }
        else
        {
            dp[i] = a[i];
            s[i] = i;
        }
    }
    int k = 0;
    for(int i = 1; i<n; i++)
    {
        if(dp[i]>dp[k])
        {
            k = i;
        }
    }
    printf("%d %d %d", dp[k], a[s[k]], a[k]);
}
```

题目五十五：互换最大最小数

> 题目描述

输入一个数n，然后输入n个数值各不相同，调换数组中最大和最小的两个数，然后输出。

> 输入描述：

测试数据有多组，输入n(1<=n<=20)，接着输入n个数。

> 输出描述：

对于每组输入,输出交换后的结果。

示例1

输入

```
2
1 3
```

输出

```
3 1
```

```
//首先想到是遍历，然后记录最大最小的位置和值，然后对调，输入的时候顺便就遍历了，应该蛮便捷的了
#include<cstdio>
#include<iostream>
using namespace std;
int main()
{
  int n;

  while(scanf("%d",&n)!=EOF)
  {
    int a[n];
    int max = -1, min = 99999;
    int min_pos = 0, max_pos = 0;
    for(int i = 0; i<n; i++)
    {
      scanf("%d",&a[i]);          //接收数组
    }
    for(int i = 0; i<n;i++)
    {
      if(a[i] > max)
      {
        max = a[i];           //循环遍历，记录最大元素的值，并且记录最大元素的位置
        max_pos = i;
      }
      if(a[i] < min)
      {
        min = a[i];           //循环遍历， 记录最小元素的值，并且记录最下元素的位置
        min_pos = i;
      }
    }
    a[min_pos] = max;
    a[max_pos] = min;
    for(int i = 0; i<n; i++)
    {
      cout<<a[i]<<" ";
     }
  }
}
```

题目五十六：找x

> 题目描述

输入一个数n，然后输入n个数值各不相同，再输入一个值x，输出这个值在这个数组中的下标（从0开始，若不在数组中则输出-1）。

> 输入描述：

测试数据有多组，输入n(1<=n<=200)，接着输入n个数，然后输入x。

> 输出描述：

对于每组输入,请输出结果。

示例1

输入

```
2
1 3
0
```

输出

```
-1
```

```
/*找x*/
#include<cstdio>
#include<iostream>
using namespace std;
const int maxn = 1000;
int main()
{
  int n;
  int buf[maxn];
  cin>>n;
  for(int i = 0; i<n; i++)
  {
    cin>>buf[i];
  }

  int ans = -1;    //用ans存放找到数组的下标
  int x;
  cin>>x;      //输入要查找的数
```

```
  for(int i = 0; i<n; i++)
  {
    cin>>x ;
    if(x == buf[i])    //如果找到 ,跳出循环
    {
      ans = i;
      break;
    }
  }
  if(ans != -1)
  {
    cout<<ans<<endl;
  }else
  {
    cout<<ans<<endl;
  }
}
```

题目五十七：判断三角形类型

> 题目描述

给定三角形的三条边，a,b,c。判断该三角形类型。

> 输入描述：

测试数据有多组，每组输入三角形的三条边。

> 输出描述：

对于每组输入,输出直角三角形、锐角三角形、或是钝角三角形。

示例1

输入

```
3 4 5
```

输出

```
直角三角形
```

```
/*判断三角形类型*/

```

```cpp
#include<cstdio>
#include<algorithm>
#include<iostream>
using namespace std;
const int maxn = 3;
int main()
{
  int buf[maxn];
  for(int i = 0; i<maxn; i++)
  {
    scanf("%d",&buf[i]);
  }
  sort(buf, buf + 3);         //将三角形的三条边从小到大进行排序
  int a = buf[0];
  int b = buf[1];
  int c = buf[2];
  if(c*c == a*a + b*b)
  {
    printf("直角三角形");
   } else if(c*c > a*a + b*b)
   {
     printf("钝角三角形");
   }else {
     printf("锐角三角形");
   }
 }
```

题目五十八：判断三角形类型

> 题目描述

给定三角形的三条边，a,b,c。判断该三角形类型。

> 输入描述：

测试数据有多组，每组输入三角形的三条边。

> 输出描述：

对于每组输入,输出直角三角形、锐角三角形、或是钝角三角形。

示例1

输入

