**7-1 输出 CQU**

给定一个长度不超过 10000 的、仅由英文字母和数字构成的字符串。请将字符重新调整顺序，按 *CQUCQU....*这样的顺序输出，并忽略其它字符。当然，三种字符（不区分大小写）的个数不一定是一样多的，若某种字符已经输出完，则余下的字符仍按 CQU 的顺序打印，直到所有字符都被输出。

输入格式：

输入在一行中给出一个长度不超过 10000 的、仅由英文字母和数字构成的非空字符串。

输出格式：

在一行中按题目要求输出排序后的字符串。题目保证输出非空。

输入样例：

pcTclnqGloRguLQrtLhugljkLhGFauPewSKgt

输出样例：

CQUCQUU

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int c=0,q=0,u=0;
    string s;
    cin>>s;
    for(int i=0;i<int(s.size());i++){
    if(s[i]=='c'||s[i]=='C')
        c++;
    else if(s[i]=='q'||s[i]=='Q')
        q++;
    else if(s[i]=='u'||s[i]=='U')
        u++;
    }
    while(c||q||u){
        if(c>0){
            cout<<"C";
            c--;
            }
        if(q>0){
            cout<<"Q";
            q--;
        }
        if(u>0){
            cout<<"U";
            u--;
        }
    }
    return 0;
}
```

**7-2 计算 BMI**

（健康应用程序：计算 BMI）身体质量指数（BMI）是以体重衡量健康程度的一种指数。以千克为单位的体重除以以米为单位的身高的平方就可以计算它的值。编写一个程序，用户输入以千克为单位的体重和以米为单位的身高。

输入格式:

输入体重，身高

输出格式:

输出 BMI 值，保留 4 位小数

输入样例:

在这里给出一组输入。例如：

43.3

1.27

输出样例:

在这里给出相应的输出。例如：

BMI is 26.8461

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    double a,b,c;
    cin>>a>>b;
    c=a/(b*b);
    cout<<"BMI is "<<setiosflags(ios::fixed)<<setprecision(4)<<c;
    return 0;
}
```

**7-3 人月神话**

人月是软件开发统计工作量的单位。某些人认为增加程序员的数量，就能提升软件开发的效率。对此 Fred Brooks 曾经讽刺道：*Nine people can't make a baby in a month.*

本题请你直接在屏幕上输出这句话。

输入格式:

本题没有输入。

输出格式:

在一行中输出 Nine people can't make a baby in a month.

输入样例:

无

输出样例:

Nine people can't make a baby in a month.

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    cout<<"Nine people can't make a baby in a month.";
    return 0;
}
```

**7-4 跟着小明画三角形**

小明特别喜欢画画，后来喜欢上了 C++程序设计，编写了很简单的代码：在屏幕上画一个直角三角形，两条直角边包含相同的字符数。现在跟他一起画吧！

输入格式:

输入在一行中给出直角三角形的直角边长 N（3≤N≤21）和组成边的某种字符 C，间隔一个空格。

输出格式:

输出由给定字符 C 画出的直角三角形。

输入样例:

例如：

5 *

输出样例:

例如：

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

```
#include<bits/stdc++.h>
#include<string>
using namespace std;
int main(){
    int a;
    char b;
    cin>>a>>b;
    for(int i=1;i<a;i++)
    {
        for(int j=0;j<i;j++){
            cout<<b;
        }
    cout<<endl;
    }
    for(int i=0;i<a;i++)
    {
    cout<<b;
    }
    return 0;
}
```

**7-5 找出二维数组中大于平均值的数的位置**

编写程序，将数据输入一个 n*n 的二维数组中，输出大于平均数的数和他们的位置，可能不止一个。所有输入数据为整数。如果没有大于平均数的数，则输出 0.

输入格式:

第 1 行，输入 n，表示二维数组的行列数。n 在 2-20 之间。

第 2 行到 n+1 行，输入 n 行数据，每行 n 个数据，每行的 n 个数据之间用一个空格分割。

输出格式:

每行输出大于平均数的数和他的位置，先输出数，再输出行索引和列索引，用一个空格分割。

输入样例:

在这里给出一组输入。例如：

```
4
25 93 78 13
12 26 88 93
75 93 22 32
56 44 36 88
```

输出样例:

在这里给出相应的输出。例如：

```
93 0 1
78 0 2
88 1 2
93 1 3
75 2 0
93 2 1
56 3 0
88 3 3
```

```cpp
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin>>n;
    int arr[n][n]={0};
    int i=0;
    int j=0;
    double avg;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cin>>arr[i][j];
            avg += arr[i][j];
        }
    }
    avg /= n;
    avg /= n;
```

```
19.         double m=avg;
20.             for(i=0;i<n;i++){
21.                 for(j=0;j<n;j++){
22.                     if(m<=arr[i][j]){
23.                         m=arr[i][j];
24.                     }
25.             }
26.         }
27.         if(m==avg)
28.             cout<<0;
29.         else{
30.             for(i=0;i<n;i++){
31.                 for(j=0;j<n;j++){
32.                     if(avg<arr[i][j])
33.                         cout<<arr[i][j]<<" "<<i<<" "<<j<<endl;
34.             }
35.         }
36.         }
37.         return 0;
38.     }
```

# 画图题

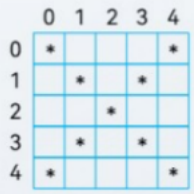
**Q3**

请你打印“X”，X的正斜线或反斜线的长度N由输入决定，X的组成符号也由输入决定。

**输入：** 5 *

**输出：**

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




```cpp
# include<iostream>
using namespace std;
int main(){
int n;
char c;
cin >> n >> c;
cout << "---------picture-----------" << endl;
for(int i = 0; i < n; i++){
for(int j = 0; j < n; j++){
if(i == j || j == n - 1 -i)
cout << c;
else
cout << ' ';
}
cout << endl;
}
}
```

**7-1 小写字母转换为大写字母**

编写一个程序，将输入字符串中的小写字母转换成大写字母后进行输出。对于字符串中的非小写字母，不做任何转换操作。

输入格式:

输入一个字符串，该字符串可以包含大写字母、小写字母、数字和其它字符（如空格），但不包含换行回车符。

输出格式:

将输入字符串中的小写字母转换为大写字母，而非小写字母保持不变。输出转换后的字符串。

输入样例:

在这里给出一组输入。例如：

Hello, C++!

输出样例:

在这里给出相应的输出。例如：

HELLO, C++!

