A.Ryan's Hanoi Again

题目描述

还记得上次 Ryan 去印度旅游时遇到的汉诺塔么？

在苦思冥想研究了一周后，Ryan 终于掌握了汉诺塔的奥妙！

不过，这时一个印度小盆友蹦出来问他：“你会把所有盘子移到第二根柱子上么？”

Ryan 瞬间就给跪了。就在这时！你出现了，拯救了 Ryan 也受到了 Ryan 的膜拜！

Input

多组测试数据。

每组测试数据包含一行。一个整数 N(1<=N<=20)，表示汉诺塔的第一根杆上共有 N 个盘子。

盘子自上至下编号 1~N。

Output

对于每组数据，输出将所有盘子移动到第二根柱子上的每一步过程。每两步中间用回车换行。

我们用 A 表示第一根柱子，B 表示第二根柱子，C 表示第三根柱子

1: A -> B 表示将第一根柱子上编号为 1 的盘子移动到第二根柱子上

Sample Input

Sample Output

```
1: A -> B
2: A -> C
1: B -> C
3: A -> B
1: C -> A
2: C -> B
1: A -> B
```

Hint

本题使用 cout 输出会超时，请使用 printf 函数。

解题思路

本题为经典的汉诺塔问题，松油老师上课刚讲过大家应该印象深刻，一起复习一下，递归注意的是一定要有基本情况（n=1 时),二是要有无穷递归至基本情况的途径(如 an=a(n-1)）汉诺塔问题的解法松油老师讲过不再细说啦~说下本题特殊的输出每次移动的盘子的编号，其实很简单，在递归中添加参数盘子数，然后每次输出移动顺序的时候捎带把盘子数输出即可。（我想问我用 cout 为什么没超时就）

参考代码

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

void towers( int, char, char, char );
int nDisks;
char A,B,C;
int main()
{
   while(cin >> nDisks)
   towers( nDisks, 'A', 'B', 'C' );
}

void towers( int disks, char start, char end, char temp )  //递归
{
   if ( disks == 1 )
      cout << "1: " << start << " -> " << end << '\n';
   else
   {  towers( disks - 1, start, temp, end );

 cout << disks<<": " << start << " -> " << end << '\n';
                                                  //记得输出盘子数
      towers( disks - 1, temp, end, start );
   }
}
```

B.Thor 的行李箱

题目描述

Thor 是一个苦逼的 ACMer，要去 regional 被各路大神虐了。没办法，谁让 Thor 这么弱呢……还是收拾行李吧。唉，看到行李，Thor 的智商又捉鸡了。
Thor 有 4 个包，好吧忽略它们的高度，你可以把它们俯视看成一个个的正方形，Thor 的行李箱忽略高度也可以看成一个正方形。包之间不能互相压着，问 Thor 的行李箱至少要多大才能装的下所有行李。你只需要输出行李箱看成正方形的边长。

Input

多组测试数据。
每组数据只有一行，四个整数 a,b,c,d(1<=a,b,c,d<=1000)。为这些忽略高度后的包作为正方形的边长。

Output

对于每组测试数据，输出一个数，即你的答案。

Sample

输入：

```
2 2 2 2
2 2 1 2
```

输出：

```
4
4
```

Hint

第一个样例，就是个田字形
第二个样例，还可以看成田字形，只是右上角只有一个 1*1 的小块。
（就是个正方形里面套正方形，Thor 真不会描述）

解题思路

仔细分析下其实就是求最大边长的问题，只要最大的两个斜着放下了其他自然放下了。所以输入数据以后求出任意两边之和的最大值，即为行李箱正方形的边长。

参考代码