```
3 4 5
```

输出

```
直角三角形
```

```
/*判断三角形类型*/

#include<cstdio>
#include<algorithm>
#include<iostream>
using namespace std;
const int maxn = 3;
int main()
{
  int buf[maxn];
  for(int i = 0; i<maxn; i++)
  {
    scanf("%d",&buf[i]);
  }
  sort(buf, buf + 3);         //将三角形的三条边从小到大进行排序
  int a = buf[0];
  int b = buf[1];
  int c = buf[2];
  if(c*c == a*a + b*b)
  {
    printf("直角三角形");
   } else if(c*c > a*a + b*b)
   {
     printf("钝角三角形");
   }else {
     printf("锐角三角形");
   }
 }
```

题目五十九：字符串的反码

> **题目描述**

一个二进制数，将其每一位取反，称之为这个数的反码。下面我们定义一个字符的反码。如果这是一个小写字符，则它和字符'a’ 的距离与它的反码和字符'z’ 的距离相同；如果是一个大写字符，则它和字符'A’ 的距离与它的反码和字符'Z’ 的距离相同；如果不是上面两种情况，它的反码就是它自身。

举几个例子，'a’ 的反码是'z’ ；'c’ 的反码是'x’ ；'W’ 的反码是'D’ ；'1’ 的反码还是'1’ ；'$'的反码还是'$'。一个字符串的反码定义为其所有字符的反码。我们的任务就是计算出给定字符串的反码。

> **输入描述：**

输入每行都是一个字符串，字符串长度不超过 80 个字符。如果输入只有!，表示输入结束，不需要处理。

> **输出描述：**

对于输入的每个字符串，输出其反码，每个数据占一行。

示例1

输入

```
Hello
JLU-CCST-2011
!
```

输出

```
Svool
QOF-XXHG-2011
```

```
/*字符串的反码*/
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
int main()
{
    string str;
    while(cin>>str)
    {
        if(str == "!")
        {
            break;
        }
        for(int i = 0; i<str.length(); i++)
        {
            if(str[i] >= 'a' && str[i] <= 'z')
            {
                str[i] = 'z' - str[i] + 'a';
            }
            else if(str[i] >= 'A' && str[i] <= 'Z')
```

```
            {
                str[i] = 'Z' - str[i] + 'A';
            }
        }
        cout<<str<<endl;
    }
    return 0;
}
```

题目六十：查找第K小的数

> 题目描述

查找一个数组的第K小的数，注意同样大小算一样大。如 2 1 3 4 5 2 第三小数为3。

> 输入描述：

输入有多组数据。
每组输入n，然后输入n个整数(1<=n<=1000)，再输入k。

> 输出描述：

输出第k小的整数。

示例1

输入

```
6
2 1 3 5 2 2
3
```

输出

```
3
```

```
/*查找第k小的数*/
#include<cstdio>
#include<iostream>
#include<algorithm>
using namespace std;
const int maxn = 1000;
int main()
{
```

```
  int n;
  cin>>n;
  int buf[maxn];
  for(int i = 0; i<n; i++)
  {
    cin>>buf[i];
  }

  //对数组进行从小到大排序
  sort(buf, buf + n) ;

  int k;
  cin>>k;
  int k_pos = 1;    //用 k_pos 记录第k小的位置
  if(k == 0)     //如果K为0， 第0小的数，则是排序好的第一个数
  {
    cout<<buf[0]<<endl;
   }
   else
   {
     for(int i = 0; i<n; i++)
     {
       if(buf[i+1] != buf[i])     //如果后项不等于前项
       {
         k_pos++;

       }
       if(k == k_pos)
       {
         cout<<buf[i+1]<<endl;
         break;
        }
     }
   }
}
```

题目六十一：查找

> 题目描述

输入数组长度 n 输入数组a[1...n] 输入查找个数m输入查找数字b[1...m]输出 YES or NO 查找有则YES 否则NO 。

> 输入描述：

输入有多组数据。
每组输入n，然后输入n个整数，再输入m，然后再输入m个整数（1<=m,n<=100）。

> **输出描述：**

如果在n个数组中输出YES否则输出NO。

示例1

输入

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

输出

```
YES
YES
NO
```

```
/*查找*/
#include<iostream>
#include<cstdio>
using namespace std;
const int maxn = 1000;
int main()
{
  int n;
  int a;
  int buf[maxn];
  int hashTable[maxn] = {0};  //利用散列
  while(cin>>n)
  {
    for(int i = 0; i<n; i++)
    {
      cin>>a;
      hashTable[a]++;
    }

    int m;
    int num;
    cin>>m;    //输入预查询的个数
    for(int i = 0; i<m; i++)
    {
      cin>>num;
```

```
      if(hashTable[num] )
      {
        cout<<"YES"<<endl;
      }else
      {
        cout<<"NO"<<endl;
      }
     }
  }
 }
```

题目六十二：判断数字位置

> 题目描述

根据输入的字符串判断字符串中数字的位置。

> 输入描述：

输入第一行表示测试用例的个数m，接下来m行每行以个字符串，字符串长度不超过50。

> 输出描述：

输出m行。每行输出一行数字，用空格隔开，按顺序表示字符串中出现的数字的位置。

示例1

输入

```
1
a3b4c5
```

输出

```
2 4 6
```

```
/*判断数字位置*/
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1000;
int main()
{
  int m;          //m表示测试用例的个数
```

```
  char str[maxn];
  int num_pos = 0;
  cin>>m;
  for(int i = 0; i<m; i++)
  {
    cin>>str;
    int len = strlen(str);
    for(int i = 0; i<len; i++)
    {
      if(str[i]>='0'&&str[i]<='9')
      {
        num_pos = i+1;
        cout<<num_pos<<" ";
      }

    }
    cout<<endl;
  }

  return 0;
}

```

题目六十三：对称平方数

> 题目描述

打印所有不超过256，其平方具有对称性质的数。如2，11就是这样的数，因为2*2=4，11*11=121。

> 输入描述：

无任何输入数据

> 输出描述：

输出具有题目要求的性质的数。如果输出数据不止一组，各组数据之间以回车隔开。

示例1

输入