```cpp
#include<bits/stdc++.h>
#include<string>
#include<cstring>
using namespace std;
int main()
{
    char a[1001];
    cin.get(a,1001);
    int s;
    s=strlen(a);
    for(int i=0;i<s;i++)
    {
       if(a[i]>='a'&&a[i]<='z')
       {
           a[i]=a[i]-32;
       }
    }
    cout<<a<<endl;
}
```

**7-2 位数和**

输入一个 4 位整数，求各个位上的数字之和。

输入格式:

一个四位数的整数。

输出格式:

一个整数，表示各位之和。

输入样例:

1234

输出样例:

10

样例解释：1+2+3+4=10

```cpp
#include<iostream>
using namespace std;
int main()
{
    int a,b,sum=0;
    cin>>a;
    while(a){
        b=a%10;
        sum+=b;
        a/=10;
    }
    cout<<sum;
    return 0;
```

```
}
```

**7-3 算时间**

夏老板在跑步机上锻炼身体。他低头一看，跑步机上记录的跑步时间是以秒为单位，屏幕上显示夏老板跑步的总时间是x秒。

夏老板想用"时分秒"的形式来表示跑步时间，他向你提问：x秒是多少个小时 多少分钟 多少秒？

输入格式:

一行，一个整数 x（0≤x≤2* 1012）。

输出格式:

一行，三个整数 a,b,c 分别表示时分秒，其间以":"作为间隔。

输入样例 1:

8000

输出样例 1:

2:13:20

输入样例 2:

1234

输出样例 2:

0:20:34

样例 1 说明：8000 秒等于 2 小时 13 分 20 秒。

```
#include<iostream>
using namespace std;
int main()
{
    long long sec;
    int a,b,c;
    cin>>sec;
    if(sec>=0&&sec<=2*10^12)
    a=sec%60;
    b=sec/60%60;
    c=sec/60/60;
    cout<<c<<":"<<b<<":"<<a;
    return 0;
}
```

**7-1 混合类型数据格式化输入**

本题要求编写程序，顺序读入浮点数 1、整数、字符、浮点数 2，再按照字符、整数、浮点数 1、浮点数 2 的顺序输出。

输入格式:

输入在一行中顺序给出浮点数 1、整数、字符、浮点数 2，其间以 1 个空格分隔。

输出格式:

在一行中按照字符、整数、浮点数 1、浮点数 2 的顺序输出，其中浮点数保留小数点后 2 位。

输入样例:

2.12 88 c 4.7

输出样例：

c 88 2.12 4.70

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    float f1;
    int n;
    char ch;
    float f2;
    cin>>f1>>n>>ch>>f2;
    cout<<ch<<" "<<n<<" "<<f1<<" "<<setiosflags(ios::fixed)<<setprecision(2)<<f2;
    return 0;
}
```

**7-4 后天**

如果今天是星期三，后天就是星期五；如果今天是星期六，后天就是星期一。我们用数字 1 到 7 对应星期一到星期日。给定某一天，请你输出那天的"后天"是星期几。

输入格式：

输入第一行给出一个正整数 D（1 ≤ D ≤ 7），代表星期里的某一天。

输出格式：

在一行中输出 D 天的后天是星期几。

输入样例：

3

输出样例：

5

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    int a,b;
    cin>>a;
    b=(a+2)%7;
    if(b==0){
        cout<<b+7<<endl;
    }
    else{
        cout<<b;
```

```
14.         }
15.         return 0;
16.     }
```

**7-5 三个整数的统计**

输入 3 个不超过 1000 的正整数，计算这 3 个整数的和、平均值、乘积、最小值和最大值。

输入格式:

输入在一行中给出 3 个整数。

输出格式:

分别输出 3 个整数的和、平均值、乘积、最小值和最大值。每个结果各占一行。

注意：对于平均值，保留 1 位小数。

输入样例:

13 27 14

输出样例:

Sum is 54

Average is 18.0

Product is 4914

Smallest is 13

Largest is 27

```
1.      #include<iostream>
2.      #include<iomanip>
3.      using namespace std;
4.      int main()
5.      {
6.          int a,b,c,d,e;
7.          double i;
8.          long long f;
9.          cin>>a>>b>>c;
10.         f=a*b*c;
11.         i=1.0*(a+b+c)/3;
12.         d=a<b?a:b;
13.         d=d<c?d:c;
14.         e=a>b?a:b;
15.         e=e>c?e:c;
16.         cout<<"Sum is "<<a+b+c<<endl;
17.          cout<<"Average is "<<setiosflags(ios::fixed)<<setprecision(1)<<i<<endl;
18.         cout<<"Product is "<<f<<endl;
19.         cout<<"Smallest is "<<d<<endl;
20.         cout<<"Largest is "<<e;
21.     return 0;
```