```
#include <iostream>
using namespace std;
int main()
{ int a, b, c, d, x1, x2, x3, x4, x5, x6;
    while ( cin >> a )
    {   cin >> b >> c >> d;
        x1 = a + b;
        x2 = b + c;
        x3 = c + d;
        x4 = a + d;
        x5 = a + c;
        x6 = b + d;
        if ( x1 <= x2 )  x1 = x2;
        if ( x1 <= x3 )  x1 = x3;
        if ( x1 <= x4 )  x1 = x4;
        if ( x1 <= x5 )  x1 = x5;
        if ( x1 <= x6 )  x1 = x6;
        cout << x1 << endl;
    }
}
```

C.新•神魔之井

题目描述

话说不知何年何月何日，天臧散人 Arthur 受神魔两界邀请，前往神魔之井对其进行了一次大规模的改造。
改造完成后，他和他的小伙伴惊讶地发现，自己被自己设计的迷宫困住出不去了……
幸运的是，Arthur 对迷宫的基本运作原理还是很清楚的（只不过这原理超出了他的探索范围而已）。
已知新的神魔之井迷宫由若干个房间构成，这些房间的变换遵从一个规律。如果当前时刻的房间数为偶数，那么下一个时辰，房间的个数就会减半。而如果是奇数，那么下一个时辰，房间的个数会乘以三再加上一。
于是机智的 Arthur 做出一个明智的决定——坐在原地等待。
已知当前房间的个数为 n，请你计算出他想要脱出迷宫需要等待的时间（单位：时辰）。

输入

多组测试数据。
每组数据为一行，包含一个整数 n（1≤n≤10000000），为当前时刻房间的个数。

输出

对于每组数据，输出一行，包含一个整数，为 Arthur 需要等待的时间。

输入样例

2
4
3

输出样例

1
2
7

HINT

这一次，请用递归实现
只有一个房间的迷宫，还出不去么？

解题思路

（本题不用递归直接用迭代也很好实现）

重点是递归的思想的熟悉吧给出递归和迭代两种解法大家自己比较下。（抓狂又不知道解题思路写什么了。哦艹我是那种短小的人吗!! 只怪题太短）

参考代码

迭代法：

```
#include <iostream>
using namespace std;
int main()
{int n, t, i = 0;
    while ( cin >> n )
    {while ( n != 1 )       //就不用递归 233
```

```
        {if ( n % 2 == 0 )    n = n / 2;
         else             n = n * 3 + 1;
         i++;}
        cout << i << endl;
     i = 0;
    }
}
```

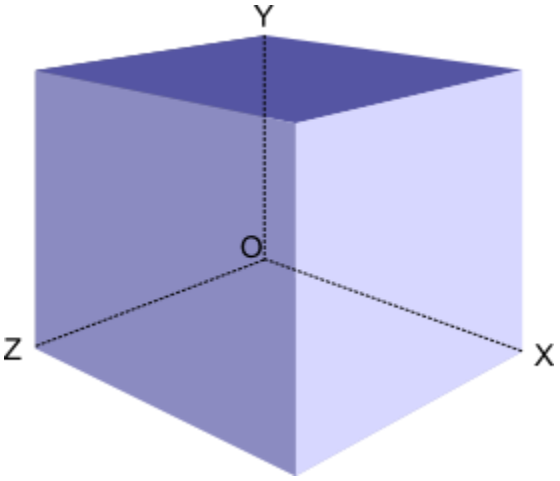
递归法：

```
#include <iostream>
using namespace std;
int dg( int );
int main()
{int n ;
    while ( cin >> n )
     cout << dg( n ) << endl ;
}
int dg( int num )        //递归
{   if ( num == 1 )  return  0 ;
    else if ( num % 2 == 0 )  return  dg ( num/2 ) + 1 ;
    else  return  dg ( num*3 + 1 ) + 1 ;
}
```

D. Dice I

题目描述

Ninja Gielch 很喜欢玩骰(tou)子(zi)。
一天，他跟朋友觉得每次总是看骰子正面朝上的点数很无聊，于是他们想出了另一种玩法：
在目力所及的范围内，看到的面上的点数的最大值作为当前的点数。
由于人为地观察骰子会存在偏差，他们决定用摄像头来观察骰子，再根据摄像头能看到的骰子的面上的点数的最大值作为他们当前的点数。
Ninja Gielch 已经把各种硬件设备配置好了，就差核心算法了。你能帮助他吗？
在他们的世界里，骰子是一个不透明的规则长方体。
为了计算方便，以骰子的一个端点为原点，过原点的三条棱为 x,y,z 轴的正方向，建立笛卡尔空间直角坐标系。
给定骰子原点所对的顶点的坐标和摄像头的坐标，问摄像头能看到的面上的点数最大值是多少。



Input

第一行有一个无符号整数 case(1<=case<=1000)，表示有 case 组数据。
接下来每组数据：

第一行包含 3 个绝对值不大于 1e6 的整数，x,y,z，表示摄像头的坐标。
第二行包含 3 个大于 0 不大于 1e6 的整数，x1,y1,z1，表示骰子另一个顶点的坐标。
第三行包含 6 个大于 0 不大于 1e6 的整数，a1,a2,...,a6，表示骰子 6 个面上的点数。

规定：a1 在 ZOX 面上，a2 在 ZOX 的平行面上，a3 在 XOY 面上，a4 在 XOY 的平行面上，a5 在 YOZ 面上，a6 在 YOZ 的平行面上。

Output

对于每组数据，输出点数的最大值。

Sample Input

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

0 0 10
3 2 3
1 2 3 4 5 6
```