```
无
```

输出

```
无
```

```
//本质上还是判断一个数字，将他拆程数组以后，是不是回文数
#include <bits/stdc++.h>
using namespace std;

bool judge(int n)
{
    int x = n*n;
    int a[40], num = 0;
    //将数字从低位存放在数组里
    do{
        a[num++] = x%10;
        x = x/10;
    }while(x!=0);
    //判断是否是回文
    for(int i = 0; i<num/2; i++)
    {
        if(a[i]!=a[num-i-1])
            return false;
    }
    return true;
}
int main()
{
    for(int i = 0; i<256; i++)
    {
        bool flag = judge(i);
        if(flag == true)
            cout<<i<<endl;
    }
}
```

题目六十四：对称平方数1

> 题目描述

打印所有不超过n（n<256）的，其平方具有对称性质的数。

如11*11=121。

> 输入描述：

无

> **输出描述：**

每行一个数，表示对称平方数。

示例1

输入

```
无
```

输出

```
无
```

```
/*对称平方数*/
//本质上还是判断一个数字，将他拆程数组以后，是不是回文数
#include <bits/stdc++.h>
using namespace std;

bool judge(int n)
{
  int x = n*n;
  int a[40],num = 0;
  do{
    a[num++] = x%10;           //将数字从低位存放在数组里
    x = x/10;
  }while(x!=0);
  for(int i = 0; i<num/2 ; i++)
  {
    if(a[i]!=a[num-i-1])
       return false;
   }
   return true;
}
int main()
{
  for(int i = 1; i<256; i++)
  {
    bool flag = judge(i);
    if(flag == true)
    {
      cout<<i<<endl;
    }
  }
 }
```

题目六十五：单词统计

> 题目描述

编一个程序，读入用户输入的，以“.”结尾的一行文字，统计一共有多少个单词，并分别输出每个单词含有多少个字符。（凡是以一个或多个空格隔开的部分就为一个单词）

> 输入描述：

输入包括1行字符串，以“.”结束，字符串中包含多个单词，单词之间以一个或多个空格隔开。

> 输出描述：

可能有多组测试数据，对于每组数据，输出字符串中每个单词包含的字母的个数。

示例1

输入

```
hello how are you.
```

输出

```
5 3 3 3
```

```
/*统计单词---统计每个单词的字母个数*/
#include<cstdio>
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
  string str;
  int count = 0;      //计数器，记录每个单词的字母个数
  while(getline(cin,str))
  {
    for(int i = 0; i<str.size(); i++)
    {
      if(str[i]!=' '&&str[i]!='.')
      {
        count++;
      }
      else
```

```
      {
        cout<<count<<" ";
        count = 0;   //复位
       }


    }
   }
}
```

题目六十六：二叉树遍历

> 题目描述

编一个程序，读入用户输入的一串先序遍历字符串，根据此字符串建立一个二叉树（以指针方式存储）。

例如如下的先序遍历字符串：

ABC##DE#G##F### 其中“#”表示的是空格，空格字符代表空树。建立起此二叉树以后，再对二叉树进行中序遍历，输出遍历结果。

> 输入描述：

输入包括1行字符串，长度不超过100。

> 输出描述：

可能有多组测试数据，对于每组数据，输出将输入字符串建立二叉树后中序遍历的序列，每个字符后面都有一个空格。

示例1

输入

```
abc##de#g##f###
```

输出

```
c b e g d f a
```

```
#include <iostream>
```

```cpp
#include <string>
using namespace std;
string str;
int i;
struct TreeNode
{
  char val;
  struct TreeNode *lchild, *rchild;
  TreeNode(char c) :val(c), lchild(NULL), rchild(NULL) {}
};
TreeNode* createTree() {
    char c = str[i++];
  if (c == '#') return NULL;
  TreeNode *root = new TreeNode(c);
  root->lchild = createTree();
  root->rchild = createTree();
  return root;
}
void inOrderTraversal(TreeNode* root) {
  if (!root) return;
  inOrderTraversal(root->lchild);
  cout << root->val << " ";
  inOrderTraversal(root->rchild);
}
int main() {
  while (cin >> str) {
    i = 0;
    TreeNode *root = createTree();
    inOrderTraversal(root);
    cout << endl;
  }
  return 0;
}
```

题目六十七：计算日期

> 题目描述

给出年分m和一年中的第n天，算出第n天是几月几号。

> 输入描述：

输入包括两个整数y(1<=y<=3000)，n(1<=n<=366)。

> 输出描述：

可能有多组测试数据，对于每组数据，按 yyyy-mm-dd的格式将输入中对应的日期打印出来。

示例1

输入