```
}
```

**7-2 n 个正整数的统计**

输入 n 个不超过 1000 的正整数（n<100），计算这 n 个非负整数的和、平均值、乘积、最小值和最大值。

输入格式:

依次输入 n 个非负整数，每个整数一行，当输入"-1"时结束。

输出格式:

分别输出这 n 个整数的和、平均值、乘积、最小值和最大值。每个结果各占一行。其中，平均值保留 1 位小数（四舍五入）。

输入样例:

13

27

14

-1

输出样例:

Sum is 54

Average is 18.0

Product is 4914

Smallest is 13

Largest is 27

提示：请注意，每次输入正整数的数量 n 是不固定的。

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int n=1;
    int sum = 0, product = 1,zuixiao=1000,zuida=0;
    double average;
    int x;
    for (;; n++)
    {
        cin >> x;
        if(x==-1)
            break;
        if(x!=-1)
        {
        sum += x;
        average= 1.0*sum / n;
        product *= x;
        zuixiao= min(zuixiao, x);
        zuida = max(zuida, x);
        }
```

```
        }
        cout<<"Sum is "<<sum<<endl;
        cout<<"Average is "<<setiosflags(ios::fixed)<<setprecision(1)<<average<<endl;
        cout<<"Product is "<<product<<endl;
        cout<<"Smallest is "<<zuixiao<<endl;
        cout<<"Largest is "<<zuida;
        return 0;

}
```

**7-3 打印指定大小的棋盘格**

考虑尽量用最少的语句条数，打印出指定大小的棋盘格图案。

输入格式:

输入正整数 N（N<20）。

输出格式:

输出由 N×N 个“*”号组成的棋盘格。注意相邻两个“*”号之间有一个空格间隔，奇数行最后有一个空格。

输入样例:

在这里给出一组输入。例如：

```
3
```

输出样例:

在这里给出相应的输出。例如：

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

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i % 2 == 0)
            cout << "* ";
            if (i % 2 == 1)
            cout <<" *";
        }
        cout << endl;
    }
    return 0;
}
```

**7-4 打印正整数的各位数字**

本题目要求读入 1 个整数 N（0≤N<10^8），然后打印输出该整数各位数字，相邻位数字之间间隔 3 个空格。

输入格式:

输入整数 N。

输出格式:

输出 N 的各位数字，相邻位数字之间间隔 3 个空格。

输入样例:

在这里给出一组输入。例如：

```
123456
```

输出样例:

在这里给出相应的输出。例如：

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

```cpp
#include <iostream>
using namespace std;
int a[100];
int main()
{
    int n, i = 0;
    cin >> n;
    if (n == 0) { cout << 0; }
    while (n != 0)
    {
        a[i++] = (n % 10);
        n /= 10;

    }
    for (int j = i - 1; j >= 0; j--)
    {
        if (a[j] < 0 && j != i-1)
            cout << a[j] * -1;
        else cout << a[j];
        if (j != 0) { cout << "   "; }
    }
```

```
22.         return 0;
23.     }
```

**7-5 估算数学常数 e 的值**

数学常数 e 是数学中一个重要的常数，它是一个无限不循环小数，其值约为 2.71828。其计算公式为：

e=1+1!1 +2!1 +3!1 +⋯ +i!1

这里输入需要的 e 值精度 n（即参与计算 e 值的项 i!1 都不小于 10n1 ），3≤n≤10。编程计算输出相应精度的 e 值（保留 n+1 位小数）。

浮点数的数据类型请使用 double。

输入格式:

输入需要的 e 值精度 n。

输出格式:

输出相应精度的 e 值，保留 n+1 位小数（四舍五入）。

输入样例:

3

输出样例:

2.7181

```
1.  #include <iostream>
2.  #include<iomanip>
3.  #include <cmath>
4.  using namespace std;
5.  int main()
6.  {
7.      int n;
8.      cin>>n;
9.      double e;
10.     double factorial = 1;
11.     for(int i=1;;i++)
12.     {
13.         factorial *= i;
14.         if(1/factorial < 1/pow(10,n))
15.             break;
16.         e += 1/factorial;
17.     }
18.         cout<<setiosflags(ios::fixed)<<setprecision(n+1)<<e+1<<endl;
19.     return 0;
20. }
```

**7-1 水仙花数**

水仙花数是指一个 N 位正整数（N≥3），它的每个位上的数字的 N 次幂之和等于它本身。例如：153=13+53+33。 本题要求编写程序,计算所有 N 位水仙花数。

输入格式:

输入在一行中给出一个正整数 N（3≤N≤7）。

输出格式:

按递增顺序输出所有 N 位水仙花数，每个数字占一行。

输入样例:

3

输出样例:

153

370

371

407

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;

int power(int a,int b){
    int res=1;
    while(b --)res *= a;
    return res;
}

int main()
{
    int i;
    cin>>i;
    int n1, n2;
    for(n1=power(10,i-1); n1<power(10,i); n1++)
    {
        n2 = 0;
        int num = 0;
        int temp = n1;
        while(temp)
        {
            if(n2>n1) break;
            num = temp % 10;
            n2 += power(num, i);
            temp = temp / 10;
        }
        if(n1 == n2)
            cout << n1 << endl;
    }
    return 0;
```

```
}
```

**7-2 猜数字游戏**

猜数字游戏是令游戏机随机产生一个 100 以内的正整数，用户输入一个数对其进行猜测，需要你编写程序自动对其与随机产生的被猜数进行比较，并提示大了（“Too big”），还是小了（“Too small”），相等表示猜到了。如果猜到，则结束程序。程序还要求统计猜的次数，如果 1 次猜出该数，提示“Bingo!”；如果 3 次以内猜到该数，则提示“Lucky You!”；如果超过 3 次但是在 N（>3）次以内（包括第 N 次）猜到该数，则提示“Good Guess!”；如果超过 N 次都没有猜到，则提示“Game Over”，并结束程序。如果在到达 N 次之前，用户输入了一个负数，也输出“Game Over”，并结束程序。

输入格式:

输入第一行中给出两个不超过 100 的正整数，分别是游戏机产生的随机数、以及猜测的最大次数 N。最后每行给出一个用户的输入，直到出现负数为止。

输出格式:

在一行中输出每次猜测相应的结果，直到输出猜对的结果或“Game Over”则结束。

输入样例:

```
58 4
70
50
56
58
60
-2
```

输出样例:

```
Too big
Too small
Too small
Good Guess!
```

```
#include<iostream>
#include<cstdlib>
#include<cmath>
using namespace std;
int main()
{
```

```
7.      int result,N;
8.      int n=0;
9.      int input;
10.     cin>>result>>N;
11.     do
12.     {
13.         cin>>input;
14.         n++;
15.         if(input<0)
16.         {
17.             cout<<"Game Over"<<endl;
18.             break;
19.         }
20.         if(input<result&&n<=N)
21.         {
22.             cout<<"Too small"<<endl;
23.         }
24.         else if(input>result&&n<=N)
25.         {
26.             cout<<"Too big"<<endl;
27.         }
28.         else if(input==result&&n==1)
29.         {
30.             cout<<"Bingo!"<<endl;
31.             break;
32.         }
33.         else if(input==result&&n>1&&n<=3)
34.         {
35.             cout<<"Lucky You!"<<endl;
36.             break;
37.         }
38.         else if(input==result&&n>3&&n<=N)
39.         {
40.             cout<<"Good Guess!"<<endl;
41.             break;
42.         }
43.         else if(n>N)
44.         {
45.             cout<<"Game Over"<<endl;
46.             break;
47.         }
48.         else
49.         {
50.             cout<<"Game Over"<<endl;
```

```
                break;
            }
        }while(n<=N);
        return 0;
    }
```

**7-3 狡猾的财主**

老管家是一个聪明能干的人。他为财主工作了整整 10 年，财主为了让自己账目更加清楚。要求管家每天记 k 次账，由于管家聪明能干，因而管家总是让财主十分满意。但是由于一些人的挑拨，财主还是对管家产生了怀疑。于是他决定用一种特别的方法来判断管家的忠诚，他把每次的账目按 1，2，3...编号，然后不定时的问管家问题，问题是这样的：在 a 到 b 号账中最少的一笔是多少？为了让管家没时间作假他总是一次问多个问题。

输入格式:

输入中第一行有两个数 m,n 表示有 m(m<=100000)笔账,n 表示有 n 个问题，n<=100000。

第二行为 m 个数,分别是账目的钱数

后面 n 行分别是 n 个问题,每行有 2 个数字说明开始结束的账目编号。

输出格式:

输出文件中为每个问题的答案。具体查看样例。

输入样例:

在这里给出一组输入。例如：

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

输出样例:

在这里给出相应的输出。例如：

```
2
3
1
```

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int m,n;
    int a[100001]={0};
```

```
        cin>>m>>n;
        for(int i=0;i<m;i++)
            cin>>a[i];
        for(int i=0;i<n;i++)
        {
            int left=0,right=0;

            cin>>left>>right;
            int s=a[left-1];
            for(int i=left;i<right;i++){
                s=min(s,a[i]);
            }

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

**7-1 交换最小值和最大值**

本题要求编写程序，先将输入的一系列整数中的最小值与第一个数交换，然后将最大值与最后一个数交换，最后输出交换后的序列。

注意：题目保证最大和最小值都是唯一的。

输入格式：

输入在第一行中给出一个正整数 N（≤10），第二行给出 N 个整数，数字间以空格分隔。

输出格式：

在一行中顺序输出交换后的序列，每个整数后跟一个空格。

输入样例：

```
5
8 2 5 1 4
```

输出样例：

```
1 2 5 4 8
```

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstdio>
    int n;
    cin>>n;
    int a[n]={0};
    int minid=0;
    for(int i=0;i<n;i++)
    {
        cin>>a[i];
```

```
            if(a[i]<a[minid])minid=i;
        }
        swap(a[minid],a[0]);
        int maxid=0;
        for(int i=0;i<n;i++)
        {
            if(a[i]>a[maxid])maxid=i;
        }
        swap(a[maxid],a[n-1]);
            for(int i=0;i<n;i++){
                cout<<a[i]<<" ";
            }
            cout<<endl;
        return 0;
    }
```

**7-2 求集合数据的均方差**

设计函数求 N 个给定整数的均方差。若将 N 个数 A[ ] 的平均值记为 Avg，则均方差计算公式为：

[(A1 – Avg)2+(A2 – Avg)2+… +(AN – Avg)2]/N 这一坨开根号 。

输入格式：

输入首先在第一行给出一个正整数 N（≤104），随后一行给出 N 个正整数。所有数字都不超过 1000，同行数字以空格分隔。

输出格式：

输出这 N 个数的均方差，要求固定精度输出小数点后 5 位。

输入样例 1：

10

6 3 7 1 4 8 2 9 11 5

输出样例 1：

3.03974

输入样例 2：

1

2

输出样例 2：

0.00000

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<iomanip>
using namespace std;
int main()
{
    int n;
    cin>>n;
```

```
        int a[n]={0};
        double avg=0.0;
        for(int i=0;i<n;i++){
            cin>>a[i];
            avg+=a[i];
        }
        avg/=n;
        double var=0.0;
        for(int i=0;i<n;i++)
            var+=(a[i]-avg)*(a[i]-avg);
        var=sqrt(var/n);
        cout<<setiosflags(ios::fixed)<<setprecision(5)<<var;
    }
```