Sample Output

```
6
4
```

Source

BUAA_LiJiancheng

解题思路

大家都知道摄像头最多只能同时看到三个面，所以先判断摄像头的位置，看看摄像头看到的是那三个面，根据摄像头坐标，骰子两个顶点坐标比较好判断。可以拿个骰子自己看着试验下，先确定一个方向（如 X 方向）的骰子顶点和摄像机的大小关系，然后就可以确定一个会出现的面的值，同理类推，得出会看到的面，然后再判断大小即可。

参考代码

```
#include<iostream>
#include<cstdio>
using namespace std;
int m( int p1, int p2, int p3, int p4, int p5, int p6 );
int main()
{   unsigned Case;
    int x1, y1, z1;
    int x, y, z;
    int a1, a2, a3, a4, a5, a6;
    int i = 1;
    cin >> Case;
    while ( i <= Case )
    {   int b1 = 0, b2 = 0, b3 = 0, b4 = 0, b5 = 0, b6 = 0;
        cin >> x >> y >> z;   //摄像头
        cin >> x1 >> y1 >> z1;    //骰子对角线顶点坐标
        cin >> a1 >> a2 >> a3 >> a4 >> a5 >> a6;   //6 个面
        if ( x > x1 ) b1 = a6;
        if ( y > y1 ) b2 = a2;
        if ( z > z1 ) b3 = a4;
        if ( x < 0 )  b4 = a5;
        if ( y < 0 )  b5 = a1;
        if ( z < 0 )  b6 = a3;   //选出摄像头所对应的面
```

```
            if ( b1 < b2 )b1 = b2;    //选出所对面最大数
            if ( b1 < b3 )b1 = b3;     //如果 b1 到 b6 有 0 直接无视掉
            if ( b1 < b4 )b1 = b4;
            if ( b1 < b5 )b1 = b5;
            if ( b1 < b6 )b1 = b6;
            cout << b1 << endl;
            i++;
        }
}
```

E. 晴天小猪的羊肉串

题目描述

冬天到了，晴天小猪和好朋友们围在一起吃烧烤。他们烤了很多很多的东西，有羊肉串、牛肉串、鸡肉串•••当然没有猪肉串。晴天小猪最喜欢鸡肉串了，他想在一堆的串中把它找出来。
我们用一个不超过 200 的字符串表示各种烤串，字符串中只会出现小写字母。如果字符串中出现了连续的三个字母，比如 abc、hij 等，就表示这是一个鸡肉串。

Input

多组测试数据。每组测试数据只有一行，先是一个整数 n（3<=n<=200），表示该串字符串的长度。接下来是一个长度为 n 的字符串。

Output

对于每组输入数据，若含鸡肉串，则输出“YES”，否则输出“NO”。

Sample Input

```
3 xyz
6 qwerty
7 qweabcr
```

Sample Output

```
YES
NO
YES
```

解题思路

字符串的判断，和上次上机题（第四次上机）的 C 题类似，不详细阐述了。

参考代码

```
#include <iostream>
using namespace std;
int main()
{int n, i, c;
    char a, b;
    while ( cin >> n )
    {   a = 0;
        c = 0;
        for( i = 1 ; i <= n ; i++ )
        {   b = a;
            cin >> a;
            if ( b == a - 1 ) c++;
            else if ( c >= 2 )  c = 2;
            else   c = 0;
        }
        if ( c >= 2 ) cout << "YES" << endl;
        else cout << "NO" << endl;
    }
}
```