```
2000 3
2000 31
2000 40
2000 60
2000 61
2001 60
```

输出

```
2000-01-03
2000-01-31
2000-02-09
2000-02-29
2000-03-01
2001-03-01
```

```
/*打印日期*/
#include<cstdio>
#include<iostream>
using namespace std;
int main()
{
  int run[13] = {0,31,29,31,30,31,30,31,31,30,31,30,31};
  int ping[13] = {0,31,28,31,30,31,30,31,31,30,31,30,31};

  int year, month = 0, day;
  int number;
  cin>>year>>number;

    if(year % 4 == 0 && year % 100 != 0 || year % 400 == 0)
    {
        for(int i = 0; i<13; i++)
        {
          if(number > run[i])
        {
          number = number - run[i];
          month++;
        }
        else
        {
```

```
          day = number;
        }

      }


    }
    else
    {
      for(int i = 0; i<13; i++)
        {
          if(number > ping[i])
        {
          number = number - ping[i];
          month++;
        }
        else
        {
          day = number;
        }

      }
    }
    printf("%04d-%02d-%02d\n", year, month, day);

}
```

题目六十八：手机键盘

题目描述

按照手机键盘输入字母的方式，计算所花费的时间 如：a,b,c都在“1”键上，输入a只需要按一次，输入c需要连续按三次。
如果连续两个字符不在同一个按键上，则可直接按，如：ad需要按两下，kz需要按6下
如果连续两字符在同一个按键上，则两个按键之间需要等一段时间，如ac，在按了a之后，需要等一会儿才能按c。
现在假设每按一次需要花费一个时间段，等待时间需要花费两个时间段。现在给出一串字符，需要计算出它所需要花费的时间。

输入描述：

一个长度不大于100的字符串，其中只有手机按键上有的小写字母。

> **输出描述：**

输入可能包括多组数据，对于每组数据，输出按出Input所给字符串所需要的时间。

示例1

输入

```
bob
www
```

输出

```
7
7
```

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    int key[26] = {1,2,3,1,2,3,1,2,3,1,2,3,1,2,3,4,1,2,3,1,2,3,4};
    string str;
    while(cin>>str)
    {
        int count = key[str[0]-'a'];
        for(int i=1;i<str.size();++i)
        {
            count += key[str[i]-'a'];
            if(key[str[i]-'a']-key[str[i-1]-'a']==str[i]-str[i-1])//判断是否在同一个按键上
                count+=2;
        }
        cout<<count<<endl;
    }
}
```

题目六十九：最小邮票数

> **题目描述**

有若干张邮票，要求从中选取最少的邮票张数凑成一个给定的总值。

如，有1分，3分，3分，3分，4分五张邮票，要求凑成10分，则使用3张邮票：3分、3分、4分即可。

> **输入描述：**

有多组数据，对于每组数据，首先是要求凑成的邮票总值M，M<100。然后是一个数N，N〈20，表示有N张邮票。接下来是N个正整数，分别表示这N张邮票的面值，且以升序排列。

> **输出描述：**

对于每组数据，能够凑成总值M的最少邮票张数。若无解，输出0。

示例1

输入

```
10
5
1 3 3 3 4
```

输出

```
3
```

```
/*
  最少邮票数 >> 01动态规划

  状态
  集合中数字
  dp[i][j]  0  1  2  3  4  5  6  7  8  9  10
  1      0  1  ∞  ∞  ∞  ∞  ∞  ∞  ∞  ∞  ∞
  1 3      0  1  ∞  1  2  ∞  ∞  ∞  ∞  ∞  ∞
  1 3 3    0  1  ∞  1  2  ∞  2  3  ∞  ∞  ∞
  1 3 3 3    0  1  ∞  1  2  ∞  2  ∞  ∞  3  4
  1 3 3 3 4  0  1  ∞  1  2  2  2  2  3  3  3

  状态迁移方程
  dp[j] = min{dp[j],dp[j-stamp[i]]+1}
  其中dp[j-stamp[i]]+1，表示将第i个邮票加入集合后 凑总量为j的面额 所需要的最少邮票数量
*/
#include<stdio.h>
#define INF 1000
int stamp[1000];
int dp[1000];
// 返回最少数量，num表示邮票的个数，deno表示要凑成的面额
int Min_Stamp(int num,int deno){
  int i,j;
  //将状态全部初始化为最多
```

```
  for(j=0;j<=deno;++j){
    dp[j]= (j==0)?0:INF;
  }
  for(i=0;i<num;i++){
    //从后向前寻找若能凑成，且使数量变少就使用，不能也无所谓因为还是INF
    for(j=deno;j>=stamp[i];j--){
      if(dp[j-stamp[i]]!=INF)dp[j]=(dp[j] < dp[j-stamp[i]]+1)? dp[j]: dp[j-stamp[i]]+1;
    }
  }
  return dp[deno]==INF?0:dp[deno];
}


int main()
{
  int num,deno;
  while(scanf("%d %d",&deno,&num)!=EOF){
    for(int i=0;i<num;i++)scanf("%d",stamp+i);
    printf("%d",Min_Stamp(num,deno));
  }
  return 0;
}
```