**7-3 打印杨辉三角**

本题要求按照规定格式打印前 *N*行杨辉三角。

输入格式：

输入在一行中给出 N（1≤N≤10）。

输出格式：

以正三角形的格式输出前 N 行杨辉三角。每个数字占固定 4 位。

输入样例：

```
6
```

输出样例：

```
   1
   1   1
   1   2   1
   1   3   3   1
   1   4   6   4   1
   1   5  10  10   5   1
```

```
    #include<iostream>
    #include<iomanip>
    using namespace std;
    int main()
    {
        int n=0,i,j;
        cin>>n;
        int a[n][n];
        for(i=0;i<n;i++)
        {
            a[i][0]=1;
            a[i][i]=1;
```

```
        }
        for(i=2;i<n;i++)
            for(j=1;j<i;j++)
                a[i][j]=a[i-1][j-1]+a[i-1][j];
        for(i=0;i<n;i++)
        {
            for(j=0;j<=n-i-2;j++){
                cout<<" ";
            }
            for(j=0;j<=i;j++){
                cout<<setw(4)<<right<<a[i][j];
            }
            cout<<endl;
        }
        return 0;
    }
```

**7-4 刮刮彩票**

“刮刮彩票”是一款网络游戏里面的一个小游戏。如图所示：


每次游戏玩家会拿到一张彩票，上面会有 9 个数字，分别为数字 1 到数字 9，数字各不重复，并以 3×3 的“九宫格”形式排布在彩票上。

在游戏开始时能看见一个位置上的数字，其他位置上的数字均不可见。你可以选择三个位置的数字刮开，这样玩家就能看见四个位置上的数字了。最后玩家再从 3 横、3 竖、2 斜共 8 个方向中挑选一个方向，方向上三个数字的和可根据下列表格进行兑奖，获得对应数额的金币。

| 数字合计 | 获得金币 | 数字合计 | 获得金币 |
| --- | --- | --- | --- |
| 6 | 10, 000 | 16 | 72 |
| 7 | 36 | 17 | 180 |

| 数字合计 | 获得金币 | 数字合计 | 获得金币 |
| --- | --- | --- | --- |
| 8 | 720 | 18 | 119 |
| 9 | 360 | 19 | 36 |
| 10 | 80 | 20 | 306 |
| 11 | 252 | 21 | 1,080 |
| 12 | 108 | 22 | 144 |
| 13 | 72 | 23 | 1,800 |
| 14 | 54 | 24 | 3,600 |
| 15 | 180 | | |

现在请你写出一个模拟程序，模拟玩家的游戏过程。

输入格式:

输入第一部分给出一张合法的彩票，即用 3 行 3 列给出 0 至 9 的数字。0 表示的是这个位置上的数字初始时就能看见了，而不是彩票上的数字为 0。

第二部给出玩家刮开的三个位置，分为三行，每行按格式 x y 给出玩家刮开的位置的行号和列号（题目中定义左上角的位置为第 1 行、第 1 列。）。数据保证玩家不会重复刮开已刮开的数字。

最后一部分给出玩家选择的方向，即一个整数： 1 至 3 表示选择横向的第一行、第二行、第三行，4 至 6 表示纵向的第一列、第二列、第三列，7、8 分别表示左上到右下的主对角线和右上到左下的副对角线。

输出格式:

对于每一个刮开的操作，在一行中输出玩家能看到的数字。最后对于选择的方向，在一行中输出玩家获得的金币数量。

输入样例:

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

输出样例:

```
1
5
6
180
```

```
#include <iostream>
using namespace std;
int main()
{
```

```
5.      int i,j,k,m,row,col,a[4][4],x,y,n,sum=0,money;
6.      for(i=1;i<=3;i++)
7.      {
8.          for(j=1;j<=3;j++)
9.          {
10.             cin>>a[i][j];
11.             if(a[i][j]==0)
12.             {
13.                 row=i;
14.                 col=j;
15.             }
16.         }
17.     }
18.     int b[9]={1,2,3,4,5,6,7,8,9};
19.     for(i=1;i<=3;i++)
20.     {
21.         for(j=1;j<=3;j++)
22.         {
23.             for(k=0;k<9;k++)
24.             {
25.                 if(a[i][j]==b[k])   b[k]=11;
26.             }
27.         }
28.     }
29.     for(k=0;k<9;k++)
30.     {
31.         if(b[k]!=11)    a[row][col]=b[k];
32.     }
33.     for(m=0;m<3;m++)
34.     {
35.         cin>>x>>y;
36.         cout<<a[x][y]<<endl;
37.     }
38.     cin>>n;
39.     switch(n)
40.     {
41.         case 1:sum=a[1][1]+a[1][2]+a[1][3];break;
42.         case 2:sum=a[2][1]+a[2][2]+a[2][3];break;
43.         case 3:sum=a[3][1]+a[3][2]+a[3][3];break;
44.         case 4:sum=a[1][1]+a[2][1]+a[3][1];break;
45.         case 5:sum=a[1][2]+a[2][2]+a[3][2];break;
46.         case 6:sum=a[1][3]+a[2][3]+a[3][3];break;
47.         case 7:sum=a[1][1]+a[2][2]+a[3][3];break;
48.         case 8:sum=a[1][3]+a[2][2]+a[3][1];break;
```

```
        }
        int c[25]={0,0,0,0,0,0,10000,36,720,360,80,252,108,72,54,180,72,180,119,36,306,1080,144,1800,3600};
        cout<<c[sum]<<endl;
        return 0;
    }
```

