模板

大数除 2

```
#include<iostream>
using namespace std;
#include<string.h>
char a[50],b[50];
void div2(char a[])
{
        int i,j;
        int d=0;
        int alen;
        alen=strlen(a);
   for(i=0;i<alen;i++)
        {
                b[i]=(((a[i]-'0')+d*10)/2)+'0';
        d=((a[i]-'0')+d*10)%2;
        }
        b[i]='\0';
        if(b[0]=='0')
        {
        for(j=0;j<i;j++)
                {
                        b[j]=b[j+1];
                }
        }
}
int main()
{
        gets(a);
        div2(a);
        puts(b);
        return 0;
```

```
}
```

大数相加

```
#include<iostream>
using namespace std;
#include<string.h>
char A[52],B[52];
void add(char a[],char b[])
{
	int i,j,k,up,x,y,z,l;
	char *c;
	if(strlen(a)>strlen(b))
		l=strlen(a)+2;
	else
		l=strlen(b)+2;
	c=(char*)malloc(l*sizeof(char));
	i=strlen(a)-1;
	j=strlen(b)-1;
	k=0;
	up=0;
	while(i>=0||j>=0)
	{
		if(i<0)
			x='0';
		else
			x=a[i];
		if(j<0)
			y='0';
		else
			y=b[j];
		z=x-'0'+y-'0';
		if(up)
			z=z+1;
		if(z>9)
```

```
                {
                        up=1;
                        z=z%10;
                }
                else
                        up=0;
                c[k++]=z+'0';
                i--;
                j--;
        }
        if(up)
                c[k++]='1';
        i=0;
        c[k]='\0';
        for(k=k-1;k>=0;k--)
                A[i++]=c[k];
        A[i]='\0';
}
void main()
{
        gets(A);
        gets(B);
        add(A,B);
        puts(A);
}
```

大数相减

```
#include<iostream>
using namespace std;
#include<string.h>
char A[52],B[52];char t[52];
void sub(char a[],char b[])
{
```

```
    int i,l1,l2,k;
    l1=strlen(a);
    l2=strlen(b);
    t[l1]='\0';
    l1--;
    for(i=l2-1;i>=0;i--,l1--)
    {
        if(a[l1]-b[i]>=0)
            t[l1]=a[l1]-b[i]+'0';
        else
       {
            t[l1]=10+a[l1]-b[i]+'0';
            a[l1-1]=a[l1-1]-1;
        }
    }
    k=l1;
    while(a[k]<0)
    {
        a[k]=a[k]+10;
        a[k-1]=a[k-1]-1;
        k--;
    }
    while(l1>=0)
    {
        t[l1]=a[l1];
        l1--;
    }
loop:
    if(t[0]=='0')
    {
        l1=strlen(a);
        for(i=0;i<l1-1;i++)
            t[i]=t[i+1];
```

```
                t[l1-1]='\0';
                goto loop;
        }
        if(strlen(t)==0)
        {
                t[0]='0';
                t[1]='\0';
        }

        for(i=0,k=0;k<strlen(t);k++)
                A[i++]=t[k];
        A[i]='\0';
}
int main()
{
        gets(A);
        gets(B);

        sub(A,B);
        puts(A);
        return 0;
}
```

给出年，月，日，计算该日是该年的第几天（年月计算）

```
#include<stdio.h>

struct data
{
        int year;
        int month;
        int day;
};

int main()
```

```
{
	data d;
	int da;
	int a[12]={31,28,31,30,31,30,31,31,30,31,30,31};
	while(scanf("%d-%d-%d",&d.year,&d.month,&d.day)!=EOF)
	{
		da=0;
		a[1]=28;
		if((d.year%4==0&&d.year%100!=0)||(d.year%100==0&&d.year%400==0))
			a[1]=29;
	for(int i=0;i<d.month-1;i++)
		da=da+a[i];
	da=da+d.day;
	printf("%d\n",da);
	}
	return 0;
}
```

给出一个正整数，求出它是几位数，分别输出每一位数字，按逆序输出各位数字（数学问题）

```
#include<iostream.h>

void main()
{
	long a;
	cin>>a;
	int b=1;
	long a1=a;
	while(a1!=a1%10)
	{
		a1=int(a1/10);
		b++;
	}
	cout<<"它是一个"<<b<<"位数。"<<endl;
	long c=0;
```

```
        long d=1;
        long e=0;
        for(int i=0;i<b;i++)
        {
    d=d*10;
                c=(a%d-e)*10/d;
                e=a%d;
                cout<<c<<endl;
        }
}
```

求两个数 m 和 n 的最大公约数和最小公倍数（欧几里德定理）

```
#include<iostream.h>

void main()
{
        int m,n;
        cin>>m>>n;
        for(int i=n;i>0;i--)
        {
                if(m%i==0&&n%i==0)
                {
                        cout<<"最大公约数"<<endl<<i<<endl;
                   break;
                }

        }
        for(int j=n;;j++)
        {
                if(j%m==0&&j%n==0)
                {

                        cout<<"最小公倍数"<<endl<<j<<endl;
                   break;
```

```
                }

        }
}
```

输出 1900 年到 2000 年中是闰年的年份

```
#include<iostream.h>

void main()
{
	for(int i=1900;i<=2000;i++)
		if(i%4==0&&i%100!=0||i%100==0&&i%400==0)
			cout<<i<<endl;
}
```

万年历（年月计算）

```
#include<iostream.h>
#include<stdio.h>
int getYearWeekDay(int y);
int getYearDays(int y);
int isLeap(int y);
int getMonthWeekDay(int y, int m);
int getMonthDays(int y, int m);
void printYear(int y);
void printMonth(int y, int m);

int count=0;
int main()
{
	int y;
	cout<<"请输入一个年份："<<endl;
	cin>>y;
	printYear(y);
	return 1;
}

int getYearWeekDay(int y)//计算 y 年的第一天是星期几，以 2000 年作为参照，其第一天是星期六
{
	int sum=0;
```

```
	int i;
	if(y>=2000)
	{
		for(i=2000;i<y;i++)
		{
			sum+=getYearDays(i);
		}
		return (sum+6)%7;
	}
	else
	{
		for(i=y;i<2000;i++)
		{
			sum+=getYearDays(i);
		}
		return (-sum%7+6)%7;
	}
}

int getYearDays(int y)
{
	return isLeap(y)?366:365;
}

int isLeap(int y)
{
	return y%4==0&&y%100!=0||y%400==0;
}

int getMonthWeekDay(int y, int m)//计算每一月的第一天是星期几
{
	int sum=0;
	int i;
```

```
	for(i=1;i<m;i++)
	{
		sum+=getMonthDays(y,i);
	}
	return (getYearWeekDay(y)+sum)%7;
}

int getMonthDays(int y,int m)
{
	switch(m)
	{
	case 1:
	case 3:
	case 5:
	case 7:
	case 8:
	case 10:
	case 12:
		return 31;
	case 4:
	case 6:
	case 9:
	case 11:
		return 30;
	case 2:
		return isLeap(y)?29:28;
	default:
		return 0;
	}
}

void printYear(int y)
{
```

```
        int i;
        count=1;
        for(i=1;i<=12;i++)
        {
                printMonth(y,i);
        }
}

void printMonth(int y, int m)
{
        int i=0;
        int w;
        cout<<endl<<"************************"<<m<<"月
**************************"<<endl<<endl;
        w=getMonthWeekDay(y,m);
        for(i=0;i<w;i++)
        {
                printf("       ");
        }
        for(i=1;i<=getMonthDays(y,m);i++)
        {
                printf("%7i",i);
                w++;
                w%=7;
                if(w==0&&i<getMonthDays(y,m))
                {
                        printf("\n");
                }
        }
        cout<<endl;
}
```

递归

Children’s Queue

Time Limit: 2000/1000 MS (Java/Others) Memory Limit: 65536/32768 K (Java/Others)
Total Submission(s): 1327 Accepted Submission(s): 397

Problem Description
There are many students in PHT School. One day, the headmaster whose name is PigHeader wanted all students stand in a line. He prescribed that girl can not be in single. In other words, either no girl in the queue or more than one girl stands side by side. The case n=4 (n is the number of children) is like
FFFF, FFFM, MFFF, FFMM, MFFM, MMFF, MMMM
Here F stands for a girl and M stands for a boy. The total number of queue satisfied the headmaster's needs is 7. Can you make a program to find the total number of queue with n children?

Input
There are multiple cases in this problem and ended by the EOF. In each case, there is only one integer n means the number of children (1<=n<=1000)

Output
For each test case, there is only one integer means the number of queue satisfied the headmaster's needs.

Sample Input
1
2
3

Sample Output
1
2
4

题目大意：
一个队伍中有 n 个学生，但规定女生不能单独站，也就是说，要么队伍中没有女生，要么有两个或两个以上的女生站在一起（这是袒护弱势力的体现！）。问有多少这样的队伍。
我的思路：
设 n 个学生生按规则排的队列有 f(n)种，那么当总数少一个时会是什么况呢？显然，要么少一个男的，要么少一个女的。故有如下况：
（1） 少一个男的，也就是说，第 n 个加进去的是男生。此时，队列 n-1 只需按原规则站好即可，故有 f(n-1)种方。
（2） 少一个女的，也就是说，第 n 个加进去的是女生。此时，前面的肯定不能是男生，因为后面加进去的女生只有一个，不符合规则，因此第 n-1 个(也就是前 1 个)必需是女生。若 n-2 个是按规则站好的，则有 f(n-2)种方法；若 n-2 个不是按规则站好的（因为第 n-1 个,第 n 个是女生，所以第 n-2 个可以是女生，第 n-3 个可以是男生。当没加进第 n-1 和第 n 个女生的时候，这是不合规则的），即第 n-2 个是女生，第 n-3 个是男生，此后第 n-4 个必需按规则站好，故有 f(n-4)种方法。
于是得递推公式：
f(n)=f(n-1)+f(n-2)+f(n-4);
下一步就要确定初始条件：
由于递推公式有 f(n-1)、f(n-2)和 f(n-4)，故必需给出 f(1)、f(2) 、f(3)和 f(4)
由题意得：f(1)=1；f(2) =2；f(3)=4； f(4)=7。
注意：由于本题数据量较量大，必需要用高精度，同时用数组留存结果来代替递归。

```
#include<iostream>

using namespace std;

int main()

{

        int i,j,n;

        int num[1000][35]={0};
```

```
    num[0][0]=1;
    num[1][0]=2;
    num[2][0]=4;
    num[3][0]=7;
    for(i=4;i<1000;i++)
        {
            for(j=0;j<35;j++)
                num[i][j]=num[i-1][j]+num[i-2][j]+num[i-4][j];
            for(j=0;j<34;j++)
            {
                if(num[i][j]>100000000)
                {
                    num[i][j+1]+=num[i][j]/100000000;
                    num[i][j]%=100000000;
                }
            }
        }
        while(scanf("%d",&n)!=EOF)
        {
            for(i=34;i>0;i--)
            {
                if(num[n-1][i]!=0)
                    break;
            }
            printf("%d",num[n-1][i]);
            for(j=i-1;j>=0;j--)
                printf("%08d",num[n-1][j]);
            printf("\n");
        }
        return 0;
}
```

正方形

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

```
__int64 f(int n)
{
	if(n==0)
		return 0;
	else
		return f(n-1)+n*n;
}
int main()
{
	int n;
	scanf("%d",&n);
	printf("%I64d",f(n));
	return 0;
}
```

Redraiment 小时候走路喜欢蹦蹦跳跳，他最喜欢在楼梯上跳来跳去。
但年幼的他一次只能走上一阶或者一下子蹦上两阶。
现在一共有 N 阶台阶，请你计算一下 Redraiment 从第 0 阶到第 N 阶共有几种走法。

输入

输入包括多组数据。
每组数据包括一行:N(1≤N≤40)。
输入以 0 结束。

输出

对应每个输入包括一个输出。
为 redraiment 到达第 n 阶不同走法的数量。

样例输入

```
1
2
0
```

样例输出

```
1
2
```

```
#include <stdio.h>
int main()
{
```

```
    int f[41] = {1, 1}, n, i;
    for(i = 2; i <= 40; i++)
        f[i] = f[i-1] + f[i-2];
    while(scanf("%d", &n), n)
        printf("%d\n", f[n]);
          return 0;
}
```

蟠桃记

孙悟空在大闹蟠桃园的时候，第一天吃掉了所有桃子总数一半多一个，第二天又将剩下的桃子吃掉一半多一个，以后每天吃掉前一天剩下的一半多一个，到第n天准备吃的时候只剩下一个桃子。这下可把神仙们心疼坏了，请帮忙计算一下，第一天开始吃的时候桃子一共有多少个桃子。

输入

输入数据有多组，每组占一行，包含一个正整数n（1≤n≤30），表示只剩下一个桃子的时候是在第n天发生的。
输入以0结束。

输出

对于每组输入数据，输出第一天开始吃的时候桃子的总数，每个测试实例占一行。

样例输入

```
2
4
0
```

样例输出

```
4
22
```

```
#include<stdio.h>
int main()
{
          int n;
          long k=1;
          do
          {
                    scanf("%d",&n);
```

```
                if(n!=0)
                {
                        for(int i=0;i<n-1;i++)
                        {
                                k=(k+1)*2;
                        }
                        printf("%d\n",k);
                        k=1;
                }
        }while(n!=0);
        return 0;
}
```

养兔子

一对成熟的兔子每天能且只能产下一对小兔子，每次都生一公一母，每只小兔子的成熟期是一天。某人领养了一对小兔子，一公一母，请问第 N 天以后，他将会得到多少对兔子。

输入

测试数据包括多组，每组一行，为整数 n(1≤n≤90)。
输入以 0 结束。

输出

对应输出第 n 天有几对兔子(假设没有兔子死亡现象，而且是一夫一妻制)。

样例输入

```
1
2
0
```

样例输出

```
1
2
```

提示

数据类型可以用 64 位整数:__int64

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

```
int main()
{
    __int64 f[91]={1,1};
    int n,i;
    for(i=2;i<=90;i++)
        f[i]=f[i-1]+f[i-2];
    while(scanf("%d",&n),n)
        printf("%I64d\n",f[n]);
            return 0;
}
```

汉诺塔 III

约 19 世纪末，在欧州的商店中出售一种智力玩具，在一块铜板上有三根杆，最左边的杆上自上而下、由小到大顺序串着由 64 个圆盘构成的塔。目的是将最左边杆上的盘全部移到右边的杆上，条件是一次只能移动一个盘，且不允许大盘放在小盘的上面。
现在我们改变游戏的玩法，不允许直接从最左(右)边移到最右(左)边(每次移动一定是移到中间杆或从中间移出)，也不允许大盘放到下盘的上面。
Daisy 已经做过原来的汉诺塔问题和汉诺塔 II，但碰到这个问题时，她想了很久都不能解决，现在请你帮助她。现在有 N 个圆盘，她至少多少次移动才能把这些圆盘从最左边移到最右边？

输入

包含多组数据，每次输入一个 N 值(1<=N<35)。

输出

对于每组数据，输出移动最小的次数。

样例输入

```
1
3
12
```

样例输出

```
2
26
531440
```

```
#include<iostream>
using namespace std;
__int64 m(__int64 n)
{
```

```
    if(n==1) return 2;
    return 3*m(n-1)+2;
}
int main()
{
    __int64 n;
    while(scanf("%I64d",&n)!=EOF)
    {
        printf("%I64d\n",m(n));
    }
    return 0;
}
```

ZOJ 1629 Counting Triangles

Given an equilateral triangle with n the length of its side, program to count how many triangles in it.