F.Time

题目描述

Thor 是一个十分珍惜时间的人。他想知道，当前时间下，这一天已经过了多少秒。

Input

多组数据。
每组数据有 1 行，即为 hh:mm:ss
保证数据合理

Output

对于每组数据输出一行一个整数 Ans 表示当前时间下，这一天过去了多少秒。

Sample Input

```
00:00:01
10:00:00
```

Sample Output

```
1
36000
```

解题思路

送分题，主要就是记得用 scanf 保持格式，然后算的时候不要乘错，没别的了

参考代码

```
#include <iostream>

#include <cstdio>

using namespace std;

int main()
{
    int h, m, s, x;

    while ( scanf( "%d:%d:%d", &h, &m, &s ) != EOF )

    {   x = s + m * 60 + h * 3600;

        cout << x << endl;

        x = 0;
    }
}
```

G. Choice

题目描述

在北航马上就要校赛的背景下，地大要校赛了，软院目前在 13 级有两个名额。去地大玩，既锻炼又能看妹子，这么好的事儿，派谁去呢？
DPY 把这事儿交给了 Thor，他给 Thor 一段长度为 n 的数列和一个数 S，表示他对 n 个同学的看好程度 A[i]，要求 Thor 从里面选择两个人，使得这两个人的被看好程度之和大于 S。
现在，问 Thor 有多少的选择方式。

Input

多组数据。
每组数据有 2 行。
第一行有 2 个整数 n,s(1<=n<=1000,0<=s<=200000)。
第二行有 n 个整数 A[i](0<=A[i]<=200000)表示每个人被看好程度。

Output

对于每组数据输出一行一个整数 Ans 表示 Thor 的选择方案数。

Sample Input

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

Sample Output

```
5
3
```

Hint

用数组吧。

解题思路

分析题目以后发现就是要求一个数列中数项之和大于所求值得组合数。所以先用数组记录数据，再用循环和计数器进行判断测试，算出满足条件的组合个数即可

参考代码

```
#include <iostream>
#include <cstring>
using namespace std;
int a[ 1000 ];
int main()
{   memset( a, 0, sizeof( a ) );  //数组快速清零(可以学习一下)
    int s, n, ans;
    while ( cin >> n >> s )
    {   int i = 0;
        while ( i < n )
        { cin >> a[ i ];
          i++; }
        i = 0;
        ans = 0;
        int c = 1;
        while ( i < n - 1 )   //核心判断方案数算法
        {while ( c < n )
            {if ( a[ c ] + a[ i ] > s )
                ans++;
                c++;}
            i++;
            c = i + 1;
        }
        cout << ans << endl;
    }
}
```

H. Power Up!!!

题目描述

由于上一次上机的逗比情况，激发了渣诚的慢性心理阴影，所以这一次，他决定出一道很简单的题目：求 a 的 π 次方。
然而这时 Thor 发话了——怎么能这么水呢？于是机智的 Thor 决定附加有个规则：
不准使用 pow 函数！
So……good luck fellows……

输入

多组测试数据。
每组测试数据为一行，只有一个正实数，表示 a。
保证输入数据合法

输出

对于每组输入数据，输出一行，包含一个实数，为计算结果，结果保留两位小数。

输入样例

1
1

输出样例

1.00
1.00

HINT

我、我的草稿纸呢？
(╯‵□′)╯ ┻━┻
Thor 表示他什么都不知道……

解题思路

直接用 a 的 π 次方能 A 的话是不是有点太简单了→_→，显然不会那样，可能会有数据溢出什么的问题，所以换个思路。现在学高数刚好学到的！对于比较大的或者比较复杂的导数呀什么的取对数！这样就不会溢出了。外带 π 的表示不要用 3.14，明显数据大了误差大会跪，可以用反三角函数表示 π，再用取对数计算，最后的代码很简单，就是题目有点坑。