**7-5 有理数比较**

本题要求编写程序，比较两个有理数的大小。

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形范围内的正整数。

输出格式：

在一行中按照"a1/b1 关系符 a2/b2"的格式输出两个有理数的关系。其中">"表示"大于"，"<"表示"小于"，"="表示"等于"。

输入样例 1：

1/2 3/4

输出样例 1：

1/2 < 3/4

输入样例 2：

6/8 3/4

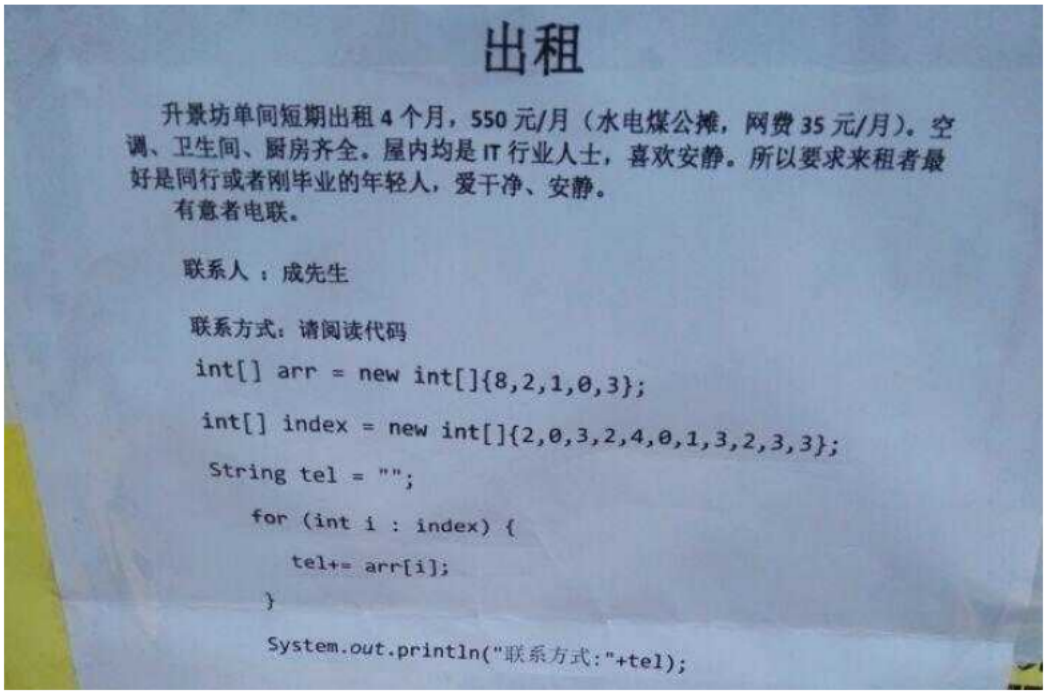
输出样例 2：

6/8 = 3/4

```
#include<iostream>
using namespace std;
int main()
{
    double a,b,c,d;
    char x,y;
    cin>>a>>x>>b>>c>>y>>d;
    if((a/b)<(c/d)){
        cout<<a<<"/"<<b<<" < "<<c<<"/"<<d;
    }
    if((a/b)>(c/d)){
        cout<<a<<"/"<<b<<" > "<<c<<"/"<<d;
    }
    if((a/b)==(c/d)){
        cout<<a<<"/"<<b<<" = "<<c<<"/"<<d;
    }
    return 0;
}
```

**7-1 出租**

下面是新浪微博上曾经很火的一张图：




一时间网上一片求救声，急问这个怎么破。其实这段代码很简单，index数组就是arr数组的下标，index[0]=2 对应 arr[2]=1，index[1]=0 对应 arr[0]=8，index[2]=3 对应 arr[3]=0，以此类推..... 很容易得到电话号码是 18013820100。

本题要求你编写一个程序，为任何一个电话号码生成这段代码 —— 事实上，只要生成最前面两行就可以了，后面内容是不变的。

输入格式：

输入在一行中给出一个由 11 位数字组成的手机号码。

输出格式：

为输入的号码生成代码的前两行，其中 arr 中的数字必须按递减顺序给出。

输入样例：

18013820100

输出样例：

int[] arr = new int[]{8,3,2,1,0};

int[] index = new int[]{3,0,4,3,1,0,2,4,3,4,4};

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string a;cin>>a;
```

```
        int find[10],num[10];
        int i;
        for(i=0;i<10;i++){
            find[i]=0;
            num[i]=-1;
        }
        for(i=0;i<11;i++){
            find[a[i]-'0']=1;
        }
        int k=0;
        for(i=9;i>-1;i--){
            if(find[i]==1){
             num[k++]=i;
            }
        }
        cout<<"int[] arr = new int[]{";
        for(i=0;i<k-1;i++){
            cout<<num[i]<<",";
        }
        cout<<num[k-1]<<"};"<<endl;
        cout<<"int[] index = new int[]{";
        for(i=0;i<11;i++){
            for(int j=0;j<k;j++){
                if( (a[i]-'0')==num[j])cout<<j;
            }
            if(i<10)cout<<",";
        }
        cout<<"};";
        return 0;
    }
```

**7-2 出生年**

以上是新浪微博中一奇葩贴："我出生于 1988 年，直到 25 岁才遇到 4 个数字都不相同的年份。"也就是说，直到 2013 年才达到"4 个数字都不相同"的要求。本题请你根据要求，自动填充"我出生于 $y$ 年，直到 $x$ 岁才遇到 $n$ 个数字都不相同的年份"这句话。

输入格式：

输入在一行中给出出生年份 y 和目标年份中不同数字的个数 n，其中 y 在[1, 3000]之间，n 可以是 2、或 3、或 4。注意不足 4 位的年份要在前面补零，例如公元 1 年被认为是 0001 年，有 2 个不同的数字 0 和 1。

输出格式：

根据输入，输出 x 和能达到要求的年份。数字间以 1 个空格分隔，行首尾不得有多余空格。年份要按 4 位输出。注意：所谓"n 个数字都不相同"是指不同的数字正好是 n 个。如"2013"被视为满足"4 位数字都不同"的条件，但不被视为满足 2 位或 3 位数字不同的条件。

输入样例 1：

1988 4

输出样例 1：

25 2013

输入样例 2：

1 2

输出样例 2：

0 0001