Input

The length n (n <= 500) of the equilateral triangle's side, one per line.

process to the end of the file

Output
The number of triangles in the equilateral triangle, one per line.
Sample Input
1
2
3
Sample Output
1
5
13

可以从递推的角度去思考！！！
每增加1列多出2*n-1个单位边长的三角形，分上三角和下三角考虑！！！
1 -> 1
2 -> 5
3 -> 13
4 -> 27

5 -> 48
10 -> 315
500 -> 31406375

```
#include<iostream>

using namespace std;

int main()

{

          int i,j,f[502];

          f[1]=1;

          for(i=2;i<=500;i+=2)

          {

                    f[i]=f[i-1]+i*(i+1)/2;

                    f[i]=f[i]+i/2;

                    j=i/2;

                    f[i]=f[i]+j*(j-1);

                    f[i+1]=f[i]+(i+1)*(i+2)/2;

                    f[i+1]=f[i+1]+j*(j+1);

          }

          while(cin>>j)

                    cout<<f[j]<<endl;

          return 0;

}
```

ZOJ 1629 Counting Triangles

Given an equilateral triangle with n the length of its side, program to count how many triangles in it.
Input
The length n (n <= 500) of the equilateral triangle's side, one per line.
process to the end of the file
Output
The number of triangles in the equilateral triangle, one per line.
Sample Input

1
2
3
Sample Output
1
5

13

Counting Triangles

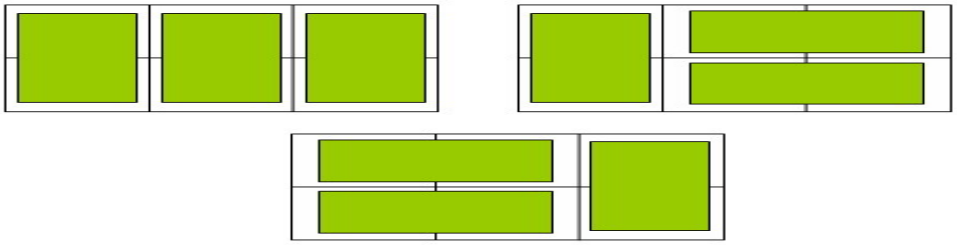
注意还有倒着的三角形...边长为 1 的三角形有 n*n 个，然后正着的三角形很好数，sigma(i * (i + 1) / 2)(1 < i < n),倒着的三角形可以这样理解:如果计算便成为 i 的三角形，那么在边长为 j 的行，一定能找到 j - i + 1 个这样的三角形，用循环累加即可。注意下，如果 i + j > n 了，那么就不会再有倒着的三角形了。

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

}

骨牌铺方格

在 2×n 的一个长方形方格中,用一个 1× 2 的骨牌铺满方格,输入 n ,输出铺放方案的总数.
例如 n=3 时,为 2× 3 方格, 骨牌的铺放方案有三种,如下图 ：



输入

输入数据由多行组成, 每行包含一个整数 n,表示该测试实例的长方形方格的规格是 2×n (0<n<=50)。

输出

对于每个测试实例, 请输出铺放方案的总数, 每个实例的输出占一行。

样例输入

```
1
3
2
```

样例输出

```
1
3
2
```

```
include<stdio.h>
int main()
{
    __int64 s[50]={1,2};
          int n,i;
          for(i=2;i<50;i++)
          {
                    s[i]=s[i-2]+s[i-1];
          }
    while(scanf("%d",&n)!=EOF)
```

```
    {
        printf("%I64d\n",s[n-1]);
    }
    return 0;
}
```

不容易系列之二

你活的不容易，我活的不容易，他活的也不容易。不过，如果你看了下面的故事，就会知道，有位老汉比你还不容易。

重庆市郊黄泥板村的徐老汉（大号徐东海，简称 XDH）这两年辛辛苦苦养了不少羊，到了今年夏天，由于众所周知的高温干旱，实在没办法解决牲畜的饮水问题，就决定把这些羊都赶到集市去卖。从黄泥板村到交易地点要经过 N 个收费站，按说这收费站和徐老汉没什么关系，但是事实却令徐老汉欲哭无泪：

（镜头回放）

近景：老汉，一群羊
远景：公路，收费站

……

收费员（彬彬有礼+职业微笑）："老同志，请交过路费！"

徐老汉（愕然，反应迟钝状）："锅，锅，锅，锅-炉-费？我家不烧锅炉呀？"

收费员（职业微笑依然）："老同志，我说的是过-路-费，就是你的羊要过这个路口必须交费，understand?"

徐老汉（近镜头 10 秒，嘴巴张开）："我-我-我知道汽车过路要收费，这羊也要收费呀？"

收费员（居高临下+不解状）："老同志，你怎么就不明白呢，那么我问你，汽车几个轮子？"

徐老汉（稍放松）："这个我知道，今天在家里我孙子还问我这个问题，4 个！"

收费员（生气，站起）："嘿！老头，你还骂人不带脏字，既然知道汽车四个轮子，难道就不知道这羊有几条腿吗？！"

徐老汉（尴尬，依然不解状）："也，也，也是 4 个呀，这有关系吗？"

收费员（生气，站起）："怎么没关系！我们头说了，只要是 4 条腿的都要收费！"

……

(画外音)

由于徐老汉没钱，收费员就将他的羊拿走一半，看到老汉泪水涟涟，犹豫了一下，又还给老汉一只。巧合的是，后面每过一个收费站，都是拿走当时羊的一半，然后退还一只，等到老汉到达市场，就只剩下 3 只羊了。

你,当代有良知的青年，能帮忙算一下老汉最初有多少只羊吗？

输入

输入数据第一行是一个整数 N，表示测试数据的组数，下面由 N 行组成，每行包含一个整数 a(0<a<=30)，表示收费站的数量。

输出

对于每个测试实例，请输出最初的羊的数量,每个测试实例的输出占一行。

样例输入

```
2
1
2
```

样例输出

```
4
6
```

```
#include<iostream>
using namespace std;
int main()
{
	int N,a;
	int result;
	cin>>N;
	while(N--)
	{
		result=3;
		cin>>a;
		for(int i=0;i<a;i++)
		{
			result=(result-1)*2;
		}
		cout<<result<<endl;
	}
	return 0;
}
```

神、上帝以及老天爷

TZC 2008 ACM contest 的颁奖晚会隆重开始了！
为了活跃气氛，组织者举行了一个别开生面、奖品丰厚的抽奖活动，这个活动的具体要求是这样的：
首先，所有参加晚会的人员都将一张写有自己名字的字条放入抽奖箱中；
然后，待所有字条加入完毕，每人从箱中取一个字条；
最后，如果取得的字条上写的就是自己的名字，那么"恭喜你，中奖了！"
大家可以想象一下当时的气氛之热烈，毕竟中奖者的奖品是大家梦寐以求的 Twins 签名照呀！不过，正如所有试图设计的喜剧往往以悲剧结尾，这次抽奖活动最后竟然没有一个人中奖！
我的神、上帝以及老天爷呀，怎么会这样呢？
不过，先不要激动，现在问题来了，你能计算一下发生这种情况的概率吗？
不会算？难道你也想以悲剧结尾？！

输入

输入数据的第一行是一个整数 C,表示测试实例的个数，然后是 C 行数据，每行包含一个整数 n(1<n<=20),表示参加抽奖的人数。

输出

对于每个测试实例，请输出发生这种情况的百分比，每个实例的输出占一行，结果保留两位小数(四舍五入)，具体格式请参照 sample output。

样例输入

```
1

2
```

样例输出

```
50.00%
```

```
#include<iostream>

using namespace std;

int main()

{

          int n,a;

          int sum,i;

    double f[100];

    cin>>n;

          while(n--)

          {

                    scanf("%d",&a);

        sum=1;

                    for(i=1;i<=a;i++)

                              sum*=i;

                    f[2]=1;
```

```
                f[3]=2;
                for(i=4;i<=a;i++)
                        f[i]=(i-1)*(f[i-1]+f[i-2]);
                if(a<7) printf("%.2lf%%\n",f[a]*100.0/sum);
                else printf("36.79%%\n");
        }
        return 0;
}
```

母牛的故事

有一头母牛，它每年年初生一头小母牛。每头小母牛从第四个年头开始，每年年初也生一头小母牛。请编程实现在第 n 年的时候，共有多少头母牛？

输入

输入数据由多个测试实例组成，每个测试实例占一行，包括一个整数 n(0<n<55)，n 的含义如题目中描述。
n=0 表示输入数据的结束，不做处理。

输出

对于每个测试实例，输出在第 n 年的时候母牛的数量。
每个输出占一行。

样例输入

```
2
4
5
0
```

样例输出

```
2
4
6
```

```
#include<iostream>
using namespace std;
int main()
{
        int n;
        int F[55]={1,2,3};
```

```
        for(int i=3;i<55;i++)
        {
                F[i]=F[i-1]+F[i-3];
        }
        while(cin>>n,n)
        {
                cout<<F[n-1]<<endl;
        }
        return 0;
}
```

字符串与数组处理

描述

琛琛今天参加高中同学聚会,知道了好多同学的联系方式,高兴 ing~~~可惜没过多久,他发现他把所有同学的名字都写到通讯录的一行里去了,他怎么也分不清他到底有哪些同学.不过还好,他记得一些好朋友的名字(尤其是女生),他希望在所有的名字里找到这些他熟悉的名字.....

输入

多组输入数据. 每组数据第一行包含一个整数 N(1<=N<=10) ,表示琛琛记得的朋友的个数.接下来一行字符串是琛琛的通讯录.最后 N 行每行有一个字符串,表示每个琛琛记得的名字. 所有字符串长度都不超过 1000,都由英文字母组成.

输出

对于每组输入数据,对琛琛记得的每个名字,查找在琛琛的通讯录中是否存在. 如果存在输出 yes,否则输出 no

样例输入

```
3
aaaab
a
ab
c
```

样例输出

```
yes
yes
no
```

```
#include<stdio.h>
int main()
```

```
{
	int N,i,j,p,q=0,b;
	char A[11][1001];
	while(scanf("%d",&N)!=EOF)
	{
		getchar();
		for(i=0;i<=1000;i++)
		{
			scanf("%c",&A[0][i]);
			if(A[0][i]=='\n')
				break;
		}
		b=i;
		for(i=1;i<=N;i++)
		{
			for(j=0;j<=1000;j++)
			{
				scanf("%c",&A[i][j]);
				if(A[i][j]=='\n')
					break;
			}
			for(p=0;p<b;p++)
			{
				if(A[i][q]==A[0][p])
					q++;
				else
				{
					if(q!=0)
						p--;
					q=0;
				}
				if(q==j)
```

```
                                        break;
                                }
                                if(q==j)
                                        printf("yes\n");
                                else
                                        printf("no\n");
                        }
                }
                return 0;
}
```

选礼物

时间限制(普通/Java):1000MS/10000MS　　运行内存限制:65536KByte
总提交:462　　测试通过:143

描述

xFengChenx MM 今天特别高兴，因为她收到了来自全国各地同学寄来的生日礼物。她按照收到包裹的顺序将所有礼物排成一行，每个礼物对应一定的价值（不一定是正数），xFengChenx MM 希望从这些礼物里挑选连续的一段(一个或连续多个)礼物，使其总价值在所有的连续片断中最大。

输入

多组数据。每组数据第一行为一个正数 N（1<=N<=100），表示礼物的个数。第二行包含 N 个整数，按照礼物到来的先后顺序，分别表示第 1 个，第 2 个。。。。第 N 个礼物的价值。每个礼物价值的绝对值不超过 100。

输出

每组数据输出一行，为一个整数，表示所有礼物中总价值最大的连续子片段。

样例输入

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

样例输出

```
15
```

题目来源

Narashy

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

```
int main()
{
    int n=0,i,l,t;
    int a[10000];
    while(scanf("%d",&n)!=EOF)
    {
        for(i=0;i<n;i++)
        {
           scanf("%d",&a[i]);
           if(i!=0)
           {
               if(a[i]+t>l)
                   l=a[i]+t;
               if(a[i]+t>a[i])
                   t=t+a[i];
               else
                   t=a[i];
               if(a[i]>l)
                    t=l=a[i];
           }
           else
               t=l=a[0];
        }
        printf("%d\n",l);
    }
    return 0;
}
```

词组缩写

```
#include <stdio.h>
int main(void)
{
  int t,i,k,flag;
```

```
  char s,result[31];
  scanf("%d",&t);
  for(i=0; i<t ; ++i)
  {
    k=0,flag=1;
    scanf(" %c",&s);
    while(s != '\n')
    {
      if(s != ' ' && flag == 1)
      {
        if(s>='a')
          s -= 32;
        result[k++]=s;
         flag = 0;
      }
      else if(s == ' ')
      {
         flag=1;
      }
      scanf("%c",&s);
    }
    result[k]='\0';
    printf("%s\n",result);
  }
  return 0;
}
```

奇偶求值

```
#include <stdio.h>
int main(void)
{
  int n,i,j,A[2001]={0},p=0,q=0,f=0,g=0;
  scanf("%d",&n);
```

```
for(j=0; j<n ; ++j)
{
  scanf("%d",&i);
  if(i>0)
    ++A[i+1000];
  else if(i<0)
    ++A[-i];
  else
    ++g;
}
for(i=1000; i>0 ; --i)
{
  if(A[i]==0)
    continue;
  f++;
  if(A[i]%2==0)
  {
    p += -i*A[i]/2;
    q += -i*A[i]/2;
  }
  else
  {
    if(f%2!=0)
      p += -i;
    else
      q += -i;
    p += -i*(A[i]-1)/2;
    q += -i*(A[i]-1)/2;
  }
  f += A[i]-1;
}
f += g;
```

```
  for(i=1001; i<2001 ; ++i)
  {
    if(A[i]==0)
      continue;
    f++;
    if(A[i]%2==0)
    {
      p += (i-1000)*A[i]/2;
      q += (i-1000)*A[i]/2;
    }
    else
    {
      if(f%2!=0)
        p += (i-1000);
      else
        q += (i-1000);
      p += (i-1000)*(A[i]-1)/2;
      q += (i-1000)*(A[i]-1)/2;
    }
    f += A[i]-1;
  }
  if(p>q)
    printf("%d\n",p-q);
  else
    printf("%d",q-p);
  system("PAUSE");
  return 0;
}
```

X 形图

自相似的图样被称为分形,例如如下图形:

它的基本形(深度为 1)是 X

由它自身组成深度为 2 的图形是

X X

X

X X

由深度为 n-1 的图形组成的深度为 n 的图形,是按照如下规则组成的:

B(n-1) B(n-1)

B(n-1)

B(n-1) B(n-1)

输入

第一行为测试数据的个数 T(T<=20)

每组输入数据一行,为图形深度(不超过 7)

输出

对于每组输入数据,输出给定图形.每组数据后有一个空行

样例输入

2

2

3

样例输出

X X

X

X X

X X X X

X X

X X X X

X X

X

X X

X X X X

X X

X X X X

提示

注意换行和空格,

样例没有对齐，可以把它复制出来到记事本查看

递归法

```
#include"math.h"
#include"string.h"
#include"stdio.h"

char a[750][750];
int f(int b,int p,int q)
{
	int m=(int)pow(3,b-1);
	int n=m/3,l=2*m/3;
	if(b==1)
	{
		a[p][q]='X';
		return 0;
	}
	else if(b>1)
	{
		f(b-1,p,q);
		f(b-1,p,l+q);
		f(b-1,l+p,q);
		f(b-1,n+p,n+q);
		f(b-1,l+p,l+q);
	}
	return 0;
}

int main()
```

```
{
        int h,e,x,y,z;
        scanf("%d",&e);
        for(x=0;x<e;x++)
        {
                scanf("%d",&h);
                memset(a,' ',sizeof(a));
           f(h,0,0);
                for(y=0;y<pow(3,h-1);y++)
                {
                        for(z=0;z<pow(3,h-1);z++)
                        {
                                printf("%c",a[z][y]);
                        }
                        printf("\n");
                }
                printf("\n");
        }
        return 0;
}
```

数学法

```
#include <stdio.h>
#include <string>
#include <math.h>
#define max 800
char chx[800][800];
void drawx(int n)
{
   int i,j,a;
   a=pow(3,n-1);
   for (i=0;i<a;i++)
   {
      for (j=0;j<a;j++)
```

```
        {
            printf("%c",chx[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}
int main()
{
    int n,m,i,j,k,a;
    memset(chx,' ',sizeof(chx));
    chx[0][0]='X';
    for (k=1;k<7;k++)
    {
        a=pow(3,k-1);
        for (i=0;i<a;i++)
        {
            for (j=0;j<a;j++)
            {
                chx[2*a+i][j]=chx[i][j];
                chx[a+i][a+j]=chx[i][j];
                chx[i][2*a+j]=chx[i][j];
                chx[2*a+i][2*a+j]=chx[i][j];
            }
        }
    }
    scanf("%d",&n);
    while (n--)
    {
        scanf("%d",&m);
        drawx(m);
    }
}
```

Lab 杯

Description

"Lab 杯"乒乓球赛就要在 PKU 的实验室之间举行了。人工智能实验室的学生都是乒乓球的狂热分子，都强烈希望代表实验室去比赛。但是有余名额限制，他们之中只能由一个人被选作代表。

为了让选择的过程公平，他们决定打一次单循环赛，每一对学生之间都打一场五局三胜的比赛。赢得最多比赛的人就将代表实验室去比赛。现在 Ava 手里有一份表，表里面记录了每一场比赛的比分。她应该让谁去比赛？