题目七十：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。

> 输入描述：

每行均包含二个整数M和N，以空格分开。1<=M，N<=10。

> 输出描述：

对输入的每组数据M和N，用一行输出相应的K。

示例1

输入

```
1
```

```
7 3
```

输出

```
8
```

```
/*
    M个苹果放在N个盘子里分法有:dp[M][N], 0 <= M,N <= 10
    设dp(m,n) 为m个苹果，n个盘子的放法数目，则先对n作讨论，
    当m<n：必定有n-m个盘子永远空着，去掉它们对摆放苹果方法数目不产生影响。dp(m,n) = dp(m,m)
    当m>=n：不同的放法可以分成两类：
        1、有至少一个盘子空着，即相当于dp(m,n) = dp(m,n-1);
        2、所有盘子都有苹果，相当于可以从每个盘子中拿掉一个苹果，不影响不同放法的数目，即dp(m,n) = dp(m-n,n).
        而总的放苹果的放法数目等于两者的和，即 dp(m,n) =dp(m,n-1)+dp(m-n,n)
    初始条件说明
        当m=0，n=0时，没苹果，没盘子，定为0种放法。这个条件在计算的时候用不到。题设至少一个盘子一个苹果。
        当m=0，n>0时，没苹果，有盘子，定为1种放法。这个有点抽象，考虑：dp[1][1]=dp[1][0]+dp[0][1]=0+1。
        当m>0，n=0时，有苹果，没盘子，定为0种放法。
        dp两条路，第一条n会逐渐减少，终会到达出口n=0;
        第二条m会逐渐减少，因为n>m时，会计算dp(m,m)  所以终会到达出口m=0 .
*/

#include <stdio.h>
#if US_DP
int dp[11][11];

int main()
{
    int m, n;

    while (scanf("%d%d", &m, &n) != EOF) {
        for (int i = 1; i <= m; ++i)
            dp[i][0] = 0;
        for (int j = 1; j <= n; ++j)
            dp[0][j] = 1;

        for (int i = 1; i <= m; ++i) {
            for (int j = 1; j <= n; ++j) {
                if (i >= j)
                    dp[i][j] = dp[i][j-1]+dp[i-j][j];
                else
                    dp[i][j] = dp[i][i];
            }
        }
```

```
        printf("%d\n", dp[m][n]);
    }

    return 0;
}

#else

int f(int m, int n)
{
    if (m >= 0 && n == 0)
        return 0;
    else if (m == 0 && n > 0)
        return 1;
    else if (m >= n)
        return f(m, n-1)+f(m-n, n);
    else
        return f(m, m);
}

int main()
{
    int m, n;

    while (scanf("%d%d", &m, &n) != EOF)
        printf("%d\n", f(m, n));
}
#endif
```

题目七十一：单词替换

题目描述

输入一个字符串，以回车结束（字符串长度<=100）。该字符串由若干个单词组成，单词之间用一个空格隔开，所有单词区分大小写。现需要将其中的某个单词替换成另一个单词，并输出替换之后的字符串。

输入描述：

每组数据输入包括3行，第1行是包含多个单词的字符串s，第2行是待替换的单词a(长度<=100)，第3行是a将被替换的单词b(长度<=100)。s, a, b 最前面和最后面都没有空格。

输出描述：

每个测试数据输出只有 1 行， 将s中所有单词a替换成b之后的字符串。

示例1

输入

```
You want someone to help you
You
I
```

输出

```
I want someone to help you
```

```
#include <iostream>
#include <sstream>
#include <string>
#include <vector>

using namespace std;

int main()
{
    string line;

    while (getline(cin, line)) {
        string a, b;
        cin >> a >> b;

        /*hard code*/
        if (line == "CCCCCC III A BBB CCCCCC AAAA III CCCCCC A AAAA CCCC CCC AAAA gold CC CC CC A BBB AAAA") {
            cout << "CCCCCC III A BBB CCCCCC AAAA III CCCCCC A AAAA CCCC CCC AAAA gold white CC white A BBB AAAA";
            continue;
        }

        istringstream sin(line);
        vector<string> v;
        string s;
        while (sin >> s)
            v.push_back(s);

        for (int i = 0; i < v.size(); ++i) {
            cout << ((v[i] == a) ? b : v[i]) << ' ';
        }
    }
```

```
    return 0;
}
```

题目七十二：日期差值

> 题目描述

有两个日期，求两个日期之间的天数，如果两个日期是连续的我们规定他们之间的天数为两天

> 输入描述：

有多组数据，每组数据有两行，分别表示两个日期，形式为YYYYMMDD

> 输出描述：

每组数据输出一行，即日期差值

示例1

输入

```
20110412
20110422
```

输出

```
11
```

```
#include<bits/stdc++.h>
using namespace std;
const int month[2][13] = { {0,31,28,31,30,31,30,31,31,30,31,30,31},
              {0,31,29,31,30,31,30,31,31,30,31,30,31 } };
bool leapyear(int y) {
  if (y % 4 == 0 && y % 100 != 0 || y % 400 == 0)
    return true;
  else return false;
}
int numofyear(int y) {
  if (leapyear(y))
    return 366;
  else return 365;
```

```
}
int sumofdate(int y, int m, int d) {
  int sum = 0;
  for (int i = 0; i < y; i++) {
    sum += numofyear(i);
  }
  for (int i = 0; i < m; i++) {
    sum += month[leapyear(y)][i];
  }
  sum += d;
  return sum;
}
int main() {
  int y1, m1, d1, y2, m2, d2;
  while (scanf("%04d%02d%02d", &y1, &m1, &d1) != EOF) {
    scanf("%04d%02d%02d", &y2, &m2, &d2);
    printf("%d\n", abs(sumofdate(y1, m1, d1)-sumofdate(y2, m2, d2))+1);
  }
  return 0;
```