```cpp
#include<iostream>
#include<cstdio>
#include<unordered_set>

using namespace std;

int main(){
    int a,k;
    cin >> a >>k;
    unordered_set<int>set;
    for(int i=a;;i++){
        set.clear();

        int x=i;
        if(x<=999)
            set.insert(0);
        while(x){
            set.insert(x%10);
            x/=10;
        }
        if(set.size()==k)
        {
            printf("%d %04d",(i-a),i);
            break;
        }
    }
    return 0;
}
```

**7-3 找鞍点**

一个矩阵元素的“鞍点”是指该位置上的元素值在该行上最大、在该列上最小。

本题要求编写程序，求一个给定的 n 阶方阵的鞍点。

输入格式：

输入第一行给出一个正整数 n（1≤n≤6）。随后 n 行，每行给出 n 个整数，其间以空格分隔。

输出格式：

输出在一行中按照“行下标 列下标”（下标从 0 开始）的格式输出鞍点的位置。如果鞍点不存在，则输出“NONE”。题目保证给出的矩阵至多存在一个鞍点。

输入样例 1：

4

1 7 4 1

4 8 3 6

1 6 1 2

0 7 8 9

输出样例 1：

2 1

输入样例 2：

2

1 7

4 1

输出样例 2：

NONE

```
#include<iostream>
#include<cstdio>

using namespace std;

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

            if(x>=arr[i][idx])
```

```
                idx=j;
            if(j==n-1)
                a[i][idx]=1;
        }
    }

    for(int i=0;i < n;i++){
        int idx =0;
        for(int j=0;j<n;j++){
            if( arr[j][i]<=arr[idx][i])
                idx=j;
            if(j==n-1 && a[idx][i]) {
                cout<<idx<<" "<<i;
                return 0;
            }
        }
    }
    cout<<"NONE";
    return 0;
}
```

**7-1 敲笨钟**

微博上有个自称“大笨钟 V”的家伙，每天敲钟催促码农们爱惜身体早点睡觉。为了增加敲钟的趣味性，还会糟改几句古诗词。其糟改的方法为：去网上搜寻压“ong”韵的古诗词，把句尾的三个字换成“敲笨钟”。例如唐代诗人李贺有名句曰：“寻章摘句老雕虫，晓月当帘挂玉弓”，其中“虫”（chong）和“弓”（gong）都压了“ong”韵。于是这句诗就被糟改为“寻章摘句老雕虫，晓月当帘敲笨钟”。

现在给你一大堆古诗词句，要求你写个程序自动将压“ong”韵的句子糟改成“敲笨钟”。

输入格式：

输入首先在第一行给出一个不超过 20 的正整数 N。随后 N 行，每行用汉语拼音给出一句古诗词，分上下两半句，用逗号 , 分隔，句号 . 结尾。相邻两字的拼音之间用一个空格分隔。题目保证每个字的拼音不超过 6 个字符，每行字符的总长度不超过 100，并且下半句诗至少有 3 个字。

输出格式：

对每一行诗句，判断其是否压“ong”韵。即上下两句末尾的字都是“ong”结尾。如果是压此韵的，就按题面方法糟改之后输出，输出格式同输入；否则输出 Skipped，即跳过此句。

输入样例：

5

xun zhang zhai ju lao diao chong, xiao yue dang lian gua yu gong.

tian sheng wo cai bi you yong, qian jin san jin huan fu lai.

xue zhui rou zhi leng wei rong, an xiao chen jing shu wei long.

zuo ye xing chen zuo ye feng, hua lou xi pan gui tang dong.

ren xian gui hua luo, ye jing chun shan kong.

输出样例：

xun zhang zhai ju lao diao chong, xiao yue dang lian qiao ben zhong.

Skipped

xue zhui rou zhi leng wei rong, an xiao chen jing qiao ben zhong.

Skipped

Skipped

```
#include <iostream>
#include <stack>
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    string strs[25];
    cin>>n;
    cin.ignore();//用于解决 cin 与 getline 连续使用导致的缓冲区回车问题
    for(int i=0; i<n; i++)
    {
        getline(cin,strs[i]);
    }
    for(int i=0; i<n; i++)
    {
        int loc_1=0;
        while(strs[i][loc_1]!=',')//寻找逗号的位置
        {
            loc_1++;
        }
        //如果上句压
        if(loc_1>=3&&strs[i].substr(loc_1-3,3)=="ong")
        {
            while(strs[i][loc_1]!='.')
            {
                loc_1++;
            }
            //如果下句也压
            if(strs[i].substr(loc_1-3,3)=="ong")
            {
                int cnt=0;
                int loc_2=strs[i].length()-1;
                while(cnt!=3)
                {
                    loc_2--;
                    if(strs[i][loc_2]==' ')
                    {
                        cnt++;
```

```
                    }
                }
                cout<<strs[i].substr(0,loc_2)+" qiao ben zhong."<<endl;
            }
            else
            {
                cout<<"Skipped"<<endl;
            }
        }
        else
        {
            cout<<"Skipped"<<endl;
        }
    }
}
```

**7-2 特立独行的幸福**

对一个十进制数的各位数字做一次平方和，称作一次迭代。如果一个十进制数能通过若干次迭代得到 1，就称该数为幸福数。1 是一个幸福数。此外，例如 19 经过 1 次迭代得到 82，2 次迭代后得到 68，3 次迭代后得到 100，最后得到 1。则 19 就是幸福数。显然，在一个幸福数迭代到 1 的过程中经过的数字都是幸福数，它们的幸福是依附于初始数字的。例如 82、68、100 的幸福是依附于 19 的。而一个**特立独行**的幸福数，是在一个有限的区间内不依附于任何其它数字的；其**独立性**就是依附于它的的幸福数的个数。如果这个数还是个素数，则其独立性加倍。例如 19 在区间[1, 100] 内就是一个特立独行的幸福数，其独立性为 2×4=8。

另一方面，如果一个大于 1 的数字经过数次迭代后进入了死循环，那这个数就不幸福。例如 29 迭代得到 85、89、145、42、20、4、16、37、58、89、..... 可见 89 到 58 形成了死循环，所以 29 就不幸福。

本题就要求你编写程序，列出给定区间内的所有特立独行的幸福数和它的独立性。

输入格式：

输入在第一行给出闭区间的两个端点：1<A<B≤10^4。

输出格式：

按递增顺序列出给定闭区间 [A,B] 内的所有特立独行的幸福数和它的独立性。每对数字占一行，数字间以 1 个空格分隔。

如果区间内没有幸福数，则在一行中输出 SAD。

输入样例 1：

10 40

输出样例 1：

19 8

23 6

28 3

31 4

32 3

注意：样例中，10、13 也都是幸福数，但它们分别依附于其他数字（如 23、31 等等），所以不输出。其它数字虽然其实也依附于其它幸福数，但因为那些数字不在给定区间 [10, 40] 内，所以它们在给定区间内是特立独行的幸福数。

输入样例 2：

110 120

输出样例 2：

SAD