Input

输入包含一组测试数据。第一行包含 n（$2 \le n \le 100$），实验室里学生的数目。接下来给出一个 $n \times n$ 矩阵 A。矩阵的每一个元素都是 0、1、2、3 中的一个。第 i 行第 j 列的元素 a_{ij} 是第 i 个学生在和第 j 个学生的比赛中赢的局数。a_{ij} 和 a_{ji}（$i \ne j$）正好有一个是 3，另外一个小于 3。矩阵的所有对角线元素都是 0。

Output

输出赢了最多比赛的学生的编号。如果有平分，选择编号最小的。

Sample Input

```
4
0 0 3 2
3 0 3 1
2 2 0 2
3 3 3 0
```

Sample Output

```
4
```

```
#include<iostream>
using namespace std;
int a[100][100];
int b[100];
int main()
{
	int c;
	cin>>c;
	int i,j;
	for(i=0;i<c;i++)
		for(j=0;j<c;j++)
			cin>>a[i][j];
```

```
    for(i=0;i<c;i++)
                for(j=0;j<c;j++)
                {
                        if(a[i][j]==3)
                                a[i][i]++;
                }
    for(i=0;i<c;i++)
                b[i]=a[i][i];
    for(i=1;i<c;i++)
        {
                if(a[i-1][i-1]>a[i][i])
                {
                        a[i][i]=a[i-1][i-1];
                }
                else
                        continue;
        }
        for(i=0;i<c;i++)
        {
                if(b[i]==a[c-1][c-1])
                {
                        cout<<i+1<<endl;
                        break;
                }
        }
        return 1;
}
```

史上最难的问题

Time Limit: 1000MS **Memory Limit:** 10000K

Total Submissions: 9958 **Accepted:** 5672

Description

儒略•凯撒生活在充满危险和阴谋的年代，而其中最艰难的状况莫过于求得生存。于是他发明了最早的密码系统之一，用于军队的消息传递。
假设你是凯撒军团中的一名军官，需要把凯撒发送的消息破译出来，并提供给你的将军。消息加密的办法是：对消息原文中的每个字母，分别用该字母之后的第 5 个字母替换（例如：消息原文中的每个字母 A 都分别替换成字母 F）。而你要获得消息原文，也就是要将这个过程反过来。

密码字母：A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
原文字母：V W X Y Z A B C D E F G H I J K L M N O P Q R S T U

注意：只有字母会发生替换，其他非字母的字符不变，并且消息原文的所有字母都是大写的。

Input

最多不超过 100 个数据集组成，每个数据集之间不会有空行，每个数据集由 3 部分组成：

1. 起始行：START
2. 密码消息：由 1 到 200 个字符组成一行，表示凯撒发出的一条消息.
3. 结束行：END

在最后一个数据集之后，是另一行：ENDOFINPUT

Output

每个数据集对应一行，是凯撒的原始消息。

Sample Input

```
START
NS BFW, JAJSYX TK NRUTWYFSHJ FWJ YMJ WJXZQY TK YWNANFQ HFZXJX
END
START
N BTZQI WFYMJW GJ KNWXY NS F QNYYQJ NGJWNFS ANQQFLJ YMFS XJHTSI NS WTRJ
END
START
IFSLJW PSTBX KZQQ BJQQ YMFY HFJXFW NX RTWJ IFSLJWTZX YMFS MJ
END
ENDOFINPUT
```

Sample Output

```
IN WAR, EVENTS OF IMPORTANCE ARE THE RESULT OF TRIVIAL CAUSES
I WOULD RATHER BE FIRST IN A LITTLE IBERIAN VILLAGE THAN SECOND IN ROME
DANGER KNOWS FULL WELL THAT CAESAR IS MORE DANGEROUS THAN HE
```

```
#include<iostream>
#include<string>
```

```
using namespace std;
char str[200];
char sw[10];
char ST[6]={"START"};
char EOI[11]={"ENDOFINPUT"};
char EN[4]={"END"};
int k;
int main()
{
	while(gets(sw))
	{
		if(strcmp(sw,ST)==0)
		{
			gets(str);
			gets(sw);
		   if(strcmp(sw,EN)==0)
			{
				for(k=0;k<200;k++)
				{
					if(str[k]==13)
				break;
		else if(str[k]>=65&&str[k]<=90)
					{
			str[k]=str[k]-5;
			if(str[k]<65)
				str[k]=str[k]+26;
					}
					else
						continue;
				}
				puts(str);
			}
			else
```

```
                    return -1;
            }
            else if(strcmp(sw,EOI)==0)
                    break;
            else
                    return -1;
        }
        return 1;
}
```

渊子赛马

赛马是一古老的游戏，早在公元前四世纪的中国，处在诸侯割据的状态，历史上称为“战国时期”。在魏国作官的孙膑，因为受到同僚庞涓的迫害，被齐国使臣救出后，到达齐国国都。
赛马是当时最受齐国贵族欢迎的娱乐项目。上至国王，下到大臣，常常以赛马取乐，并以重金赌输赢。田忌多次与国王及其他大臣赌输赢，屡赌屡输。一天他赛马又输了，回家后闷闷不乐。孙膑安慰他说：“下次有机会带我到马场看看，也许我能帮你。”
孙膑仔细观察后发现，田忌的马和其他人的马相差并不远，只是策略运用不当，以致失败。
比赛前田忌按照孙膑的主意，用上等马鞍将下等马装饰起来，冒充上等马，与齐王的上等马比赛。第二场比赛，还是按照孙膑的安排，田忌用自己的上等马与国王的中等马比赛，在一片喝彩中，只见田忌的马竟然冲到齐王的马前面，赢了第二场。关键的第三场，田忌的中等马和国王的下等马比赛，田忌的马又一次冲到国王的马前面，结果二比一，田忌赢了国王。
就是这么简单，现在渊子也来赛一赛马。假设每匹马都有恒定的速度，所以速度大的马一定比速度小的马先到终点（没有意外！！）。不允许出现平局。最后谁赢的场数多于一半(不包括一半)，谁就是赢家(可能没有赢家)。渊子有 N(1≤N≤1000)匹马参加比赛。对手的马的数量与渊子马的数量一样，并且知道所有的马的速度。聪明的你来预测一下这场世纪之战的结果，看看渊子能否赢得比赛。

输入

输入有多组测试数据。
每组测试数据包括 3 行：
第一行输入 N(1≤N≤1000)。表示马的数量。
第二行有 N 个整型数字，即渊子的 N 匹马的速度。
第三行有 N 个整型数字，即对手的 N 匹马的速度。
当 N 为 0 时退出。

输出

若通过聪明的你精心安排，如果渊子能赢得比赛，那么输出“YES”。
否则输出“NO”。

样例输入

```
5

2 3 3 4 5

1 2 3 4 5

4

2 2 1 2
```

```
2 2 3 1
0
```

样例输出

```
YES
NO
```

```
#include<stdio.h>
void qsort(long array[],int left,int right)
{
	int l,r,k;
	if(left<right)
	{
		l=left;
		r=right;
		k=array[l];
		do
		{
			while((l<r)&&(array[r]>=k))
				r=r-1;
			if(l<r)
			{
				array[l]=array[r];
				l=l+1;
			}
			while((l<r)&&(array[l]<=k))
				l=l+1;
			if(l<r)
			{
				array[r]=array[l];
		r=r-1;
			}
		}while(l!=r);
		array[l]=k;
		qsort(array,left,l-1);
```

```
                qsort(array,l+1,right);
        }
}
int main()
{
        long a[1001],b[1001],N;
        int flag;
        int i;
        do
        {
                scanf("%d",&N);
                if(N!=0)
                {
                        for(i=0;i<N;i++)
                                scanf("%d",&a[i]);
                        for(i=0;i<N;i++)
                                scanf("%d",&b[i]);
        qsort(a,0,N-1);
                        qsort(b,0,N-1);
                        if(N%2==1)
                        {
                                flag=1;
                                for(i=N-1;i>N/2-1;i--)
                                {

                                if(a[i]<=b[i-N/2])
                                        {
                                                printf("NO\n");
                                                flag=0;
                                                break;
                                        }
                                }
                                if(flag==1)
```

```
                printf("YES\n");
        }
        if(N%2==0)
        {
                flag=1;
                for(i=N-1;i>=N/2-1;i--)
                {

                if(a[i]<=b[i-N/2+1])
                        {
                                printf("NO\n");
                                flag=0;
                                break;
                        }
                }
                if(flag==1)
                        printf("YES\n");
        }
    }
  }while(N!=0);
  return 0;
}
```

简单密码破解

密码是我们生活中非常重要的东东，我们的那么一点不能说的秘密就全靠它了。哇哈哈.
接下来渊子要在密码之上再加一套密码，虽然简单但也安全。
假设渊子原来一个 BBS 上的密码为 zvbo941987,为了方便记忆，他通过一种算法把这个密码变换成 YUANzi1987，这个密码是他的名字和出生年份，怎么忘都忘不了，而且可以明目张胆地放在显眼的地方而不被别人知道真正的密码。
他是这么变换的，大家都知道手机上的字母： 1--1， abc--2, def--3, ghi--4, jkl--5, mno--6, pqrs--7, tuv--8 wxyz--9, 0--0,就这么简单，渊子把密码中出现的小写字母都变成对应的数字，数字和其他的符号都不做变换，声明：密码中没有空格，而密码中出现的大写字母则边成小写之后往后移一位，如：X，先边成小写，再往后移一位，不就是 y 了嘛，简单吧。记住，z 往后移是 a 哦。

输入

输入包括多个测试数据。输入是一个明文，密码长度不超过 100 个字符，输入直到文件结尾。

输出

输出渊子真正的密文。

样例输入

```
YUANzi1987
```

样例输出

```
zvbo941987
```

题目来源

```
#include<stdio.h>
#include<string.h>
int main()
{
	char a[101];
	while(gets(a))
	{
	for(int i=0;i<strlen(a);i++)
	{
		if(a[i]>=97&&a[i]<=99)
			a[i]='2';
		if(a[i]>=100&&a[i]<=102)
			a[i]='3';
		if(a[i]>=103&&a[i]<=105)
			a[i]='4';
		if(a[i]>=106&&a[i]<=108)
			a[i]='5';
		if(a[i]>=109&&a[i]<=111)
			a[i]='6';
		if(a[i]>=112&&a[i]<=115)
			a[i]='7';
		if(a[i]>=116&&a[i]<=118)
			a[i]='8';
		if(a[i]>=119&&a[i]<=122)
			a[i]='9';
		if(a[i]>=65&&a[i]<=89)
```

```
                a[i]=a[i]+33;
            if(a[i]==90)
                a[i]=97;
        }
        puts(a);
        }
        return 0;
}
```

英文金曲大赛

我们在"渊子数"的题目中已经了解了渊子是个什么样的人了，他在大一的时候参加过工商学院的"英语聚乐部"。告诉你个秘密，这个俱乐部是个好地方，不但活动精彩而且有 MM。
这不，英语俱乐部举办了一个叫做"英文金曲大赛"的节目。这个节目有好多人参加，这不，成绩出来了，渊子当是很勇敢，自告奋勇接下了算出大家的总得分的任务。
当时有 7 个评委，每个评委都要给选手打分，现在要求去掉一个最高分和去掉一个最低分，再算出平均分。结果精确到小数点后两位。

输入

测试数据包括多个实例。
每组数据包括 7 个实数，代表评委们对该选手的评分。紧接着是选手的名字，名字的长度不超过 30 个字符。
输入直到文件结束。

输出

算出每位选手名字和最终得分，结果保留两位小数。

样例输入

```
10 10 10 10 10 10 9 xiaoyuanwang
0 0 0 0 0 0 0 beast
```

样例输出

```
xiaoyuanwang 10.00
beast 0.00
```

```
#inclde <stdio.h>
int main()
{
   double sum, a[7];
   char s[31];
   while(scanf("%lf%lf%lf%lf%lf%lf%lf%s", &a[0], &a[1], &a[2], &a[3], &a[4], &a[5], &a[6], s) !=
EOF)
```

```
    {
        sum = a[0];
        int i;
        double min1 = a[0], max1 = a[0], ave;
        for(i = 1; i < 7; i++)
        {
            if(min1 > a[i])
                min1 = a[i];
            if(max1 < a[i])
                max1 = a[i];
            sum += a[i];
        }
        ave = (sum - min1 - max1) / 5;
        printf("%s %.2lf\n", s, ave);
    }
    return 0;
}
```

TOJ 1051 A × B problem（大数相乘）

A × B problem

Redraiment 碰到了一个难题，需要请你来帮忙：给你两个整数，请你计算 A × B。

输入

数据的第一行是整数 T(1 ≤ T ≤ 20)，代表测试数据的组数。
接着有 T 组数据，每组数据只有一行，包括两个非负整数 A 和 B。
但 A 和 B 非常大，Redraiment 能保证这些数用 long 来保存一定会溢出。
但 A 和 B 的位数最大不会超过 100 位。

输出

对应每组测试数据，你都要输出两行：
第一行为："Case #:", # 代表这是第几组测试数据。
第二行是一个等式："A * B = Sum", Sum 代表 A × B 的结果。
你要注意这个等式里包含了几个空格。
要求每组数据之间都需要保留一个空行。

样例输入

```
2
1 2
123456789 987654321
```

样例输出

```
Case 1:
1 * 2 = 2

Case 2:
123456789 * 987654321 = 121932631112635269
```

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
   int t;
   char a[101],b[101];
   int c[10001];
         int y;
         int i;
   cin>>t;int h=1;
   while(t--)
   {
      cin>>a>>b;
      int l1=strlen(a);
      int l2=strlen(b);
      int m=0;int d=0;int k;
      for(i=0;i<10000;i++)c[i]=0;
      for(i=l1-1,y=0;i>=0;i--,y++)
      {
         d=0;
         m=y;
         for(int j=l2-1;j>=0;j--)
           {
              k=(a[i]-'0')*(b[j]-'0')+d+c[m];
              c[m++]=k%10;
              d=k/10;
```

```
            }
            while(d>0)
            {
               c[m++]+=d%10;
               d/=10;
            }
        }
        cout<<"Case "<<h<<':'<<endl;
        cout<<a<<" * "<<b<<' '<<'='<<' ';
        for( i=m-1;i>=0;i--)if(c[i]!=0){m=i;break;}
        if(i<0)cout<<0;
        else
        {
           for(i=m;i>=0;i--)cout<<c[i];
        }
        if(t!=0)cout<<endl<<endl;
        h++;
    }
    return 0;
}
```

字符统计

给出一串字符，要求统计出里面的字母、数字、空格以及其他字符的个数。
字母:A, B, ..., Z、a, b, ..., z 组成
数字:0, 1, ..., 9
空格:" "(不包括引号)
剩下的可打印字符全为其他字符。

输入

测试数据有多组。
每组数据为一行(长度不超过 100000)。
数据至文件结束(EOF)为止。

输出

每组输入对应一行输出。
包括四个整数 a b c d，分别代表字母、数字、空格和其他字符的个数。

样例输入

```
A0 ,
```

样例输出

```
1 1 1 1
```

```
#include<stdio.h>
#include<ctype.h>
int main()
{
   char str[100001],ch;
   while(gets(str))
   {
      long a=0,b=0,c=0,d=0,i=0;
      for(i=0;str[i]!='\0';i++)
      {
         if(isalpha(str[i]))
                                    a++;
         else if(isdigit(str[i]))
                                    b++;
         else if(str[i]==' ')
                                    c++;
         else
                                    d++;
      }
      printf("%ld %ld %ld %ld\n",a,b,c,d);
   }
   return 0;
}
```

漂亮菱形

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

上面的菱形漂亮吗？

现给出菱形的高度，要求你打印出相应高度的菱形，比如上面的菱形高度为 7。

输入

测试数据包括多行，每行 1 个整数 h，h 为奇数，代表菱形的高度。
输入以 0 结束。

输出

输出每组对应的菱形。

样例输入

```
1
7
0
```

样例输出

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

```
#include<iostream>
using namespace std;
int main()
{
	int n;
	int i,j,k;
	while(cin>>n,n)
	{
		for(i=0;i<(n+1)/2;i++)
		{
			for(j=(n-1)/2-i;j>0;j--)
			{
				cout<<' ';
			}
			for(k=0;k<1+2*i;k++)
			{
```

```
                cout<<'*';
            }
            cout<<endl;
        }
        for(i=0;i<(n-1)/2;i++)
        {
            for(j=0;j<1+i;j++)
            {
                cout<<' ';
            }
            for(k=0;k<n-2*(i+1);k++)
            {
                cout<<'*';
            }
            cout<<endl;
        }
    }
    return 0;
}
```

C 语言实验题—矩阵转置

```
#include<iostream>
using namespace std;
int main()
{
    int N;
    int i,j;
    int a[101][101];
    cin>>N;
    for(i=0;i<N;i++)
    {
        for(j=0;j<N;j++)
        {
```