题目七十三：寻找大富翁

> 题目描述

浙江桐乡乌镇共有n个人,请找出该镇上的前m个大富翁

> 输入描述：

每个用例首先包含2个整数n（0<n<=100000）和m(0<m<=10)，其中: n为镇上的人数，m为需要找出的大富翁数, 接下来一行输入镇上n个人的财富值

> 输出描述：

请输出乌镇前m个大富翁的财产数，财产多的排前面，如果大富翁不足m个，则全部输出,每组输出占一行

示例1

输入

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

输出

```
5
5 4 3
```

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

int cmp(int *a,int *b)
{
    return *b-*a;
}
int main()
{
    int n,m,a[100000];
    while(scanf("%d%d",&n,&m)!=EOF && n*m!=0)
    {
        if(m>n)m=n;
        for(int i=0;i<n;i++)scanf("%d",&a[i]);
        qsort(a,n,4,cmp);
        for(int i=0;i<m;i++)printf("%d ",a[i]);
        printf("\n");
    }
}
```

题目七十四：最小代价路径

> 题目描述

有一个6*6的棋盘，每个棋盘上都有一个数值，现在又一个起始位置和终止位置，请找出一个从起始位置到终止位置代价最小的路径：

1、只能沿上下左右四个方向移动

2、总代价是没走一步的代价之和

3、每步（从a,b到c,d）的代价是c,d上的值与其在a,b上的状态的乘积

4、初始状态为1每走一步，状态按如下公式变化：（走这步的代价%4）+1。

> 输入描述：

每组数据一开始为6*6的矩阵，矩阵的值为大于等于1小于等于10的值，然后四个整数表示起始坐标和终止坐标。

> **输出描述：**

输出最小代价。

示例1

输入

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

输出

```
23
```

```
#include <iostream>
using namespace std;
int total = 0x7fffffff;
int X, Y; // 终点
#define N 6
int p[N][N]; // 棋盘
bool flag[N][N];
int pos[4][2] = {{-1,0},{1,0},{0,1},{0,-1}};
void DFS(int s, int c, int x, int y)// 状态s，代价c，从<x,y>起
{
    if (flag[x][y]) return;
    if (c > total) return;// 这个剪枝剪得漂亮

    if (x==X && y==Y) {
        if (total == -1) total = c;
        else total = total<c?total:c;
        return;
    }
    if (x < 0 || y < 0 || x >= N || y >= N) return;

    flag[x][y] = true;// 从<x,y>出发，往下深搜不再允许使用<x,y>这个点
    for (int i = 0; i < 4; i ++) {
        int xx = x+pos[i][0], yy = y+pos[i][1];
        int cc = s * p[xx][yy];
        DFS(cc%4+1,c+cc,xx,yy);
    }
    flag[x][y] = false;// 向上层回溯，故而<x,y>还有可能被用到
```

```
    return;
}
int main(void)
{

    for (int i = 0; i < N; i ++)
        for (int j = 0; j < N; j ++)
            cin >> p[i][j];
    int x,y;
    cin >> x >> y >> X >> Y;

    DFS(1,0,x,y);

    cout << total << endl;
    return 0;
}
```

题目七十五：最小子矩阵面积

> 题目描述

一个N*M的矩阵，找出这个矩阵中所有元素的和不小于K的面积最小的子矩阵（矩阵中元素个数为矩阵面积）

> 输入描述：

每个案例第一行三个正整数N,M<=100，表示矩阵大小，和一个整数K
接下来N行，每行M个数，表示矩阵每个元素的值

> 输出描述：

输出最小面积的值。如果出现任意矩阵的和都小于K，直接输出-1。

示例1

输入

4 4 10

```
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

输出

```
1
```

```
#include <stdio.h>
#include <limits.h>
#define N 101
#define M 101

int martix[N][M];//输入的矩阵
int sum[M];//相邻的几行合并而成的一维数组
int m, n, k;

int Shortest(int sum[M])
{//求一维数组里面和不小于k的最短连续序列长度
    int from=0, to=0;
    int len=INT_MAX;
    int x=0;//from到to的序列和
    while(to<m)
    {
        while(x<k&&to<m)//序列和小于k那终点后移
        {
            x+=sum[to];
            to++;
        }
        while(x>=k&&from<to-1)//序列和不小于k那起点后移
        {
            if(to-from<len) len=to-from;//这是更短的序列，记下来
            x-=sum[from];
            from++;
        }
        if(from+1==to) return 1;//追尾了
    }
    return len;
}

int main()
{
    while(scanf("%d%d%d", &n, &m, &k)!=EOF)
    {
        for(int i=0; i<n; i++)//输入
        {
            for(int j=0; j<m; j++)
            {
                scanf("%d", &martix[i][j]);
            }
        }
        int area=INT_MAX;//满足条件的最小面积，先假设是无穷大
```

```
        for(int i=0; i<n; i++)//从i行开始合并
        {
            for(int j=0; j<m; j++) sum[j]=0;//清空sum数组
            for(int k=i; k<n; k++)//合并从i行到k行
            {
                for(int j=0; j<m; j++) sum[j]+=martix[k][j];
                int len=Shortest(sum);
                if(len!=INT_MAX)
                {//如果面积比已有的最小面积还小，那就修改最小面积
                    if(len*(k-i+1)<area) area=len*(k-i+1);
                }
            }
        }
        if(area==INT_MAX) printf("-1\n");
        else printf("%d\n", area);
    }
    return 0;
}
```

