2014 级第五次上机解题报告

巩毅琛 14211072

A jhljx 学 gcd

Problem Description

大家都知道 gcd 是最大公约数的意思。jhljx 准备开始学习 gcd 了。他想要求出 n 个数的最大公约数 gcd 和最小公倍数 lcm。请你帮帮他。

Input

输入多组数据。

每组数据两行，第一行为一个正整数 n,表示有多少个数(2<=n<=20)。

第二行有 n 个正整数，每个数之间用空格隔开。

Output

输出这 n 个数共同的最大公约数和最小公倍数（保证结果在 int 范围内）。

Sample Input

2

9 15

3

24 60 18

Sample Output

3 45

6 360

解题思路：求两个数的最大公约数用辗转相除法，我们都应该很熟悉。那么 a1，a2，a3 的最大公约数就是 a1 和 a2 的最大公约数与 a3 的最大公约数，以此类推，a1~an 的最大公约数就是 a1~an-1 的最大公约数与 an 的最大公约数，于是可以循环使用辗转相除法来完成计算。求 n 个数的最小公倍数也是用这种思路，其中求两个数的最小公倍数有两个方法。（一），两个数的最小公倍数肯定是较大的那个数的某一个倍数，所以从较大的那个数开始测试其倍数是否是较小数的倍数即可。（二），在求出两个数的最大公约数之后，直接用两数乘积除最大公约数得到最小公倍数，利用 a1~an 的最小公倍数是 a1~an-1 的最小公倍数与 an 的最小公倍数这一规律，循环计算即可。

参考代码：（一）

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

int f(int a, int b)//求两数最小公倍数
{
```

```
    int i;
    if(a > b)
    {
        a = a + b;
        b = a - b;
        a = a - b;
    }
    for(i = b;; i+=b)
        if(i % a == 0&&i%b==0)
        {
            return i;
            break;
        }
}

int gcd(int a , int b)//求最大公约数
{
    int t;
    if (a < b)
    {
        t = a;
        a = b;
        b = t;
    }

    while(b!=0 )
    {
        t = a % b;
        a = b;
        b = t;
    }
    return a;
}

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

```
        int zy =gcd(a[0], a[1]);
        for(int i=2;i<n;i++)// a1~an 的最大公约数就是 a1~an-1 的最大公约数与 an 的最大公
约数
        {
            zy =gcd(zy, a[i]);
        }
        cout<<zy<<" ";

        int lcm=f(a[0],a[1]);
        for(int i=2;i<n;i++)//a1~an 的最小公倍数是 a1~an-1 的最小公倍数与 an 的最小公倍
数
        {
            lcm=f(lcm, a[i]);
        }
        cout<<lcm<<endl;
    }
}
```

（二）

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

int gcd(int a , int b)//求最大公约数
{
    int t;
    if (a < b)
    {
        t = a;
        a = b;
        b = t;
    }

    while(b!=0 )
    {
        t = a % b;
        a = b;
        b = t;
    }
    return a;
}


int main()
{
```

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

        int zy =gcd(a[0], a[1]);
        int lcm= a[0]*a[1]/zy;
        for(int i=2;i<n;i++)
        {
            zy =gcd(zy, a[i]);
            lcm =lcm*a[i]/gcd(lcm, a[i]);// 在求出两个数的最大公约数之后，直接用两数乘积除最大公约数得到最小公倍数
        }
        cout<<zy<<" "<<lcm<<endl;
    }
}
```

B jhljx 学素数

Problem Description

函数是一个重要的知识点。jhljx 一改丧心病狂的风格，来点小清新。

他让你用函数实现判断一个数是否为素数。

Input

输入多组数据。

输入一个非负整数 n。(保证 n 在 long long 范围内,但不会很大)

Output

如果这个数是素数，输出"jhljx is good!"，否则输出"jhljx is sangxinbingkuang!"。

Sample Input

1

2

Sample Output

jhljx is sangxinbingkuang!

jhljx is good!

解题思路：判断一个数是否是质数的方法应该都知道（用该数除 2,除 3，一直除到该数开方加 1（开方后可能有被消去的小数部分），若都不能被整除则是素数）。这里就是再练习一下定义函数，实际上我把整个函数都放到了子函数里，主函数只是调用子函数。

参考代码：#include <iostream>

#include <cmath>

```
using namespace std;

int f(long long a)
{
	long long k=sqrt(a)+1;
	long long i=2;
	if(a==1)
		cout<<"jhljx is sangxinbingkuang!"<<endl;
	else
	{
		while(i<=k)
	   {
		if(a%i==0)
			break;
		i++;
	   }
	if(i>=k)
		cout<<"jhljx is good!"<<endl;
	else cout<<"jhljx is sangxinbingkuang!"<<endl;
	}
}

int main()
{
	long long n;
	while(cin>>n)
	{
		f(n);
	}
}
```

C jhljx 学下棋

Problem Description

jhljx 最近喜欢上了下棋，他要和 Last_Day 下棋。

Last_Day 给了他一个 n*n 的棋盘。jhljx 决定在棋盘上放上小兵。如果小兵放在（x,y）位置，那么他会攻击处在（x-1,y）,(x+1,y),（x,y-1）,(x,y+1)在四个位置的棋子（如果这几个位置存在）。请问 jhljx 最多可以放多少个小兵，保证他们不会相互攻击。

Input

输入多组数据。

每组数据一行，为一个数 n。(保证 n 在 int 范围内)