```
                cin>>a[i][j];
            }
        }
        for(i=0;i<N;i++)
        {
            for(j=0;j<N;j++)
            {
                cout<<a[j][i];
                if(j<N-1)
                    cout<<' ';
            }
            cout<<endl;
        }
        return 0;
}
#include"math.h"
#include"string.h"
#include"stdio.h"
char a[750][750];
int f(int b,int p,int q)
{
    int m=(int)pow(3,b-1);
    int n=m/3,l=2*m/3;
    if(b==1)
    {
        a[p][q]='X';
        return 0;
    }
    else if(b>1)
    {
        f(b-1,p,q);
        f(b-1,p,l+q);
```

```
                f(b-1,l+p,q);
                f(b-1,n+p,n+q);
                f(b-1,l+p,l+q);
        }
        return 0;
}

int main()
{
        int h,e,x,y,z,m;
        scanf("%d",&e);
        for(x=0;x<e;x++)
        {
                scanf("%d",&h);
                memset(a,' ',sizeof(a));
            f(h,0,0);
                m=pow(3,h-1);
                for(y=0;y<m;y++)
                {
                        for(z=0;z<m;z++)
                        {
                                printf("%c",a[z][y]);
                        }
                        printf("\n");
                }
                printf("\n");
        }
        return 0;
}
```

数字对

输入 N（2≤N≤100）个数字，每个数字在 0 与 9 之间，根据输入的数字对，统计出该数字对出现的次数，比如 N=20 时，下面的数字中：0 1 5 9 8 7 2 2 2 3 2 7 8 7 8 7 9 6 5 9，数字对（7，8）=2 （8，7）=3。

输入

输入的第一行为 N，第二行为 N 个数字。第三行为数字对的个数 M，接下来是 M 行数据，每行为一个数字对。相邻数字之间均用空格分开。

输出

输出数字对以及每个数字对出现的次数，格式如下：

(7,7)=2

如果没有找到数字对，请输出

Not Found！

样例输入

```
20
0 1 5 9 8 7 2 2 2 3 2 7 8 7 8 7 9 6 5 9
3
7 8
8 7
9 0
```

样例输出

```
(7,8)=2
(8,7)=3
Not Found!
```

```
#include<iostream>
using namespace std;
int main()
{
        int N,M;
        int i,j;
        char num[101];
        char tnum[100][2];
        int result[100];
```

```
        cin>>N;
        for(i=0;i<N;i++)
        {
                cin>>num[i];
        }
        cin>>M;
        for(i=0;i<M;i++)
        {
                cin>>tnum[i][0]>>tnum[i][1];
                result[i]=0;
        }
        for(i=0;i<N-1;i++)
        {
                for(j=0;j<M;j++)
                {
                        if(num[i]==tnum[j][0]&&num[i+1]==tnum[j][1])
                                result[j]++;
                }
        }
        for(i=0;i<M;i++)
        {
                if(result[i]!=0)
                        cout<<'('<<tnum[i][0]<<','<<tnum[i][1]<<")="<<result[i]<<endl;
                else
                        cout<<"Not Found!"<<endl;
        }
        return 0;
}
```

数塔问题
7 3 8

8 1 0

2 7 4 4

4 5 2 6 5

(Figure 1)

Figure 1 shows a number triangle. Write a program that calculates the highest sum of numbers passed on a route that starts at the top and endssomewhere on the base. Each step can go either diagonally down to the left or diagonally down to the right.

输入

Your program is to read from standard input. There are several test cases.For each case ,the first line contains one integer N: the number of rows in the triangle. The following N lines describe the data of the triangle. The number of rows in the triangle is > 1 but <= 100. The numbers in the triangle, all integers, are between 0 and 99. The end of input is indicated by N = 0 .

输出

Your program is to write to standard output. The highest sum is written as an integer.

样例输入

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

样例输出

```
30
```

```
#include<stdio.h>
int a[101],b[101];
int main(){
   int n,i,j,temp,*p1,*p2,*t;
   while(1){
      scanf("%d",&n);
      if(n==0)break;
      p1=a;
      p2=b;
      for(p1[1]=0,i=1;i<=n;i++)
      {
         for(j=1;j<=i;j++)
         {
            scanf("%d",&temp);
            if(j==1){
               p2[1]=p1[1]+temp;
            }
```

```
            else if(j==i){
                p2[j]=p1[j-1]+temp;
            }
            else if(j>1){
                p2[j]=p1[j-1]+temp;
                if(p2[j]<p1[j]+temp)
                    p2[j]=p1[j]+temp;
            }
        }
        t=p1;
        p1=p2;
        p2=t;
    }
    for(temp=p2[1],i=2;i<=n;i++){
        if(temp<p1[i])
            temp=p1[i];
    }
    printf("%d\n",temp);
  }
  return 0;
}
```

第 K 小的数

你需要在 N 个数字中找到第 K 个小的数字

输入

多组数据。

每组数据第一行是两个整数 N,K(N<=5000000 ,K<=N). 下面一行是读入 N 个数字

读到 N=0 K=0时 结束

输出

对于每个输入，输出第 K 小的数字

样例输入

```
5 3
6 2 5 1 4
7 2
1 2 3 4 5 6 7
20 13
1 7 8 14 19 17 2 12 13 20 16 3 9 10 15 6 18 11 4 5
0 0
```

样例输出

```
4
2
13
```

提示

Please use scanf instead of cin or you will Time Limit Exceeded.

```
#include <iostream>
#include <algorithm>
using namespace std;
int a[5000001];
int main()
{
   int N,K,i;
   while(scanf("%d%d",&N,&K))
   {
      if(N*K==0)
         break;
      for(i=0;i<N;i++)
         scanf("%d",&a[i]);
      sort(a,a+N);
      printf("%d\n",a[K-1]);
   }
   return 0;
}
```

```
#include<stdio.h>
#include<stdlib.h>
int compare_integers(void const *a,void const *b)
{
   register int const *pa = a;
   register int const *pb = b;
   return *pa > *pb? 1:*pa < *pb ? -1:0;
}
int a[5000001];
int main(){
   int n,i,k;
   while(1){
      scanf("%d%d",&n,&k);
      if(n*k==0)break;
      for(i=0;i<n;i++)
         scanf("%d",&a[i]);
      qsort(a,n,sizeof(int),compare_integers);
      printf("%d\n",a[k-1]);
   }
   return 0;
}
```

可怜的毅毅

毅毅刚刚上小学三年级。有一天，老师给他出了一道十进制数的加法题目，1＋1=几。结果毅毅做错了，老师很不高兴，罚毅毅回家做 N 道大整数的加法（最大有80位）。可怜的毅毅需要你的帮助.....

输入

本题有多组测试数据，每组测试数据有两行，分别表示两个非负整数。

输出

对每组测试数据输出一行，表示两数之和。

样例输入

```
1
1
123
456
```

样例输出

```
2
579
```

提示

Leading zeros should be ignored.

```
#include<stdio.h>
#include<string.h>
int main(){
   char a[81],b[81];
   int c[82],i,la,lb;
   while(scanf("%s%s",a,b)!=EOF){
      la=strlen(a);
      lb=strlen(b);
      c[0]=(la>lb?la:lb);
      for(i=1;i<=c[0]+1;i++)
         c[i]=0;
      for(i=1;i<=c[0];i++){
         c[i]+=a[--la]+b[--lb]-'0'-'0';
         if(c[i]>=10){
            c[i]-=10;
            c[i+1]++;
         }
         if(la<=0||lb<=0)break;
      }
      while(la>0)c[++i]+=a[--la]-'0';
      while(lb>0)c[++i]+=b[--lb]-'0';
      if(c[c[0]+1]!=0)printf("%d",c[c[0]+1]);
      for(i=c[0];i>=1;i--)printf("%d",c[i]);
      printf("\n");
   }
   return 0;
}
```

比较字符串

你得到一些长度相等的字符串，现在你需要比较这些字符串，如果它们中间有某些位置的字符不一样，则需要用'?'来代替。
比如 "contest.info" 和 "content.info"，比较出来的串是"conte?t.info"。

输入

第一行输入一个数字 n，表示一共有多少字符串（n<=50）
下面 n 行，每行输入一个字符串，它们的长度都相等（字符串长度也<=50）

输出

输出一行变换后的串

样例输入

```
2
contest.txt
context.txt
```

样例输出

conte?t.txt

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char a[51][50];
char* compare(int n){
   int i,j,length;
   char *result;
   length=strlen(a[1]);
   result=(char*)malloc((length+1)*sizeof(char));
   strcpy(result,a[1]);
   for(i=2;i<=n;i++){
      for(j=0;j<=length;j++)
         if(a[i][j]!=result[j])
            result[j]='?';
   }
   return result;
}
int main(){
   int n,i;
   scanf("%d",&n);
   for(i=1;i<=n;i++)
      scanf("%s",a[i]);
   printf("%s\n",compare(n));
   return 0;
}
```

做一个正气的南航人

做人要有一身正气，南航学子都应该如此。比如我们今天的考试就应该做到"诚信"为上。
每次考试的第一个题目总是很简单，今天也不例外，本题是要求输出指定大小的"NUAA"字符串，特别地，为了体现"正气"二字，我们要求输出的字符串也是正方形的（行数和列数相等）。

输入

输入的第一行包含一个正整数 N（N<=20），表示一共有 N 组数据，接着是 N 行数据，每行包含一个正整数 M（M<=20），表示一行内有 M 个"NUAA"相连。

输出

输出指定大小的方形字符串，输出格式参见样例数据。

样例输入

```
2
1
2
```

样例输出

```
NUAA
NUAA
NUAA
NUAA
NUAANUAA
NUAANUAA
```

NUAANUAA
NUAANUAA
NUAANUAA
NUAANUAA
NUAANUAA
NUAANUAA

```
#include<stdio.h>
int main(){
   int i,j,n,m;
   scanf("%d",&n);
   while(n-->0){
      scanf("%d",&m);
      for(i=1;i<=m;i++){
         for(j=1;j<=4*m;j++){
            printf("NUAA");
            if(j%m==0)
               printf("\n");
         }
      }
   }
   return 0;
}
```

古韵之茶道

茶道者，烹茶饮茶之艺术也。沏茶、赏茶、饮茶，修身养性、淡泊明志之道也。其和谐之风，清静之韵，博文人雅士之心；怡匹夫小民之情；解日高路长之渴；去荒芜杂乱之念；显生活品悟之道。古人尝云："一饮涤昏寐，情思朗爽满天地；再饮清我神，忽如飞雨洒轻尘；三饮便得道，何须苦心破烦恼。"佳茗还须好法沏，水温茶量待斟酌。水温、茶量为茶质感之限。

输入

现有两数值，数值一为水温，其范围不过零至一百，用二进制表示；数据二即茶量（十进制），弗多于四千万有八。

输出

请君输出水温与茶量之积（十进制）。

样例输入

Input1:

```
0 0 0 0 1 1 0
180795
```

Input2:

```
0 1 1 0 1 0 0
12580
```

样例输出

Output1:

```
1084770
```

Output2:

654160

题目来源

古韵一

```
#include<stdio.h>
#include<string.h>
int a[7];
int fn(int n){
   int i,t=2;
   if(n==0)
      return 1;
   for(i=2;i<=n;i++)
      t*=2;
   return t;
}
int toten(int *a){
   int r=0;
   int j;
   for(j=0;j<7;j++)
      r+=a[j]*fn(6-j);
   return r;
}
int main(){
   int i;
   unsigned long n,t;
   for(i=0;i<7;i++)
      scanf("%d",&a[i]);
   scanf("%lu",&n);
   t=(unsigned long)toten(a);
   printf("%lu\n",n*t);
   return 0;
}
```

谁是组长

信息组需要选一个组长。信息组一共有 n 个人，分别用1到 n 编号，其中 m 个人参与了投票。得票数过半（票数大于 m div 2）的人将被选为组长。

输入数据将告知这 m 个人分别将票投给了谁，请统计出谁将担任信息组的组长。

输入

第一行两个数 n 和 m。

第二行有 m 个数，这些数都是不超过 n 的正整数，表明这 m 个人的选择。

1<=n<=maxlongint

1<=m<=10000

输出

输出将被选为组长的人。如果没有人的票数过半，请输出-1。

样例输入

```
7 4
7 7 2 7
```

样例输出

```
7
```

```
#include<stdio.h>
typedef struct People{
    long no;
    int n;
    int next;
}People;
People head[10000];
void Insert(long no,long count){
    int i;
    for(i=0;i<count;i++)
        if(head[i].no==no){
            head[i].n++;
            return;
        }
    head[i].no=no;
    head[i].n=1;
}
int main(){
    long n;
    int m,i,temp,count;
    scanf("%d%d",&n,&m);
    for(count=0,i=1;i<=m;i++){
        scanf("%d",&temp);
        Insert(temp,count);
        count++;
    }
    for(i=0;i<count;i++)
        if(head[i].n > m/2){
            printf("%d\n",head[i].no);
            break;
        }
    if(i >=count)
        printf("%d\n",-1);
    return 0;
}
```

念数字

编一个"念数字"的程序，它能让计算机完成以下工作：当你输入一个 0 至 99 之间的数后，计算机就会用汉语拼音印出这个数。
如果输入的数不在 0 到 99 之间，就印出"CUO LE"。
注：为了使不熟悉汉语拼音的同学也能做这个题，把"零，一，二，三，……，九，十"的拼音法写在下面。
零 LING 一 YI 二 ER 三 SAN 四 SI 五 WU
六 LIU 七 QI 八 BA 九 JIU 十 SHI

输入

输入数据有多组，每组数据占一行，内容为一个数字，数据以 EOF 作为结束。

输出

输出对应的汉语拼音，字母全部为大写。每组数据占一行

样例输入

```
35
0
11
100
```

样例输出

```
SAN SHI WU
LING
SHI YI
CUO LE
```

```
#include<iostream>
using namespace std;
int main()
{
	char py[11][5]={"LING","YI","ER","SAN","SI","WU","LIU","QI","BA","JIU","SHI"};
	int n;
	int g,s;
	while(cin>>n)
	{
		if(n<0||n>99)
		{
			cout<<"CUO LE"<<endl;
			continue;
		}
		g=s=0;
		g=n%10;
		s=n/10;
		if(s==0)
			cout<<py[g]<<endl;
		else if(s==1&&g==0)
			cout<<py[10]<<endl;
```

```
                else if(s==1&&g>0)
                        cout<<py[10]<<' '<<py[g]<<endl;
                else if(s>1&&g==0)
                        cout<<py[s]<<' '<<py[10]<<endl;
                else if(s>1&&g>0)
                        cout<<py[s]<<' '<<py[10]<<' '<<py[g]<<endl;
        }
        return 0;
}
```

More Beautiful

当老师不容易，尤其是当小学的老师更难:现在的小朋友做作业喜欢滥用括号。
虽然不影响计算结果，但不够美观，容易出错，而且可读性差。但又不能一棒子打死，也许他们就是将来的"陈景润"呢！
为了减轻老师的工作，不至于他们工作到深夜，我们来写个程序把小朋友的作业做一下简单地处理，去掉那些多余的括号。
为了简化问题，所有式子里只存在小括号，运算符号包括+(加)、-(减)、*(乘)、/(除)、^(幂)。
注意：去掉多余的小括号不是指运算结果一样就可以。
比如：(1+2)^1 = 3。虽然把括号去掉 1+2^1 也等于 3，但我们说这个括号不能去。
但如：1+(2+3) = 1+2+3 只要是允许的，因为加法是满足交换律和结合律的。

输入

输入包括多组测试数据。
每组测试数据为一行算术表达式，只包括数字和运算符号，长度小于 16。
输入以#行结束，该行不做处理。

输出

对应每组数据输入都有一行输出。
输出去掉多余的括号后的表达式。

样例输入

```
2*(1*(4*3))
3*(2*(4/1))
((4*2)/1)*3
(1)+(2)+(3)+(4)
1-(2-(3-4))
((3*2)*4)^1
#
```