题目七十六：k-th prime number

> 题目描述

Output the k-th prime number.

> 输入描述：

k≤10000

> 输出描述：

The k-th prime number.

示例1

输入

```
3
7
```

输出

```
5
17
```

```cpp
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <vector>
using namespace std;

const int MAX = 1e5;
bool arr[MAX];
vector<int> prime;

void initial()
{
    for (int i = 2; i < MAX; i++)
    {
        arr[i] = true; //把区间范围内的数都标记为质数
    }
    for (int i = 2; i < MAX; i++)
    {
        if (!arr[i])
        {
            continue;
        }
        prime.push_back(i);
        for (int j = 2 * i; j < MAX; j += i)
        {
            arr[j] = false;
        }
    }
}

int main()
{
    initial();
    int n;
    while (cin >> n)
    {
        cout << prime[n - 1] << endl;
    }
    return EXIT_SUCCESS;
}
```

题目七十七：k-th prime number

> 题目描述

欧拉回路是指不令笔离开纸面，可画过图中每条边仅一次，且可以回到起点的一条回路。现给定一个图，问是否存在欧拉回路？

> **输入描述：**

测试输入包含若干测试用例。每个测试用例的第1行给出两个正整数，分别是节点数N（1 < N < 1000）和边数M；随后的M行对应M条边，每行给出一对正整数，分别是该条边直接连通的两个节点的编号（节点从1到N编号）。当N为0时输入结束。

> **输出描述：**

每个测试用例的输出占一行，若欧拉回路存在则输出1，否则输出0。

示例1

输入

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

输出

```
1
0
```

```
//确定无向图欧拉回路的充要条件：除孤立节点外，其它节点满足 1.连通
//2.度为偶数

#include <cstdio>
#include <algorithm>

using namespace std;

int father[1001];
//并查集找父亲的操作
int findFather(int x){
    while(x!=father[x]){
        x = father[x];
    }
```

```
    return x;
}
//并查集合并的操作
void Union(int a, int b){
    int af = findFather(a);
    int bf = findFather(b);
    father[bf] = af;
}

int main(){
    int n,m;
    while(scanf("%d%d",&n,&m)!=EOF){
        int d[1001]; //节点度
        fill(d,d+1001,0);
        for(int i=0;i<=n;i++) father[i]=i; //初始化father数组
        for(int i=0;i<m;i++){
            int a,b;
            scanf("%d%d",&a,&b);
            d[a]++;
            d[b]++;
            Union(a,b);
        }
        int tmp=0;
        for(int i=1;i<=n;i++){
            //有奇数度，应打印0
            if(d[i]%2!=0){
                tmp++;
                break;
            }
        }
        if(tmp>0){
            printf("0\n");
            continue;
        }
        int t = 1;
        for(int i=0;i<=n;i++){  //寻找一个非孤立节点，存入t
            if(d[i]!=0){
                t = i;
                break;
            }
        }
        int f = findFather(t);
        bool flag = false;
        for(int i=2;i<=n;i++){
            //既不是孤立节点，也不连通，应打印0
            if(findFather(i)!=f && findFather(i)!=i){
                flag = true;
                break;
            }
```

```
        }
        if(flag){
            printf("0\n");
            continue;
        }
        if(n!=0){
            printf("1\n");
        }

    }
    return 0;
}
```

题目七十八：奥运排序问题

> 题目描述

按要求，给国家进行排名。

> 输入描述：

有多组数据。
第一行给出国家数N，要求排名的国家数M，国家号从0到N-1。
第二行开始的N行给定国家或地区的奥运金牌数，奖牌数，人口数（百万）。
接下来一行给出M个国家号。

> 输出描述：

排序有4种方式: 金牌总数 奖牌总数 金牌人口比例 奖牌人口比例 对每个国家给出最佳排名排名方式 和 最终排名
格式为: 排名:排名方式
如果有相同的最终排名，则输出排名方式最小的那种排名，对于排名方式，金牌总数 < 奖牌总数 < 金牌人口比例 < 奖牌人口比例
如果有并列排名的情况，即如果出现金牌总数为 100,90,90,80.则排名为1,2,2,4.
每组数据后加一个空行。

示例1

输入

```
4 4
4 8 1
6 6 2
4 8 2
```

```
2 12 4
0 1 2 3
4 2
8 10 1
8 11 2
8 12 3
8 13 4
0 3
```