参考代码

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{   int a;
    double pai, x;
    while ( cin >> a )
    {   pai = asin( 1 ) * 2;  //求出 π
        x = exp( pai * log( a ) );   //取对数求法！
        cout << fixed << setprecision( 2 ) << x << endl;
    }
}
```

I. Aftermath!!

题目描述

Aftermath——有劫后余生之意，在如今的网络文化中特指数学考试过后的惨状。
于是本题便是由此而来，劫后余生，出一道水题吧。
给你一个包含 n 个元素的整数序列，为 a1、a2、…、an。
然后给你一个数 i,和一个数 j，求解上述序列中第 i 个元素与第 j 个元素间所有元素的和（第 i 个元素与第 j 个元素也包括在内）

输入

多组测试数据。
对于每组数据，第一行为两个整数 n,m(1≤m,n≤500000)，用空格隔开，分别表示序列元素的个数 n 与查询的个数 m；
第二行为 n 个整数，用空格隔开，整数取值均在[0,100]之间。
接下来 m 行，每行为两个整数 i,j（1≤i≤j≤n），具体含义见题目描述

输出

对于每组数据，输出 m 行，每行一个整数，对应一次查询的结果。
每组数据之间输出一个换行

输入样例

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

输出样例

```
3
6
5

1
```

HINT

本题使用 cin 和 cout 会超时，请使用 scanf 与 printf

解题思路

题目看着比较高大上的样子，其实就是求数列中从第 a 项到第 b 项的和而已，换个思路会好很多。用数组表示前 n 项和，求第几项到第几项的和减一下即可

参考代码

```
#include<iostream>
#include <cstdio>
int number[ 500000 ] ;
using namespace std;
int main()
{ int n, m, i, j, num;
 while( scanf( "%d%d", &n, &m )!= EOF )
 { number [ 0 ] = 0 ;
  for( int k = 1 ; k <= n ; k++ )
  { scanf ( "%d", &num );
   number[k] = number[ k-1 ] + num ;}//每一个数组里的数储存前面数之和
  for( int t = 0 ; t < m ; t++ )
  { scanf ( "%d%d", &i, &j );
    printf ( "%d\n", number[ j ] - number [ i - 1 ] );}  //输出差值即为数之和
    printf( "\n" );
 }
}
```

J. Sequence Detector

题目描述

Darkness Wong 最近在准备他的创新杯项目。

现在他的项目需要一个对数列进行检测的功能。对于一个给定的数列，该程序需要输出这个数列的类型编号。已知的类型及编号如下：

1） 仅为等差数列
2） 仅为等比数列
3） 既是等差数列又是等比数列
4） 既不是等差数列也不是等比数列

请帮助 Darkness 实现这样的程序。

Input

多组测试数据。

对于每组测试数据，第一行只有一个整数 N(3<=N<=5000)，代表数列的元素个数。

第二行包含 N 个用空格隔开的正整数，表示整个数列。

Output

对于每组数据，输出一行，数列的类型编号。(详见样例)

Sample Input

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

Sample Output

```
1
4
```

解题思路

判断等差和等比有公式，如 a2-a1=a3-a2，a2^2=a1*a3，既是等差又是等比只能是常数列，所以对于输入的数据，先判断前三个，然后每输入一个数据，判断一次是否为等差或等比或既是等差又是等比，最后输出结果即可。判断条件都是高中数学知识，整体算法没有难度。

参考代码

```
#include <iostream>
using namespace std;
int main()
{   int n, a, x, y, m, ans;
    while ( cin >> n )
    {   m = 0;
        cin >> x >> y >> a;
        if ( x == a && y == a ) ans = 3;
        else if ( y - x == a - y ) ans = 1;
        else if ( y * y == a * x ) ans = 2;
        else  ans = 4;
        x = y;
        y = a;
        m = ans;
        for ( int i = 4 ; i <= n ; i++ )
        {   cin >> a;
            if ( x == a && y == a )  ans = 3;
            else if ( y - x == a - y ) ans = 1;
            else if ( y * y == a * x )  ans = 2;
            else   ans = 4;
            if ( ans != m || m == 4 )  m = 4;
            x = y;
            y = a;
        }
        if ( m != 4 )  cout << ans << endl;
        else   cout << m << endl;
    }
}
```