样例输出

```
2*1*4*3
```

```
3*2*4/1
4*2/1*3
1+2+3+4
1-(2-(3-4))
(3*2*4)^1
//#include "stdafx.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
//#include "iostream.h"

int Operate(char str[16],int pos,int &start,int &end,bool bSign[4]);
void Hendle(char str[16],int start,int end,bool bSign[4]);
int main()
{
	int pos,start,end;
	int length=0;
	bool bSign[4];
	char str[16];
	while(gets(str),str[0]!='#')
	{
		pos=0;
		start=0;
		end=0;
		bSign[0]=bSign[1]=bSign[2]=bSign[3]=false;
		length=strlen(str);
	while(pos<length)
		{
			if(str[pos]=='(')
			{
			//start=pos;
			pos+=Operate(str,pos,start,end,bSign);
```

```
                        //start--;
                        Hendle(str,start,end,bSign);
                        }
                pos++;
                }
        for(int i=0;i<length;i++)
                {
                        if(str[i]!=' ')
                                printf("%c",str[i]);
                }
        printf("\n");
        }
        return 0;
}

int Operate(char str[],int pos,int &start,int &end,bool bSign[])
{
        int TemStart=0;
        bool bTemSign[4];
        //bSign[0]=bSign[1]=bSign[2]=bSign[3]=false;
        start=pos;
        pos++;
        while(str[pos]!=')')
        {
                if(str[pos]!='(')
                {
                        switch (str[pos])
                        {
                        case '+':
                                bSign[0]=true;
                                break;
                        case '-':
```

```
                                        bSign[1]=true;
                                        break;
                                case '*':
                                        bSign[2]=true;
                                        break;
                                case '/':
                                        bSign[3]=true;
                                        break;
                                }
                        }
                        else
                        {
                                TemStart=start;
                                bTemSign[0]=bTemSign[1]=bTemSign[2]=bTemSign[3]=false;
                                pos+=Operate(str,pos,TemStart,end,bTemSign);
                                Hendle(str,TemStart,end,bTemSign);
                        }
                        pos++;
                }
                end=pos;
                return end-start;
}


void Hendle(char str[],int start,int end,bool bSign[4])
{
                if(str[end+1]!='^')
                {
                        if(str[start-1]!='+'&&str[start-1]!='-'&&str[start-1]!='*'&&str[start-
1]!='/'&&str[start-1]!='^'&&
                                str[end+1]!='+'&&str[end+1]!='-
'&&str[end+1]!='*'&&str[end+1]!='/')
                                str[start]=str[end]=' ';
                        if(bSign[0]==false&&bSign[1]==false&&bSign[2]==false&&bSign[3]==false)
```

```
                    str[start]=str[end]=' ';
                if(str[end+1]=='*'||str[end+1]=='/'&&str[start-1]!='*'&&str[start-1]!='/')
                {
                    if(bSign[0]==false&&bSign[1]==false)
                        str[start]=str[end]=' ';
                }
                if(start-1<0&&(str[end+1]=='+'||str[end+1]=='-'))
                {
                    str[start]=str[end]=' ';
                }
                if(start-1>=0&&(str[end+1]=='+'||str[end+1]=='-')&&str[start-1]=='(')
                {
                    str[start]=str[end]=' ';
                }
                if(str[start-1]=='/')
                {
        if(bSign[0]==false&&bSign[1]==false&&bSign[2]==false&&bSign[3]==false)
                        str[start]=str[end]=' ';
                }
                if(str[start-1]=='-')
                {
                    if(bSign[0]==false&&bSign[1]==false)
                        str[start]=str[end]=' ';
                }
                if(str[start-1]=='*'&&str[end+1]!='*'&&str[end+1]!='/')
                {
                    if(bSign[0]==false&&bSign[1]==false)
                        str[start]=str[end]=' ';
                }
                if(str[start-1]=='+')
                    str[start]=str[end]=' ';
```

```
                //if(str[])
                        //str[start]=str[end]=' ';


        }
        else if(bSign[0]==false&&bSign[1]==false&&bSign[2]==false&&bSign[3]==false)
                str[start]=str[end]=' ';


}
```

两数组最短距离

已知元素从小到大排列的两个数组 x[]和 y[], 请写出一个程序算出两个数组彼此之间差的绝对值中最小的一个, 这叫做数组的距离。

输入

第一行为两个整数 m, n(1≤m, n≤1000), 分别代表数组 f[], g[]的长度。
第二行有 m 个元素, 为数组 f[]。
第三行有 n 个元素, 为数组 g[]。

输出

数组的最短距离

样例输入

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

样例输出

```
1
```

提示

你能想出 O(n+m)的算法吗?^_^
加油!

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

```
int m,n;
int i,j;
int k;
char f[1001],g[1001];
cin>>m>>n;
for(i=0;i<m;i++)
        cin>>f[i];
for(i=0;i<n;i++)
        cin>>g[i];
i=0;
j=0;
k=abs(f[0]-g[0]);
while(i<m&&j<n)
{
        if(f[i]-g[j]==0)
        {
                k=0;
                break;
        }
        if(abs(f[i]-g[j]<k))
        {
                k=abs(f[i]-g[j]);
        }
        if(f[i]>g[j])
        {
                j++;
        }
        if(f[i]<g[j])
        {
                i++;
        }
}
```

```
        cout<<k<<endl;

        return 0;

}
```

等值数目

已知两个整数数组 f[]和 g[]，它们的元素都已经从小到大排列。例如 f[]中可能有 1，2，2，3，3，g[]中有 1，2，2，2，3。
请写一个程序，算出这两个数组彼此之间有多少组相同的数据。就以上例而言：
f[0]于 g[0]是第一组；
f[1]于 g[1]是第二组；
f[2]于 g[2]是第三组；
f[3]于 g[4]是第四组。

输入

第一行为两个整数 m, n(1≤m, n≤1000)，分别代表数组 f[], g[]的长度。
第二行有 m 个元素，为数组 f[]。
第三行有 n 个元素，为数组 g[]。

输出

输出等值数目。

样例输入

```
5 5

1 2 2 2 3

1 2 2 3 3
```

样例输出

```
4
```

```
#include<iostream>

using namespace std;

int main()

{

        int m,n;

        int i,j;

        int k=0;

        char f[1001],g[1001];

        cin>>m>>n;

        for(i=0;i<m;i++)
```

```
                cin>>f[i];
        for(i=0;i<n;i++)
                cin>>g[i];
        i=0;
        j=0;
        while(i<m&&j<n)
        {
                if(f[i]==g[j])
                {
                        k++;
                        i++;
                        j++;
                }
                if(f[i]>g[j])
                {
                        j++;
                }
                if(f[i]<g[j])
                {
                        i++;
                }
        }
        cout<<k<<endl;
        return 0;
}
```

FJ's N ($1 \le N \le 10{,}000$) cows conveniently indexed 1..N are standing in a line. Each cow has a positive integer height (which is a bit of secret). You are told only the height H ($1 \le H \le 1{,}000{,}000$) of the tallest cow along with the index I of that cow.

FJ has made a list of R ($0 \le R \le 10{,}000$) lines of the form "cow 17 sees cow 34". This means that cow 34 is at least as tall as cow 17, and that every cow between 17 and 34 has a height that is strictly smaller than that of cow 17.

For each cow from 1..N, determine its maximum possible height, such that all of the information given is still correct. It is guaranteed that it is possible to satisfy all the constraints.

输入

Line 1: Four space-separated integers: *N*, *I*, *H* and *R*
Lines 2..*R*+1: Two distinct space-separated integers *A* and *B* (1 ≤ *A*, *B* ≤ *N*), indicating that cow *A* can see cow *B*.

输出

Lines 1..*N*: Line *i* contains the maximum possible height of cow *i*.

样例输入

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

样例输出

```
5
4
5
3
4
4
5
5
```

```
#include<iostream>
using namespace std;
int a[10000][2]={0};
int fun(int m,int n,int k)
{
          for(int i=0;i<k;i++)
          {
                    if(m==a[i][0]&&n==a[i][1])
                              return 0;
          }
          return 1;
}
```

```
int main()
{
	int N,I,H,R;
	int h[10001]={0};
	int sum;
	int A,B;
	int i,j;
	int k=0;
	cin>>N>>I>>H>>R;
	for(i=0;i<R;i++)
	{
		cin>>A>>B;
		k++;
		if(fun(A,B,k))
		{
			if(A<=B)
			{
			a[i][0]=A;
			a[i][1]=B;
			h[A]--;
		       h[--B]++;
			}
		else
			{
			a[i][0]=A;
			a[i][1]=B;
			h[B]--;
			h[--A]++;
			}
		}
	}
	for(i=0;i<N;i++)
```

```
        {
                sum=0;
                for(j=0;j<=i;j++)
                {
                        sum=sum+h[j];
                }
                sum=sum+H;
                cout<<sum<<endl;
        }
        return 0;
}
```

数论

```
#include <stdio.h>

const int A[]={0,1,2,3,4,5,6,7,8,9,11,22,33,44,55,66,77,88,99}; // 如果得出的结果是100以下,直接求出来

const int B[]={0,9,18,108,198,1098,1998,10998,19998,109998};    // 记录的是结果的位数与它是第几个数的关系

const int C[]={0,10,10,100,100,1000,1000,10000};                // 求结果的除首尾部分要用到

reverse(int n,int m)
{
    int Arr[]={1,10,100,1000};
    long r,i,j;
    r=j=n;
    if(m == 1)
        return n;
    for(i=0 ; i<m/2 ; ++i)
        r *= 10;
    for(i=m/2 ; i>0 ; --i)
    {
        if(m%2 != 0 && i == m/2)
            j /= 10;
        r += (j%10)*Arr[i-1];
```

```
        j /= 10;
    }
    return r;
};
int main()
{
    long i,j,k,m,n,t,*N;
    scanf("%ld",&i);
    N = (long*)malloc(i*sizeof(long));
    if(!N)
        exit(-1);
    for(j=0; j<i ; ++j)
        scanf("%ld",&N[j]);              // 输入要计算的第几个结果的数
    for(j=0; j<i ; j++)
    {
        if(N[j] <= 18)                   // 如果要求的结果小于100,直接在数组中查找
        {
            printf("%ld\n",A[N[j]]);
            continue;
        }
        for(k=3; k<=9 ; ++k)
            if(N[j] <= B[k])
                break;                   // 求结果是共几位的数字,保存到k中
        t = (N[j] - B[k-1] -1)/C[k-2] + 1; // 为什么要减一?因为是数字的中间的对称部分是从零开始的,加一是因为前面已经有n个循环了,必须加一
        n=t;
        for(m=0 ; m<k-1 ; ++m)
            t *= 10;
        t += n;                          // 求得数字的首尾部分
        n=(N[j] - B[k-1] -1)%C[k-2];     // 求数字的中间对称部分的一半
        t = t + 10*reverse(n,k-2);
        printf("%ld\n",t);
```

```
    }
    free(N);
    return 0;
}
```

完美数

```
#include <stdio.h>
int main()
{
    int s[5]={6,28,496,8128,33550336};
    int n,m,i,j,k;
    while(scanf("%d %d",&n,&m)!=EOF)
    {
        if(n==0&&m==0)
                            break;
        for(i=0;i<5;i++)
          if(n<=s[i])
              break;
        for(j=0;j<5;j++)
          if(m<=s[j])
              break;
        if(j-i==0)
                            printf("No\n");
        else
        {
            for(k=i;k<j-1;k++)
                   printf("%d ",s[k]);
            printf("%d\n",s[k]);
        }
    }
    return 0;
}
```

亲和数

```
#include<iostream>
using namespace std;
int main()
{
	int M;
	int A,B;
	cin>>M;
	while(M--)
	{
		cin>>A>>B;
		if((A==220&&B==284)||(A==284&&B==220))
			printf("YES\n");
		else if((A==1184&&B==1210)||(A==1210&&B==1184))
			printf("YES\n");
		else if((A==2620&&B==2924)||(A==2924&&B==2620))
			printf("YES\n");
		else if((A==5020&&B==5564)||(A==5564&&B==5020))
			printf("YES\n");
		else if((A==6232&&B==6368)||(A==6368&&B==6232))
			printf("YES\n");
		else if((A==10744&&B==10856)||(A==10856&&B==10744))
			printf("YES\n");
		else if((A==12285&&B==14595)||(A==14595&&B==12285))
			printf("YES\n");
		else if((A==17296&&B==18416)||(A==18416&&B==17296))
			printf("YES\n");
		else if((A==63020&&B==76084)||(A==76084&&B==63020))
			printf("YES\n");
		else if((A==66928&&B==66992)||(A==66992&&B==66928))
			printf("YES\n");
		else if((A==67095&&B==71145)||(A==71145&&B==67095))
			printf("YES\n");
```

```
                else if((A==69615&&B==87633)||(A==87633&&B==68615))
                        printf("YES\n");
                else if((A==79750&&B==88730)||(A==88730&&B==79750))
                        printf("YES\n");
                else
                        printf("NO\n");
        }
}
```

输出 100->200 之间的素数的个数，以及所有的素数。

输入

无

输出

100->200 之间的素数的个数，以及所有的素数。

样例输入

样例输出

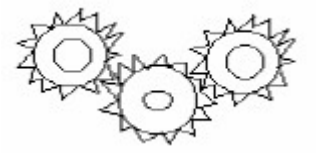
```
21
101 103 ... 197 199
#include<iostream.h>
#include<math.h>

int main()
{
        cout<<21<<endl;
        for(int i=100;i<=200;i++)
        {
                for(int j=sqrt(i);j>1;j--)
                {
                        if(i%j==0)
                                break;
                        else
                        {
```

```
                                        if(j==2)
                                        {
                                                if(i==199)
                                                {
                                                        cout<<i<<endl;
                                                }
                                                else
                                                {
                                                        cout<<i<<' ';
                                                }
                                        }
                                }
                        }
                }
        return 0;
}
```

三齿轮问题：三个齿轮啮合

如图在齿轮箱里三个齿轮互相衔接，某瞬间两对齿相遇，问各转多少圈后，这两对齿同时重逢。



输入

输入数据有多组，每组数据一行，每行为3个数a,b,c，分别代表三个齿轮的齿数（均为正整数）。数与数之间用空格隔开。当a，b，c中有一个为0时，输入结束。

输出

输出每组数据中，每个齿轮所转的圈数，用空格隔开。

样例输入

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

样例输出

```
1 1 1
1 1 1
#include<iostream>
using namespace std;
int gcd(int a,int b)
{
	if(b==0)
		return a;
	else
		return gcd(b,a%b);
}
int lcm(int a,int b)
{
	return a*b/gcd(a,b);
}
int main()
{
	int a,b,c;
	int d,e,f;
	int g;
	while(cin>>a>>b>>c)
	{
		if(a==0||b==0||c==0)
			break;
		g=lcm(lcm(a,b),c);
		d=g/a;
		e=g/b;
		f=g/c;
		cout<<d<<' '<<e<<' '<<f<<endl;
	}
	return 0;
```

```
}
```

寻找素数对描述

哥德巴赫猜想大家都知道一点吧.我们现在不是想证明这个结论,而是想在程序语言内部能够表示的数集中,任意取出一个偶数,来寻找两个素数,使得其和等于该偶数.
做好了这件实事,就能说明这个猜想是成立的.
由于可以有不同的素数对来表示同一个偶数,所以专门要求所寻找的素数对是两个值最相近的, 而且素数对中的第一个数不大于第二个数.

输入

输入中是一些偶整数 M(5<M<=10000).

输出

对于每个偶数,输出两个彼此最接近的素数,其和等于该偶数.