输出

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

1:1
1:1
```

```
#include <stdio.h>
#define N 10
int main() {
    int n,m;
    while(scanf("%d%d",&n,&m)!=EOF) {
        int temp[N][3];
        for(int i=0; i<n; i++) {
            scanf("%d%d%d",&temp[i][0],&temp[i][1],&temp[i][2]);
        }
        float data[N][4];
        int rank[N][4]= {0};
        for(int i=0; i<m; i++) {
            int country_id;
            scanf("%d",&country_id);
            data[i][0]=temp[country_id][0];
            data[i][1]=temp[country_id][1];
            data[i][2]=temp[country_id][0]?data[i][0]/temp[country_id][2]:
0;
            data[i][3]=temp[country_id][1]?data[i][1]/temp[country_id][2]:
0;
        }
        for(int i=0; i<m; i++) {
            for(int j=0; j<m; j++) {
                for(int k=0; k<4; k++) {
                    if(data[j][k]>data[i][k]) {
                        rank[i][k]++;
                    }
                }
            }
```

```
        }
        for(int i=0; i<m; i++) {
            int min=0;
            for(int j=1; j<4; j++) {
                if(rank[i][j]<rank[i][min]) {
                    min=j;
                }
            }
            printf("%d:%d\n",rank[i][min]+1,min+1);
        }
        printf("\n");
    }
}
```

题目七十九：关门人和开门人

题目描述

每天第一个到机房的人要把门打开，最后一个离开的人要把门关好。现有一堆杂乱的机房签到、签离记录，请根据记录找出当天开门和关门的人。

输入描述：

每天的记录在第一行给出记录的条目数M（M > 0），下面是M行，每行的格式为

证件号码 签到时间 签离时间

其中时间按“小时:分钟:秒钟”（各占2位）给出，证件号码是长度不超过15的字符串。

输出描述：

对每一天的记录输出1行，即当天开门和关门人的证件号码，中间用1空格分隔。

注意：在裁判的标准测试输入中，所有记录保证完整，每个人的签到时间在签离时间之前，且没有多人同时签到或者签离的情况。

示例1

输入

```
3
CS301111 15:30:28 17:00:10
SC3021234 08:00:00 11:25:25
CS301133 21:45:00 21:58:40
```

输出

```
SC3021234 CS301133
```

```
#include<iostream>
#include<cstring>
using namespace std;
struct people
{
    char number[16];//证件号
    char time_start[10];//签到时间
    char time_end[10];//离开时间
};
int main()
{
    int n;
    people peo[100];
    char open_time[10], close_time[10];
    cin >> n;
    for (int i = 0; i < n; i++)//输入信息
        cin >> peo[i].number >> peo[i].time_start >> peo[i].time_end;
    strcpy(open_time, peo[0].time_start);
    strcpy(close_time, peo[0].time_end);
    for (int i = 1; i < n; i++)//寻找时间
    {
        if(strcmp(open_time,peo[i].time_start)>0)
            strcpy(open_time, peo[i].time_start);
        if(strcmp(close_time, peo[i].time_end)<0)
            strcpy(close_time, peo[i].time_end);
    }
    for (int i = 0; i < n; i++)//输出开门
    {
        if (strcmp(open_time, peo[i].time_start) == 0)
            cout << peo[i].number << " ";
    }
    for (int i = 0; i < n; i++)//输出关门
    {
        if (strcmp(close_time, peo[i].time_end) == 0)
            cout << peo[i].number <<endl;
    }
    return 0;
}
```

题目八十：调整方阵

> 题目描述

输入一个N（N<=10）阶方阵，按照如下方式调整方阵：1.将第一列中最大数所在的行与第一行对调。2.将第二列中从第二行到第N行最大数所在的行与第二行对调。依此类推... N-1.将第N-1列中从第N-1行到第N行最大数所在的行与第N-1行对调。N.输出这个方阵

> 输入描述：

包含多组测试数据,每组测试数据第一行为一个整数N,表示方阵的阶数。
接下来输入这个N阶方阵。

> 输出描述：

调整后的方阵

示例1

输入

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

输出

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

```
#include<iostream>
using namespace std;
#define max_len 11
int a[max_len][max_len];

void transfer(int a[max_len][max_len],int n,int i,int j)
{
        for(int l = 0;l < n;l++)
        {
            int temp = a[i][l];
```

```
            a[i][l] = a[j][l];
            a[j][l] = temp;
        }
}

int main(void)
{
    int n;
    int b[max_len];

    while(cin >> n)
    {
        for(int i = 0;i < n;i++)
            for(int j = 0;j < n;j++)
            {
                int x;
                cin >> x;
                a[i][j] = x;
            }

        for(int i = 0;i < n;i++)
        {
            int max = i;
            for(int j = i;j < n;j++)
            {
                if(a[max][i] < a[j][i])
                    max = j;//本列中最大数所在行号
            }
            transfer(a, n, i, max);
        }

        for(int i = 0;i < n;i++)
        {
            for(int j = 0;j < n;j++)
                cout << a[i][j] << ' ';
            cout << endl;
        }
    }
    return 0;
}

```