```
#include<iostream>
#include<cstring>
#include<string>
#include<cstdio>
using namespace std;
int fa[10010];
int getfa(int x)//得到各位数字的平方和
{
    int sum=0;
    while(x)
    {
        sum=sum+(x%10)*(x%10);
        x=x/10;
    }
    return sum;
}
int find_root(int x)//找到最终指向的根，如果这个数是幸福数，则最终一定会指到 1，1 的根就是自己
{
    int ans=x;
    int i=0;
    while(fa[ans]!=ans)
    {
        ans=fa[ans];
        i++;
        if(i>=10000)
            return -1;
    }
    return ans;
}
bool fun(int x,int n,int m)//判断 nm 范围内否有数字指向 x，即判断是否独立
{
    for(int i=n;i<=m;i++)
        if(fa[i]==x)
```

```
            return false;
        return true;
    }
    bool sushu(int x)//判断素数
    {
        for(int i=2;i*i<=x;i++)
            if(x%i==0)
                return false;
        return true;
    }
    int main()
    {
        int n,m;
        cin>>n>>m;
        int c=0;
        for(int i=1;i<=10000;i++)//初始化令所有数字先指向自己
            fa[i]=i;
        for(int i=1;i<=10000;i++)//令所有数字指向自己的各位平方和
            fa[i]=getfa(i);
        for(int i=n;i<=m;i++)
        {
            if(fun(i,n,m)&&find_root(i)==1)//如果找根能找到1，而且n到m没有数字指向它，就是特立独行的幸福数
            {
                cout<<i<<" ";
                int flag=1,x=i;
                while(getfa(x)!=1)//计算迭代次数
                {
                    x=getfa(x);
                    flag++;
                }
                if(sushu(i))flag*=2;
                cout<<flag<<endl;
                c++;//记录输出了多少个特立独行的幸福数
            }
        }
        if(c==0)
            cout<<"SAD"<<endl;
        return 0;
    }
```

**7-1 简单求阶乘问题**

本题要求编写程序，计算 N 的阶乘。

输入格式:

输入在一行中给出一个不超过 12 的正整数 N。

输出格式:

在一行中输出阶乘的值。

输入样例:

4

输出样例:

24

```
#include <bits/stdc++.h>
using namespace std;
int factorial(int num)
{
    int sum = 1;
    for (int i = 1; i <= num; i++)
        sum *= i;
    return sum;
}
int main()
{
    int i;
    cin>>i;
    cout<<factorial(i);
    return 0;
}
```

**7-2 谷歌的招聘**

2004 年 7 月，谷歌在硅谷的 101 号公路边竖立了一块巨大的广告牌（如下图）用于招聘。内容超级简单，就是一个以 .com 结尾的网址，而前面的网址是一个 10 位素数，这个素数是自然常数 e 中最早出现的 10 位连续数字。能找出这个素数的人，就可以通过访问谷歌的这个网站进入招聘流程的下一步。

自然常数 e 是一个著名的超越数，前面若干位写出来是这样的：e = 2.71828182845904523536028747135266249775724709369995957496696762772407663035354759457138217852516642**7427466391**932003059921... 其中粗体标出的 10 位数就是答案。

本题要求你编程解决一个更通用的问题：从任一给定的长度为 L 的数字中，找出最早出现的 K 位连续数字所组成的素数。

输入格式：

输入在第一行给出 2 个正整数，分别是 L（不超过 1000 的正整数，为数字长度）和 K（小于 10 的正整数）。接下来一行给出一个长度为 L 的正整数 N。

输出格式：

在一行中输出 N 中最早出现的 K 位连续数字所组成的素数。如果这样的素数不存在，则输出 404。注意，原始数字中的前导零也计算在位数之内。例如在 200236 中找 4 位素数，0023 算是解；但第一位 2 不能被当成 0002 输出，因为在原始数字中不存在这个 2 的前导零。

输入样例 1：

20 5

23654987725541023819

输出样例 1：

49877

输入样例 2：

10 3

2468001680

输出样例 2：

404

```
#include<iostream>
#include<string>
using namespace std;
int prim(int n)
{
    if(n == 0 || n == 1) return 0;
    for(int i = 2;i*i <= n;i++){
        if(n%i == 0){
            return 0;
        }
    }
    return 1;
}
int main()
{
    int l,k,a;
    int flag = 0;
    string s,temp;
    cin >> l >> k;
    cin >> s;
    for(int i = 0;i <= l-k;i++){
        temp = s.substr(i,k);//截取 i 到 k 的字符
        a = stoi(temp); //将 string 类型转化为 int 类型
        if(prim(a)){
            cout << temp << endl;
            flag = 1;
            break;
        }
    }
    if(flag == 0)
    cout << "404" << endl;
```

```
32.         return 0;
33.     }
```