样例输入

```
20
30
40
```

样例输出

```
7 13
13 17
17 23
```

```
#include<stdio.h>
bool fun(int n)
{
        int a=2;
```

```
        while(a<=n-1)
        {
                if(n%a==0)
                {
                        return true;//合数
                }
                else
                {
                        a++;
                }
        }
        if(a==n)//素数
        {
                return false;
        }
}
int main()
{
        int m=0,i;
        int sushu1,sushu2;
        while(scanf("%d",&m)!=EOF)
        {
                for(i=0;i<=m/2;i++)
                {
                        if(fun(i)==false&&fun(m-i)==false)
                        {
                                sushu1=i;
                                sushu2=m-i;
                        }
                }
                printf("%d %d\n",sushu1,sushu2);
        }
```

```
        return 0;
}
```

Description

找出输入整数的所有因子（包括重复因子），并按从小到大的顺序依次输出。

Input

输入一组待分解整数，每个整数 k 占一行。

保证所有的输入数字 1 <= k < 2^21

Output

输出每个输入整数的所有因子（按因子从小到大的顺序输出），因子之间用空格隔开。

Sample Input

4

7

12

Sample Output

2 2

7

2 2 3

这道题问的是将他的所有素数因子求出来吧，例如 12 的话是 2 2 3

如果是这样的话，我的思路是：先打一个素数表，然后再求，这样的话效率会高很多，如果用暴力法的话 2^21 次方，大约 2000000 多万，也不会超时吧，不过太不优化了，我开的是 10000 以内的素数，其实开 2000 的就行了，因为 sqrt(2^21)<2200，下面是我的程序：

```
#include<stdio.h>
#include<memory.h>
int a[10001];
int pri[10000];
void prime()
{
 int i,j;
 memset(a,0,sizeof(a));
 for(i=2;i<=100;i++){
  if(!a[i]){
   for(j=2*i;j<=10000;j+=i)
    a[j]=1;
  }
```

```
 }
 for(i=0,j=2;j<=10000;j++)
  if(!a[j]) pri[i++]=j;
}
int main()
{
 int fac[1000],n,i,j,k;
 prime();
 while(scanf("%d",&n)==1){
  j=0;
  for(i=0;pri[i]*pri[i]<=n;i++){
   if(n%pri[i]==0){
    while(n%pri[i]==0){
     fac[j]=pri[i];
     j++;
     n/=pri[i];
    }
   }
  }
  if(n>1) fac[j++]=n;
  for(i=0;i<j;i++)
   if(i==j-1) printf("%d\n",fac[i]);
   else printf("%d ",fac[i]);
 }
}
```

如果你是在某个 OJ 做题的话，要注意一下我的输出格式，就是输出时的判断，那是正确的做法，否则按照你的写法，由于你最后元素输出时还有一个空格，会出现 Presentation Error 的

```
#include<stdio.h>
#include<memory.h>
int a[1000001];
int pri[1000000];
```

```
void prime()
{
        int i,j;
    memset(a,0,sizeof(a));
    for(i=2;i<=1000;i++)
        {
                if(!a[i])
                {
                for(j=2*i;j<=1000000;j+=i)
          a[j]=1;
                }
        }

    for(i=0,j=2;j<=1000000;j++)
        {
                if(!a[j])
                {
                        pri[i]=j;
                i++;
                }
        }
}
int main()
{
        long fac[1000],n,i,j,k,n1;
    prime();
        while(scanf("%d",&n)!=EOF)
        {
                if(n==1)
                {
                        printf("1 = 1\n");
                        continue;
```

```
            }
    memset(fac,0,sizeof(fac));
            n1=n;
            j=0;
            for(i=0;pri[i]*pri[i]<=n;i++)
            {
                    if(n%pri[i]==0)
                    {
                            while(n%pri[i]==0)
                            {
                                    fac[j]=pri[i];
            j++;
            n/=pri[i];
                            }
                    }
            }
            if(n>1)
                    fac[j++]=n;
            printf("%d = ",n1);
            int k;
            for(i=0;i<j;i++)
            {
                    if(fac[i]==fac[i+1])
                    {
                            k=1;
                            for(;i<j;i++)
                            {
                                    if(fac[i]==fac[i+1])
                                    {
                                            k++;
                                    }
                                    else
```

```
                                                {
                                                        if(i==j-1)
                                                        {

            printf("%d^%d\n",fac[i-1],k);
                                                        }
                                                        else
                                                        {
                                                                printf("%d^%d *
",fac[i-1],k);
                                                        }
                                                        break;


                                                }
                                        }
                                }
                                else if(i==j-1)
                                        printf("%d\n",fac[i]);
              else
                                        printf("%d * ",fac[i]);
                        }
            }
            return 0;
}
```

水仙花数

```
#include<iostream.h>

void main()
{
            int a,b,c;
            for(int i=100;i<=999;i++)
            {
```

```
                    c=i%10;
                    b=(i%100-c)/10;
                    a=(i-b*10-c)/100;
                    if(i==a*a*a+b*b*b+c*c*c)
                              cout<<i<<"="<<a<<"的立方+"<<b<<"的立方+"<<c<<"的立方
"<<"="<<a*a*a<<"+"<<b*b*b<<"+"<<c*c*c<<endl;
          }
```

10000 以内完数

```
#include<iostream.h>

void main()
{
          int sum;
          int j;
          int k;
          for(int i=1;i<=10000;i++)
          {
                    sum=0;
                    j=1;
                    do
                    {
                              if(i%j==0)
                              {
                                        sum=sum+j;
                              }
                              j++;
                    }while(j<i);
                    if(sum==i)
                    {
                              cout<<i;
                              cout<<" 它的因子是: ";
                              k=1;
```

```
                do
                {
                        if(i%k==0)
                                cout<<k<<",";
                        k++;
                }while(k<i);
                cout<<"它是这些因子的和.";
                cout<<endl;
        }

}
```

约数之和

给你一个数字 求它的所有约数的和。

比如12, 约数有1, 2, 3, 4, 6, 12 加起来是28

现在给你一个数字 I。 (1 <= I <= 1,000,000).

输入

一个数字 I

输出

约数之和

样例输入

12

样例输出

28

题目来源

USACO 2006 Open

源代码--1089

```
#include <stdio.h>
#include <math.h>
int main(){
   long i,n,temp,sum;
   scanf("%d",&n);
```

```
    temp=(int)sqrt(n);
    for(sum=0,i=1;i<=temp;i++){
        if(n%i==0){
            sum+=i;
            sum+=n/i;
        }
    }
    printf("%d\n",sum);
    return 0;
}

}
```

求解 GCD 描述

给出 n 个 1到200000的正整数 (1<=n <=100)

你需要求出这 n 个数字的最大公约数。

输入

第一行是数字 n

下面 n 行是需要求的 n 个数字

输出

输出这 n 个数字的最大公约数

样例输入

```
3
18
63
36
```

样例输出

```
9
```

题目来源

2008 NUAA 省赛

源代码--1102

```
#include<stdio.h>
#include<stdlib.h>
int a[100];
int compare(const void *a,const void *b){
    return *((int*)a) > *((int*)b);
}
```

```
int gcd(int a,int b){
   if(b==0)
      return a;
   else return gcd(b,a%b);
}
int main(){
   int n,i,g;
   scanf("%d",&n);
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   qsort(a,n,sizeof(int),compare);
   for(g=a[0],i=1;i<n;i++)
      g=gcd(g,a[i]);
   printf("%d\n",g);
   return 0;
}
```

传球游戏

描述

N 个人围圈玩传球游戏，开始时第一个人拿着球，每个人把球传给左手的第 K 个人。满足 1<=K<=N/2。求 K 的最大值，使得第一个人重新拿到球之前，每个人都拿过球。

输入

输入数据有多组，每组一个正整数 N（3<=N<=10^{200}）

输出

输出 K 的最大值。

样例输入

```
3
```

样例输出

```
1
```

题目来源

TOJ

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
          char a[200];
```

```
int b[200];
int alen;
int i,j;
while(gets(a))
{
        alen=strlen(a);
        if(a[(alen-1)]-'0'==1&&alen==1)
                cout<<0<<endl;
        else if(a[(alen-1)]-'0'==2&&alen==1)
                cout<<1<<endl;
        else
        {
                int d;
                if((a[(alen-1)]-'0')%2==0)
                {
                d=0;
    for(i=0;i<alen;i++)
                        {
                        b[i]=((a[i]-'0')+d*10)/2;
        d=((a[i]-'0')+d*10)%2;
                        }
        for(i=alen-1;i>=0;i--)
                        {
                if(b[i]!=0)
                                {
                                b[i]=b[i]-1;
                        break;
                                }
                else
                                {
                                        b[i]=9;
                                }
```

```
                    }
if(b[alen-1]%2==0)
                    {
                              for(i=alen-1;i>=0;i--)
                              {
                              if(b[i]!=0)
                                        {
                                        b[i]=b[i]-1;
                              break;
                                        }
                    else
                                        {
                              b[i]=9;
                                        }
                              }
for(i=0;i<alen;i++)
                              {
                              if(b[i]!=0)
                    break;
                              }
                    for(;i<alen;i++)
                              {
                              cout<<b[i];
                              }
          cout<<endl;
                    }
          else
                              {
for(i=0;i<alen;i++)
                              {
                    if(b[i]!=0)
                    break;
```

```
                                        }
                        for(;i<alen;i++)
                                        {
                                cout<<b[i];
                                        }
                        cout<<endl;
                                }
                        }
                else
                        {
                        d=0;
            for(i=0;i<alen;i++)
                                {
                                b[i]=((a[i]-'0')+d*10)/2;
                d=((a[i]-'0')+d*10)%2;
                                }
                for(i=0;i<alen;i++)
                                {
                                if(b[i]!=0)
                        break;
                                }
                        for(;i<alen;i++)
                                {
                                cout<<b[i];
                                }
                        cout<<endl;
                        }
                }
        }
        return 0;
}
```

Problem 2 黑色星期五

题目描述

13 号又量星期五，是一个不寻常的日子吗？

13 号在星期五比在其他日少吗？为了回答这个问题，写一个程序来计算在 n 年里 13 日落在星期一，星期二，……，星期日的次数。这个测试从 1900 年 1 月 1 日到 1900+n-1 年 12 月 31 日。n 是一个非负整数且不大于 400。

这里有一些你要知道的：

- 1900 年 1 月 1 日是星期一。
- 4，6，11 和 9 月有 30 天。其他月份除了 2 月有 31 天。闰年 2 月有 29 天，平年 2 月有 28 天。
- 年份可以被 4 整除的为闰年（1992=4*498 所以是闰年，但是 1990 年不是闰年）。
- 以上规则不适合世纪年。可以被 400 整除的世纪年为闰年，否则为平年，所以 1700，1800，1900，2100 是平年，而 2000 年是闰年。

输入

一个整数 n。

输出

七个在一行，且用一个个空格分开的整数，它们代表 13 日是星期六，星期日，星期一，……，星期五的次数。

输入示例

20

输出示例

36 33 34 33 35 35 34

```
#include<stdio.h>
int main()
{
        int n;
        int a[12]={31,28,31,30,31,30,31,31,30,31,30,31};
        int b[7];
        int sum;
        while(scanf("%d",&n)!=EOF)
        {
                sum=0;
```

```
            for(int k=0;k<7;k++)
                b[k]=0;
        for(int i=1900;i<1900+n;i++)
            {
            a[1]=28;
            if((i%4==0&&i%100!=0)||(i%100==0&&i%400==0))
                {
                a[1]=29;
                }
        for(int j=0;j<12;j++)
                {
                if(i==1900&&j==0)//第一年特殊处理
                    {
                    sum=sum+13;
                    b[sum%7]++;
                    }
                else if(j==0&&i!=1900)//1 月 13 好是 12 月 13 号加上 12 月的天数
                    {
                    sum=sum+a[11];
                    b[sum%7]++;
                    }
                else//本月的 13 好加上本月的日期就是下月的 13 号。
                    {
                    sum=sum+a[j-1];
                    b[sum%7]++;
                    }
                }
            }
        printf("%d %d %d %d %d %d %d\n",b[6],b[0],b[1],b[2],b[3],b[4],b[5]);
        }
        return 0;
}
```

杨辉三角

描述

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
上面的图形熟悉吗？它就是我们中学时候学过的杨辉三角。

输入

输入数据包含多组测试数据。
每组测试数据的输入只有一个正整数 n（1≤n≤30），表示将要输出的杨辉三角的层数。
输入以 0 结束

输出

对应于每一个输入，请输出相应层数的杨辉三角，每一层的整数之间用一个空格隔开，每一个杨辉三角后面加一个空行。

样例输入

2

3

0

样例输出

1

1 1

1

1 1

1 2 1

```
#include<stdio.h>

int main()

{

          int a[31][31],i,j;

          int n;

    while(scanf("%d",&n)!=EOF&&n!=0)

    {
```

```
                for(i=0;i<n;i++)
                {
                        for(j=0;j<=i;j++)
                        {
                                if(i==j||j==0)
                                        a[i][j]=1;
                                else
                                        a[i][j]=a[i-1][j]+a[i-1][j-1];
                                printf("%d",a[i][j]);
                                if(i==j)
                                        printf("\n");
                                else
                                        printf(" ");
                        }
                }
                printf("\n");
        }
        return 0;
}
```

删数问题

从 N 位数字串中删去 M 个数使

剩下的数字串所表示的数值最小.

```
#include<iostream.h>
#include<string.h>
void deleteS(char *p)
{
        while(*p)
        {
                *p=*(p+1);//将下一个指针的值赋给这个指针。
                p++;//指针前移, 相当与消除节点。
        }
```

```
}
int main()
{
        char S[240];
        int i,m;
        cout<<"请输入数字串:"<<endl;
        cin>>S;
        cout<<"请输入须删掉多少个数:"<<endl;
        do
        {
                cout<<"0<N<数字串的长度"<<endl;
                cin>>m;
        }
        while(m<=0||m>=(int)strlen(S));
        while(m>0)
        {
                i=0;
                while(S[i]<=S[i+1]&&i+1<strlen(S))//删数问题的关键算法，删数后让高位的
数字尽可能小。
                        i++;
                deleteS(&S[i]);
                m--;
        }
                while(S[0]=='0'&&strlen(S)!=1)//删除S串中高位的0;
                        deleteS(S);//将数字串的首地址传给指针
                cout<<S<<endl;
                return 0;
}
```

整数游戏

描述

众所周知，2个不同的数字可以组成2个两位数，3个不同的数字可以组成6个三位数，依此类推.....

对于一个由 N 个数字（1~N）组成的 N 位数，定义一个操作，该操作将返回比当前数大的所有数中最小的一个。如果当前数已经是最大数，则返回该数列中最小的一个。

现给定 N，询问经过 K 次该操作后的数是多少。初始状态为所有排列中最小的一个数据。

输入

本题有多组测试数据，每组测试数据只有一行，包含两个整数，分别代表 N、K。

(1<=N<=9; 1<=K<=2*10^9)

输出

对于每组测试数据，输出一行，表示经过 K 次操作后的数。

样例输入

```
1 10
3 2
```

样例输出

```
1
213
```

```
#include<stdio.h>
void move(long d[],int m,int n)
{
   int i,tmp;
   tmp=d[n];
   for(i=n;i>m;i--)
      d[i]=d[i-1];
   d[m]=tmp;
}

int main(){
   int n,i,m;
   long a[10],b[10],k,r,t;
   for(a[1]=1,i=2;i<=9;i++)
      a[i]=i*a[i-1];
   while(scanf("%d%d",&n,&k)!=EOF)
   {
      b[0]=n;
      for(i=1;i<=n;i++)
         b[i]=i;
      k%=a[n];
      do{
         for(i=1;i<=n;i++){
            if(k>=a[i]&&k<a[i+1])
            {
               break;
            }
         }
         m=n-i;
         t=k/a[i];
         move(b,m,m+t);
         k=k%a[i];
```

```
    }while(k);
    for(i=1,r=0;i<n;i++)
    {
       r+=b[i];
       r*=10;
    }
    r+=b[i];
    printf("%ld\n",r);
  }
  return 0;
}
```

整数游戏描述

众所周知，2 个不同的数字可以组成 2 个两位数，3 个不同的数字可以组成 6 个三位数，依此类推.....

对于一个由 N 个数字（1~N）组成的 N 位数，定义一个操作，该操作将返回比当前数大的所有数中最小的一个。如果当前数已经是最大数，则返回该数列中最小的一个。

现给定 N，询问经过 K 次该操作后的数是多少。初始状态为所有排列中最小的一个数据。

输入

本题有多组测试数据，每组测试数据只有一行，包含两个整数，分别代表 N、K。

(1<=N<=9; 1<=K<=2*10^9)

输出

对于每组测试数据，输出一行，表示经过 K 次操作后的数。

样例输入

```
1 10
3 2
```

样例输出

```
1
213
```

代码如下：

```
#include<cstdio>
#include<cstring>
int array[12]={0,1,2,3,4,5,6,7,8,9};
int result[12];
int fn[10]={1,1,2,6,24,120,720,5040,40320,362880};
int flag=0;
int find(int m)//查找第几小的数的函数
```

```
{       int ans=0;
    for(int t=1;t<10;t++)
        {       if(array[t]!=0)
                    ans++;
                if(ans==m)
                {       m=array[t];
            array[t]=0;
                    return m;
                }
        }
}
void fill(int l,int r)//填数字函数从小到大
{       if(l>r) return;
        int cur=1;
        for(int i=l;i<=r;i++)
        {       result[i]=find(cur);


        }
}
void fillr(int l,int r)//填数字从大到小
{       if(l>r) return;
        int cur=1;
        for(int i=r;i>=l;i--)
        {       result[i]=find(cur);

        }
}
void dfs(int n,int k,int s)//从右边往左数的第 s 个位置
{               int m;
                for(m=s;m>=1;m--)
                {       if(m*fn[s-1]<k)
                        {       m=m+1;
                            result[n+1-s]=find(m);
```

```
                    break;
                }
            }
            k=k-(m-1)*fn[s-1];
            if(k==0)//判断临界条件
            {fillr(n+1-s+1,n); return;
            }
            int i;
            for(i=s;i>=1;i--)
                if(k==fn[i])
                {   fill(n+1-s+1,n+1-i-1);//将上次之后的与 i 之前的空填上 从小到大取
                    fillr(n+1-i,n);//s 位置极其之后从大到小赋值
                    return;
                }
                else if(fn[i]<k)
                {   i=i+1;
                    fill(n+1-s+1,n-i);
                    break;//n+1-i-1=n-i;
                }
            dfs(n,k,i);//从右边数第 i 个位置

}
int main()
{   int n;
   int k;
while(scanf("%d %d",&n,&k)==2)
{   k=(k+1)%fn[n];
  if(k==0)//判断临界条件
        fillr(1,n);
  else
  {   for(int i=n;i>=1;i--)
            if(k==fn[i])//可以赋值无须进行多余判断
```

```
                {       for(int j=1;j<=n+1-i-1;j++)//阶乘之内的（从右数的第 i 位置）与左数从 1 到 n+1-i 位置 转换
                                result[j]=array[j];
                        for(int s=n+1-i;s<=n;s++)//s 位置极其之后从大到小赋值
                                result[s]=array[n+1+n-i-s];
                        break;
                }
                else if(fn[i]<k)
                {       i=i+1;
                        fill(1,n-i);//n+1-i-1=n-i;
                        dfs(n,k,i);
                        break;//从右边数第 i 个位置
                }
 }

        for(int t=1;t<=n;t++)
                printf("%d",result[t]);
        printf("\n");
        memset(result,0,sizeof(result));
        for(int g=0;g<=9;g++)
                array[g]=g;


}
        return 0;
}
```

红色标记的为后来修改的内容。

整数游戏

众所周知，2 个不同的数字可以组成 2 个两位数，3 个不同的数字可以组成 6 个三位数，依此类推.....

对于一个由 N 个数字（1~N）组成的 N 位数，定义一个操作，该操作将返回比当前数大的所有数中最小的一个。如果当前数已经是最大数，则返回该数列中最小的一个。

现给定 N，询问经过 K 次该操作后的数是多少。初始状态为所有排列中最小的一个数据。

输入

本题有多组测试数据，每组测试数据只有一行，包含两个整数，分别代表 N、K。

(1<=N<=9; 1<=K<=2*10^9)

输出

对于每组测试数据，输出一行，表示经过 K 次操作后的数。

样例输入

```
1 10
3 2
```

样例输出

```
1
213
```

代码如下：

```
#include<cstdio>
#include<cstring>
int array[12]={0,1,2,3,4,5,6,7,8,9};
int result[12];
int fn[10]={1,1,2,6,24,120,720,5040,40320,362880};
int flag=0;
int find(int m)//查找第几小的数的函数
{       int ans=0;
   for(int t=1;t<10;t++)
        {       if(array[t]!=0)
                        ans++;
                if(ans==m)
                {       m=array[t];
          array[t]=0;
                        return m;
                }
        }
}
void fill(int l,int r)//填数字函数从小到大
{       if(l>r) return;
        int cur=1;
```

```
        for(int i=l;i<=r;i++)
        {       result[i]=find(cur);


        }
}
void fillr(int l,int r)//填数字从大到小
{       if(l>r) return;
        int cur=1;
        for(int i=r;i>=l;i--)
        {       result[i]=find(cur);

        }
}
void dfs(int n,int k,int s)//从右边往左数的第 s 个位置
{               int m;
                for(m=s;m>=1;m--)
                {       if(m*fn[s-1]<k)
                        {       m=m+1;
                            result[n+1-s]=find(m);
                                break;
                        }
                }
                k=k-(m-1)*fn[s-1];
                if(k==0)//判断临界条件
                {fillr(n+1-s+1,n); return;
                }
                int i;
                for(i=s;i>=1;i--)
                        if(k==fn[i])
                        {       fill(n+1-s+1,n+1-i-1);//将上次之后的与 i 之前的空填上 从小到大取
                                fillr(n+1-i,n);//s 位置极其之后从大到小赋值
                                return;
                        }
```

```
                else if(fn[i]<k)
                {       i=i+1;
                        fill(n+1-s+1,n-i);
                        break;//n+1-i-1=n-i;
                }
            dfs(n,k,i);//从右边数第 i 个位置

}
int main()
{       int n;
   int k;
while(scanf("%d %d",&n,&k)==2)
{       k=(k+1)%fn[n];
 if(k==0)//判断临界条件
        fillr(1,n);
 else
 {       for(int i=n;i>=1;i--)
            if(k==fn[i])//可以赋值无须进行多余判断
            {       for(int j=1;j<=n+1-i-1;j++)//阶乘之内的（从右数的第 i 位置）与左数从 1 到 n+1-i 位置转换
                        result[j]=array[j];
                for(int s=n+1-i;s<=n;s++)//s 位置极其之后从大到小赋值
                        result[s]=array[n+1+n-i-s];
                break;
            }
            else if(fn[i]<k)
            {       i=i+1;
                    fill(1,n-i);//n+1-i-1=n-i;
                    dfs(n,k,i);
                    break;//从右边数第 i 个位置
            }
 }
```

```
        for(int t=1;t<=n;t++)
                printf("%d",result[t]);
        printf("\n");
        memset(result,0,sizeof(result));
        for(int g=0;g<=9;g++)
                array[g]=g;

}
        return 0;
}
```

红色标记的为后来修改的内容。

```
#include "stdio.h"
//#include "conio.h"
#define maxsize 100//只能寻找 100 之内的数
void Depart(int x)
{
    int a[maxsize],sum[maxsize],b[maxsize];
    int i,j,k;
    int flag=0;
    for(i=1;i<=(x+1)/2;i++)
    {
        a[i]=i;//初始化数列 a[i]
        b[i]=0;
    }
    for(k=1;k<=(x+1)/2;k++)
    {
        sum[k]=0;
        for(i=k;i<=(x+1)/2;i++)
        {
            sum[k]=sum[k]+a[i];//从每一个数开始往下加
            if(sum[k]==x)
            {
```

```
                b[k]=i;//记录相加的数列有几个数
                flag=1;
                break;
            }
        }
    }
    if(flag)//如果找到了这样的数列, 标记为1, 输出这些数列
    {
        for(i=1;i<=(x+1)/2;i++)
       {
          if(sum[i]==x)
          {
               for(k=i;k<=b[i];k++)
              {
                 printf("%d ",a[k]);
              }
               printf("\n");
          }
       }
     }
     else
     {
         printf("NONE");
     }

}
main()
{   int data;
    scanf("%d",&data);
    Depart(data);
    //getch();
}
```

组合问题

```
#include <stdio.h>

unsigned F(int n,int m)

{

  unsigned i,j,k;

  int
A[31]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30};

  int
B[31]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30};

  for(i=m+1;i<=n;++i)

      for(j=n-m;j>=1;--j)

      {

          if(A[i]%B[j]==0 && B[j] != 1)

          {

              A[i] /= B[j];

              B[j] = 1;

          }

      }

  for(j=1,i=m+1;i<=n;i++)

      j *= A[i];

  for(k=1,i=1;i<=n-m;++i)

      k *= B[i];

  return j/k;

}

int main(void)

{

    int n,m;

    while(scanf("%d %d",&n,&m)==2)

    {

        if(m==0 && n==0)

            break;

        printf("%d\n",F(n,m));

    }
```

```
    return 0;
}
```

圆桌会议

描述

ACM 集训队的队员在暑假集训时经常要讨论自己在做题中遇到的问题.每当面临自己解决不了的问题时,他们就会围坐在一张圆形的桌子旁进行交流,经过大家的讨论后一般没有解决不了的问题,这也只有 ACM 集训队特有的圆桌会议,有一天你也可以进来体会一下哦:),在一天在讨论的时候,Eddy 想出了一个极为古怪的想法,如果他们在每一分钟内,一对相邻的两个 ACM 队员交换一下位子,那么要多少时间才能得到与原始状态相反的座位顺序呢?(即对于每个队员，原先在他左面的队员后来在他右面，原先在他右面的队员在他左面),这当然难不倒其他的聪明的其他队友们,马上就把这个古怪的问题给解决了,你知道是怎么解决的吗?

输入

对于给定数目 N(1<=N<=32767)，表示有 N 个人,求要多少时间才能得到与原始状态相反的座位顺序(reverse)即对于每个人，原先在他左面的人后来在他右面，原先在他右面的人在他左面。

输出

对每个数据输出一行，表示需要的时间(以分钟为单位)

样例输入

```
4
5
6
```

样例输出

```
2
4
6
```

题目来源

HDOJ

题目上传者

crq

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

```
int main()
{
	int N;
	int result;
	int i;
	while(cin>>N)
	{
		result=0;
		if(N==2)
			result=1;
		if(N%2==0)
		{
			for(i=1;i<N/2;i++)
			{
				result=result+2*i;
			}
		}
		if(N%2==1)
		{
			for(i=1;i<N/2;i++)
			{
				result=result+2*i;
			}
			result=result+N/2;
		}
		cout<<result<<endl;
	}
	return 0;
}
```

题目：有一个整数 n，将 n 分解成若干个整数之和，问如何分解能使这些数的乘积最大，输出这个乘积 m。

讲解

1 无法分拆,

2 分拆为 1*1；

3 分拆为 1*2；

4 分拆为 2*2；

5 分拆为 2*3；

6 分拆为 3*3；

7 分拆为 3*4；

8 分拆为 3*3*2；

9 分拆为 3*3*3;

总结

显然，3 最好。

算法：

1. **当 n 可以被 3 整除时，就让 n 分解成 k 个 3，之后，调用一个算 3 的 k 次方的函数 p(k)，输出 p(k)。**
2. **当 n 除 3 余 1 时，可分解为 1，3，3，...，3。从使乘积最大的角度出发，我们不希望分解 1 来，这时，宁可让 1 与其中的一个 3 合并，为 4。接着调用 p(k)，但输出时再将原来减去的 4 当作一个乘数乘进来，即输出 4*p(k)。**
3. **当 n 除 3 余 2 时，可分解为 2，3，3，...，3。可以沿用上诉思路求 p(k)。最后输出时将 2 乘入，即输出 2*p(k)。**

PC 国大选

时间限制(普通/Java):1000MS/3000MS　　运行内存限制:65536KByte
总提交:236　　测试通过:90

描述

PC 国大选是按各州的投票结果来确定最终的结果的，如果得到超过一半的州的支持就可以当选，而每个州的投票结果又是由该州选民投票产生的，如果某个州超过一半的选民支持马奥巴，则他将赢得该州的支持。现在给出每个州的选民人数，请问马奥巴至少需要赢得多少选民的支持才能当选？

输入

多组输入数据
每组数据的第一行包括一个整数 N（1<=N<=101）,表示 PC 国的州数，N=0表示输入结束
接下来一行包括 N 个正整数，分别表示每个州的选民数，每个州的选民数不超过100

输出

对于每组数据输出一行，表示马奥巴至少需要赢得支持的选民数

样例输入

```
3
5 7 5
0
```

样例输出

```
6
```

提示

From YanHao

题目来源2008年南航 ACM 月赛

解题思路：

将所有州的超过一半的选民算出来，即 n=n/2+1；将所得的新数列按非降序排序，把前 n/2+1个州的人数总数求出来就是，他获胜所需的最少人数。

```
#include<stdio.h>
int a[102];
int main(){
   int i,j,n,temp,sum,min;
   while(scanf("%d",&n)){
      if(!n)
         break;
      for(i=1;i<=n;i++){
         scanf("%d",&temp);
         a[i]=temp/2+1;
      }
      for(i=1;i<n;i++){//选择法排序
         for(min=i,j=i+1;j<=n;j++)
         {
            if(a[min]>a[j])
               min=j;
         }
         if(min!=i){
            temp=a[min];
            a[min]=a[i];
            a[i]=temp;
         }
      }
      temp=n/2+1;
      for(sum=0,i=1;i<=temp;i++)
         sum+=a[i];
      printf("%d\n",sum);
   }
   return 0;
}
```

Home Work

描述

临近开学了，大家都忙着收拾行李准备返校，但 I_Love_C 却不为此担心!
因为他的心思全在暑假作业上：目前为止还未开动(-_-!!还以为他有多冷静呢)。

暑假作业是很多张试卷，我们这些从试卷里爬出来的人都知道，卷子上的题目有选择题、填空题、简答题、证明题等。
而做选择题的好处就在于工作量很少，但又因为选择题题目都普遍很长。
如果有 5 张试卷，其中 4 张是选择题，最后一张是填空题，很明显做最后一张所花的时间要比前 4 张长很多。
但如果你只做了选择题，虽然工作量很少，但表面上看起来也已经做了 4/5 的作业了。
I_Love_C 决定就用这样的方法来蒙混过关。

他统计出了做完每一张试卷所需的时间以及它做完后能得到的价值（按上面的原理，选择题越多价值当然就越高咯）。
现在就请你帮他安排一下，用他仅剩的一点时间来做最有价值的作业。

输入

测试数据包括多组。
每组测试数据以两个整数 M，N(1≤M≤20, 1≤N≤10000)开头，分别表示试卷的数目和 I_Love_C 剩下的时间。
接下来有 M 行，每行包括两个整数 T，V(1≤T≤N,0<V<10000)，分别表示做完这张试卷所需的时间以及做完后能得到的价值！
输入以 0 0 结束。

输出

对应每组测试数据输出 I_Love_C 能获得的最大价值。
保留小数点 2 位

样例输入

```
4 20

4 10

5 22

10 3

1 2

0 0
```

样例输出

```
37.00
```

提示

float 的精度可能不够。
你应该使用 double 类型。

题目来源

ZJGSU

题目上传者

crq

```
#include<iostream>
using namespace std;
struct HomeWork
{
          int T,V;
          double a;
};
HomeWork h[20];
int M,N;
double result;
void qsort(HomeWork array[],int left,int right)
{
          int l,r;
          HomeWork temp;
          if(left<right)
          {
                    l=left;
                    r=right;
               temp=array[l];
                    do
                    {
                              while((l<r)&&(array[r].a<=temp.a))
                                        r=r-1;
                              if(l<r)
                              {
                                        array[l]=array[r];
                                        l=l+1;
                              }
                              while((l<r)&&(array[l].a>=temp.a))
                                        l=l+1;
```

```
                                if(l<r)
                                {
                                        array[r]=array[l];
                                r=r-1;
                                }
                        }while(l!=r);
                        array[l]=temp;
                        qsort(array,left,l-1);
                        qsort(array,l+1,right);
                }
}
int main()
{
                int i;
                while(cin>>M>>N,M,N)
                {
                        result=0;
                        for(i=0;i<M;i++)
                        {
                        cin>>h[i].T>>h[i].V;
                        h[i].a=(double)h[i].V/h[i].T;
                        }
                        qsort(h,0,M-1);
                        for(i=0;i<M;i++)
                        {
                                if(h[i].T<=N)
                                {
                                        result=result+h[i].V;
                                        N=N-h[i].T;
                                }
                                else if(h[i].T>N&&N>0)
                                {
```

```
                                        result=result+h[i].a*N;
                                        break;
                                }
                        }
                        printf("%.2f\n",result);
                }
                return 0;
}
```

放苹果

Description

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。

Input

第一行是测试数据的数目 t（0 <= t <= 20）。以下每行均包含二个整数 M 和 N，以空格分开。1<=M, N<=10。

Output

对输入的每组数据 M 和 N，用一行输出相应的 K。

Sample Input

```
1
7 3
```

Sample Output

```
8
```

```
#include<stdio.h>

int F(int m,int n)
{
        if(m<0)
                return 0;
        if(m==0||n==1)
                return 1;
```

```
        return F(m-n,n)+F(m,n-1);
}

int main()
{
        int M,N,K,t,i;
        scanf("%d",&t);
        for(i=1;i<=t;i++)
        {
                scanf("%d %d",&M,&N);
            K=F(M,N);
       printf("%d\n",K);
        }
        return 0;
}
```

钱币兑换问题

描述

在一个国家仅有 1 分，2 分，3 分硬币，将钱 N 兑换成硬币有很多种兑法。请你编程序计算出共有多少种兑法。

输入

输入数据有多组，每组数据一行，每行只有一个正整数 N，N 小于 32768。

输出

对应每个输入，输出兑换方法数。

样例输入

```
2934
12553
```

样例输出

```
718831
13137761
```

```
#include<iostream>
```

```
using namespace std;
int sum[32768][4];
int fun(int m,int n)
{
    if(sum[m][n]>0)
                    return sum[m][n];
    if(m<0||n<1)
                    return 0;
    if(m==1||n==1||m==0)
                    return 1;
    return sum[m][n]=fun(m-n,n)+fun(m,n-1);
}
int main()
{
    int m;
    while(cin>>m)
                    cout<<fun(m,3)<<endl;
    return 0;
}
```

下沙的沙子有几粒？

Time Limit: 2000/1000 MS (Java/Others) Memory Limit: 65536/32768 K (Java/Others)
Total Submission(s): 521 Accepted Submission(s): 252

Problem Description

2005 年 11 月份，我们学校参加了 ACM/ICPC 亚洲赛区成都站的比赛，在这里，我们获得了历史性的突破，尽管只是一枚铜牌，但获奖那一刻的激动，也许将永远铭刻在我们几个人的心头。借此机会,特向去年为参加 ACM 亚洲赛而艰苦集训了近半年的各位老队员表示感谢。
实际上，除了获奖以外，在这次比赛期间还有一件事也让我们记忆深刻。那是比赛当天等待入场的时候，听到某个学校的一个队员在说："有个学校的英文名很有意思，叫什么 Hangzhou Dianzi University". 哈哈，看来我们学校的英文名起的非常好，非常吸引人呀。
不过，事情的发展谁也没有料到，随着杭电英文校名的这一次曝光，影响越来越大，很多人开始对杭电英文校名进行研究，不久以后甚至还成立了一个专门的研究机构，叫做"HDU 校名研究会"。并不断有报道说-相-当-多的知名科学家改行，专门对该问题进行研究，学术界称之为"杭电现象"。很多人在国际知名期刊上发表了研究论文，这其中，尤以中国超级女科学家宇春小姐写的一篇研究报告最为著名，报告发表在 science 上，标题是"杭电为什么这样红？" 文中研究发现：Hangzhou Dianzi University 这个校名具有深刻的哲学思想和内涵，她同时提出了一个大胆的猜想："假定一个字符串由 m 个 H 和 n 个 D 组成，从左到右扫描该串，如果字符 H 的累计数总是不小于字符 D 的累计数,那么，满足条件的字符串总数就恰好和下沙的沙粒一样多。"
这就是当今著名的"宇春猜想"！

虽然还没能从数学上证明这个猜想的正确性，但据说美国方面在小布什的亲自干预下，已经用超级计算机验证了在（1<=n<=m<=1000000000000）时都是正确的。my god! 这是一个多么伟大的猜想！虽然我们以前总说，21 世纪是属于中国的，可还是没想这一天来的这么早，自豪 ing... + 感动 ing...
感动和自豪之余，问题也来了，如果已知 m 和 n 的值，请计算下沙的沙粒到底有多少。

Ps:
1. 中国有关方面正在积极行动，着手为宇春小姐申报诺贝尔奖。
2、"宇春猜想"中提到的 H 和 D 组成的字符串现在被学术界成为"杭电串串"（"杭电串串"前不久被一个卖羊肉串的注册了商标，现在我校正在积极联系买断，据说卖方的底价是 1000 万欧元，绝不打折，看来希望不大，sigh...）

Input

输入数据包含多个测试实例，每个占一行，由两个整数 m 和 n 组成，m 和 n 分别表示字符串中 H 和 D 的个数。由于我们目前所使用的微机和老美的超级计算机没法比，所以题目给定的数据范围是（1<=n<=m<=20）。

Output

对于每个测试实例，请输出下沙的沙粒到底有多少，计算规则请参考"宇春猜想"，每个实例的输出占一行。

Sample Input

```
1 1

3 1
```

Sample Output

```
1

3
```

Source

HDU 2006-4 Programming Contest

Recommend

lxj

"假定一个字符串由 m 个 H 和 n 个 D 组成，从左到右扫描该串，如果字符 H 的累计数总是不小于字符 D 的累计数,那么，满足条件的字符串总数

```
#include<iostream>

using namespace std;

__int64 a[21][21];

__int64 fun(int m,int n)

{

          if(a[m][n])

                    return a[m][n];

          else if(m<n)

                    return 0;

          else if(m==0)

                    return 0;

          else if(n==0)

                    return 1;

          else

                    return a[m][n]=fun(m-1,n)+fun(m,n-1);

}


int main()

{

          int m,n;

          while(cin>>m>>n)

          {

                    printf("%I64d\n",fun(m,n));

          }

          return 0;

}
```

乘积最大描述

2000年是国际数学联盟确定的“2000——世界数学年”，又恰逢我国著名数学家华罗庚先生诞辰90周年。在华罗庚先生的家乡江苏金坛，组织了一场别开生面的数学智力竞赛的活动，你的一个好朋友 XZ 也有幸得以参加。活动中，主持人给所有参加活动的选手出了这样一道题目：

设有一个长度为 N 的数字串，要求选手使用 K 个乘号将它分成 K+1个部分，找出一种分法，使得这 K+1部分的乘积能够为最大。

同时，为了帮助选手能够正确理解题意，主持人还举了如下的一个例子：

有一个数字串：312， 当 N=3，K=1时会有以下两种分法：

1） 3*12=36

2） 31*2=62

这时，符合题目要求的结果是：31*2=62

现在，请你帮助你的好朋友 XZ 设计一个程序，求得正确的答案。

输入

程序的输入共有两行：

第一行共有2个自然数 N，K（6≤N≤10，1≤K≤6）

第二行是一个长度为 N 的数字串。

输出

输出所求得的最大乘积（一个自然数）, 答案在 long long 数据范围之内。

样例输入

```
4 2
1231
```

样例输出

```
62
```

解题思路：

递归求解。

```
#include<stdio.h>
#include<string.h>
int main(){
   char a[12];
   int b[12];
   int i,n,k;
   long f(int b[],int start,int end,int k);

   scanf("%d%d",&n,&k);
   scanf("%s",a);
   for(i=0;i<n;i++)
      b[i]=a[i]-'0';

   printf("%ld\n",f(b,0,n,k));

   return 0;
}
long f(int b[],int start,int end,int k){
   long r;
```

```
    int tmp,i;
    r=0;
    if(k==0){
        for(i=start;i<end;i++){
            r*=10;
            r+=b[i];
        }
    }
    else{
        for(i=0;i<end-k+1;i++){
            tmp=f(b,start,start+i+1,0)*f(b,start+i+1,end,k-1);
            if(r<tmp)
                r=tmp;
        }
    }
    return r;
}
```

1020: [NKPC2]小学生游戏

Judge type: Multi-cases (Detailed Mode - 10 cases)
Total Submit : 900 (153 users) Accepted Submit : 131 (87 users) Page View : 5355

Font Style: Aa Aa Aa

某天，无聊的小杰叫上几个同学玩游戏，其中有比较笨的小凤，比较傻的小雪，可爱的小鑫和自以为是的小练。他们去找聪明的小艺去给他们当裁判。判定谁取得游戏胜利。而这个游戏是：由小艺给出一个数 a，再给出一个数 b，经过规定的运算，使得数 a 变换成数 b，且使用最少的变换次数 n .谁先说对这个 n，谁就取得胜利。当然，因为都是小学生，所以假定如果 n>6，就算是没有答案。那么裁判小艺试图通过编程来使自己尽快的获得答案。请你帮帮他吧.....

问题描述：
题目给出数 a(a 是一个正整数，不超过 50 位)，再给出目标数 b(同样是一个正整数，不超过 50 位)，数的运算有三种：
1：使当前数加上 1985429
2：使当前数加上 2006
3：使当前数乘 2

需要你求出这个最小的 n，如果 n>6，输出-1。(此为负一)。

例 1：小艺给出数 a=1，给出数 b=1987437
那么最快我们经过 3 次指定运算可以使 1 变成 1987437
1*2=2；（第 3 种变换）
2+1985429=1985431；（第 1 种变换）
1985431+2006=1987437；（第 2 种变换）

例 2：小艺给出数 a=1，给出数 b=128
那么最快我们经过 7 次指定运算可以使 1 变成 128
1*2*2*2*2*2*2*2=128（均采用第 3 种变换），但是因为 n>6，所以按题意输出-1。

Input

输入仅包含两个整数 A、B，每行一个数字，0<A<1e+50，0<B<1e+50。

Output

输出只有一行，即为最少的变换次数 n，若 n>6 则输出-1。
请注意结尾含有一个换行。

Sample Input

1

1987437

Sample Output

3

Hint

Best User : Sempr

```
#include<iostream>
using namespace std;
#include<string.h>
#include<stdlib.h>
char A[52]={'\0'},B[52]={'\0'};
int best=7;
void add(char a1[],char b1[])
{
        int i,j,k,up,x,y,z,l;
        char *c;
        if(strlen(a1)>strlen(b1))
                l=strlen(a1)+2;
        else
                l=strlen(b1)+2;
        c=(char*)malloc(l*sizeof(char));
        i=strlen(a1)-1;
        j=strlen(b1)-1;
        k=0;
        up=0;
        while(i>=0||j>=0)
```

```
        {
                if(i<0)
                        x='0';
                else
                        x=a1[i];
                if(j<0)
                        y='0';
                else
                        y=b1[j];
                z=x-'0'+y-'0';
                if(up)
                        z=z+1;
                if(z>9)
                {
                        up=1;
                        z=z%10;
                }
                else
                        up=0;
                c[k++]=z+'0';
                i--;
                j--;
        }
        if(up)
                c[k++]='1';
        i=0;
        c[k]='\0';
        for(k=k-1;k>=0;k--)
                a1[i++]=c[k];
        a1[i]='\0';
}
```

```
void sub(char a[],char b[])
{
	char t[52];

	int i,l1,l2,k;
	l1=strlen(a);
	l2=strlen(b);
	t[l1]='\0';
	l1--;
	for(i=l2-1;i>=0;i--,l1--)
	{
		if(a[l1]-b[i]>=0)
			t[l1]=a[l1]-b[i]+'0';
		else
	    {
			t[l1]=10+a[l1]-b[i]+'0';
			a[l1-1]=a[l1-1]-1;
		}
	}
	k=l1;
	while(a[k]<0)
	{
		a[k]=a[k]+10;
		a[k-1]=a[k-1]-1;
		k--;
	}
	while(l1>=0)
	{
		t[l1]=a[l1];
		l1--;
	}
loop:
```

```
        if(t[0]=='0')
        {
                l1=strlen(a);
                for(i=0;i<l1-1;i++)
                        t[i]=t[i+1];
                t[l1-1]='\0';
                goto loop;
        }
        if(strlen(t)==0)
        {
                t[0]='0';
                t[1]='\0';
        }

        for(i=0,k=0;k<strlen(t);k++)
                a[i++]=t[k];
        a[i]='\0';
}

void div2(char a[])
{
        char b[52];
        int i,j,k;
        int d=0;
        int alen;
        alen=strlen(a);
    for(i=0;i<alen;i++)
        {
                b[i]=(((a[i]-'0')+d*10)/2)+'0';
        d=((a[i]-'0')+d*10)%2;
        }
        b[i]='\0';
```

```
        if(b[0]=='0')
        {
        for(j=0;j<i;j++)
                {
                        b[j]=b[j+1];
                }
        }
        for(i=0,k=0;k<strlen(b);k++)
                a[i++]=b[k];
        a[i]='\0';
}

int compare(char a[],char b[])
{
        int alen,blen;
        alen=strlen(a);
        blen=strlen(b);
        if(alen>blen)
                return 1;
        else if(alen<blen)
                return -1;
        else
        {
                for(int i=0;i<alen;i++)
                {
                        if(a[i]>b[i])
                                return 1;
                        else if(a[i]<b[i])
                                return -1;
                }
        }
        return 0;
```

```
}



void search(int k,char a[])
{
          if(k==7)
                    return;
          else
          {
                    if(compare(a,B)>0)
                              return;
                    if(compare(a,B)==0)
                    {
                              if(k<best)
                              {
                                        best=k;
                              }
                    }
                    else
                    {
                              add(a,a);
                              //puts(a);
            search(k+1,a);
                              div2(a);
                              //puts(a);
            add(a,"1985429");
                              //puts(a);
            search(k+1,a);
                              sub(a,"1985429");
                    add(a,"2006");
                              //puts(a);
                    search(k+1,a);
```

```
                    sub(a,"2006");
                }
            }
}
int main()
{
            gets(A);
            gets(B);
            search(0,A);
            if(best==7)
                    cout<<-1<<endl;
            else
                    cout<<best<<endl;
            return 0;
}
```

1020: [NKPC2]小学生游戏

Time Limit: 2000 ms Memory Limit: 10000 kB
Judge type: Multi-cases (Detailed Mode - 10 cases)
Total Submit : 900 (153 users) Accepted Submit : 131 (87 users) Page View : 5355

Font Style: Aa Aa Aa

某天，无聊的小杰叫上几个同学玩游戏，其中有比较笨的小凤，比较傻的小雪，可爱的小鑫和自以为是的小练。他们去找聪明的小艺去给他们当裁判。判定谁取得游戏胜利。而这个游戏是：由小艺给出一个数 a，再给出一个数 b，经过规定的运算，使得数 a 变换成数 b，且使用最少的变换次数 n .谁先说对这个 n，谁就取得胜利。当然，因为都是小学生，所以假定如果 n>6，就算是没有答案。那么裁判小艺试图通过编程来使自己尽快的获得答案。请你帮帮他吧.....

问题描述：
题目给出数 a(a 是一个正整数，不超过 50 位)，再给出目标数 b(同样是一个正整数，不超过 50 位)，数的运算有三种：
1：使当前数加上 1985429
2：使当前数加上 2006
3：使当前数乘 2

需要你求出这个最小的 n，如果 n>6，输出-1。(此为负一)。

例 1：小艺给出数 a=1，给出数 b=1987437
那么最快我们经过 3 次指定运算可以使 1 变成 1987437
1*2=2；（第 3 种变换）
2+1985429=1985431；（第 1 种变换）
1985431+2006=1987437；（第 2 种变换）

例 2：小艺给出数 a=1，给出数 b=128

那么最快我们经过 7 次指定运算可以使 1 变成 128
1*2*2*2*2*2*2*2=128（均采用第 3 种变换），但是因为 n>6，所以按题意输出-1。

Input

输入仅包含两个整数 A、B，每行一个数字，0<A<1e+50，0<B<1e+50。

Output

输出只有一行，即为最少的变换次数 n，若 n>6 则输出-1。
请注意结尾含有一个换行。

Sample Input

```
1

1987437
```

Sample Output

```
3
```

Best User : Sempr

```
#include<iostream>

using namespace std;

#include<string.h>

#include<stdlib.h>

char A[52]={'\0'},B[52]={'\0'};

int best=7;

void add(char a1[],char b1[])

{

          int i,j,k,up,x,y,z,l;

          char *c;

          if(strlen(a1)>strlen(b1))

                    l=strlen(a1)+2;

          else

                    l=strlen(b1)+2;

          c=(char*)malloc(l*sizeof(char));

          i=strlen(a1)-1;

          j=strlen(b1)-1;
```

```
k=0;
up=0;
while(i>=0||j>=0)
{
        if(i<0)
                x='0';
        else
                x=a1[i];
        if(j<0)
                y='0';
        else
                y=b1[j];
        z=x-'0'+y-'0';
        if(up)
                z=z+1;
        if(z>9)
        {
                up=1;
                z=z%10;
        }
        else
                up=0;
        c[k++]=z+'0';
        i--;
        j--;
}
if(up)
        c[k++]='1';
i=0;
c[k]='\0';
for(k=k-1;k>=0;k--)
        a1[i++]=c[k];
```

```
        a1[i]='\0';
}



int compare(char a[],char b[])
{
        int alen,blen;
        alen=strlen(a);
        blen=strlen(b);
        if(alen>blen)
                return 1;
        else if(alen<blen)
                return -1;
        else
        {
                for(int i=0;i<alen;i++)
                {
                        if(a[i]>b[i])
                                return 1;
                        else if(a[i]<b[i])
                                return -1;
                }
        }
        return 0;
}



void search(int k,char a[])
{
        if(k==7)
                return;
```

```
else
{
        if(compare(a,B)>0)
                return;
        if(compare(a,B)==0)
        {
                if(k<best)
                {
                        best=k;
                }
        }
        else
        {
                char temp1[52];
                for(int count = 0;count < 52;++count)
                        temp1[count] = a[count];
                add(temp1,temp1);

  search(k+1,temp1);
                char temp2[52];
                for(count = 0;count < 52;++count)
                        temp2[count] = a[count];
  add(temp2,"1985429");

  search(k+1,temp2);
                char temp3[52];
                for(count = 0;count < 52;++count)
                        temp3[count] = a[count];
        add(temp3,"2006");

        search(k+1,temp3);
        }
```

```
            }
}
int main()
{
            gets(A);
            gets(B);
            search(0,A);
            if(best==7)
                        cout<<-1<<endl;
            else
                        cout<<best<<endl;
            return 0;

}
```