Output

输出最多可放置的小兵的个数。

Sample Input

1

2

3

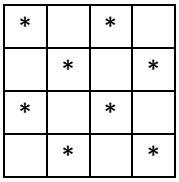
Sample Output

1

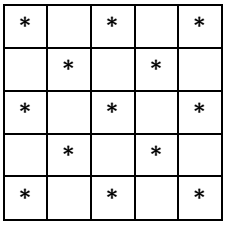
2

5

解题思路：画图即可找到规律。如果 n 为偶数，则小兵即为所有位置的一半，即 n*n/2



如果 n 为奇数，则小兵即为（所有位置+1）的一半，即（n*n+1）/2



参考代码：#include <iostream>

```
using namespace std;

int main()
{
	long long n;
	while(cin>>n)
	{
		if(n==1)
		{
			cout<<"1"<<endl;
		}
		else
		{
			 if(n%2==0)
			cout<< n*n/2<<endl;
			else
			cout<< (n*n+1)/2<<endl;
		}

	}
}
```

D jhljx 的强迫症

Problem Description

jhljx 最近有点不太正常，他觉得自己貌似患上了一种奇奇怪怪的病，这种病好像叫做强迫症。。噗。。一天，树荫姐给了 jhljx 两个数 n 和 m,树荫姐说我们来做 n 和 m 的模运算吧。。

jhljx 叫道：“好吖好吖”。jhljx 虽然数数数不清，但他不喜欢别人 chaofeng 他数数数不清。。

于是，jhljx 决定证明给你们看。jhljx 拿着 n 这个数左右把玩，他不断地对 n 累加，于是得到了 n,2n,3n,4n……拿着许许多多的数 jhljx 很开心。但是他想知道这些数模上 m 的结果(举个例子吖，就是 n%m,2n%m,3n%m,……)是不是能够得到 0~m-1 之间的所有数，只有得到了 0~m-1 之间的所有数 jhljx 才心满意足，如果没有得到，他连觉都睡不好。

Input

输入多组测试数据直到文件结束。

每组测试数据只有一行，为 n 和 m 的值。n 和 m(n>0,m>0 且保证 n 和 m 在 int 范围内)之间用空格隔开。

Output

如果 n 的所有倍数模上 m 的值能够取遍 0~m-1 之间的所有数，输出“jhljxshidadoubi”,反之，输出“shuishuowoshidadoubi”。

Sample Input

3 5

Sample Output

jhljxshidadoubi

Hint

童鞋快看这里。

3+0=3,3%5=**3**;

3+3=6,6%5=**1**;

3+3+3=9,9%5=**4**;

3+3+3+3=12,12%5=**2**;

3+3+3+3+3=15,15%5=**0**;

3+3+3+3+3+3=18,18%5=**3**;

m=5,这些余数取到了 0，1，2，3，4，满足！get!

解题思路：其实就是判断两个数的最大公约数是不是 1（判断是否互质），因为如果不是互质，则在用越来越大的 n 的倍数的时候，总有所谓的“空缺”，n 的倍数摸 m 得不到某个或某些数，于是不能取遍 0~m-1 之间的所有数，而互质就可以。但其中有特殊情况，若 m=1，则甭管 n 为多少，最大公约数都为 1，但此时只能 n 的倍数摸 1 只能取到 0，没有 1，所以输出 shuishuowoshidadoubi。若 n=1 且 m≠1 则一定能取到所有的数，想想就知道。然后在最后 n≠1 且 m≠1 的时候就用一开始说的那思路吧。

参考代码：#include <iostream>

using namespace std;

```
int f(int a,int b)
{
	int t;
	 if (a < b)
		{
			t = a;
			a = b;
			b = t;
		}

		while(b!=0 )
		{
		t = a % b;
		a = b;
		b = t;
		}
		return a;
}

int main()
{
	int n, m;
	while(cin>>n>>m)
	{
		if(m==1)
			cout<<"shuishuowoshidadoubi"<<endl;
		else if(n==1&&m!=1)
			cout<<"jhljxshidadoubi"<<endl;
		else if(n!=1&&m!=1)
		{
			if(f(n, m)==1)
				cout<<"jhljxshidadoubi"<<endl;
			else cout<<"shuishuowoshidadoubi"<<endl;
		}
	}
}
```

E 汉诺塔再度来袭

Problem Description

汉诺塔（又称河内塔）问题是源于印度一个古老传说的益智玩具。大梵天创造世界的时候做了三根金刚石柱子，在一根柱子上从下往上按照大小顺序摞着 64 片黄金圆盘。大梵天命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱子上。并且规定，在小圆盘上不能放大圆盘，在三根柱子之间一次只能移动一个圆盘。

假设三根柱子分别是 A,B,C。盘子编号为 1，2，3……n,最开始时，按照编号从小到大的顺序放在 A 柱子上。n 号盘子在最下方，1 号盘子在最上方。

Input

输入多组数据。
每组数据一个 n,表示黄金圆盘的个数。(1<=n<=20)

Ouput

输出需要移动的步数和移动的具体方案。详细请参见样例。
比如 1 A->C 表示将 1 号盘子从 A 柱子上移到 C 柱子上。

Sample Input

1
2

Sample Output

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

Hint

样例解释：
Samle 1：输出的结果 1 A->C 数字和字母之间有一个空格
本题请用 scanf 和 printf 进行输出，用 cin 和 cout 会超时。

解题思路：对于输出移动的具体方案，在练习赛中我们都练过了，解题报告中我也写的很详细，我直接复制粘贴上回我写的吧
（这是用递归来实现的
先注意下，在这定义的递归函数里，移动的方向都是第一个到第三个，不论第一个第二个第三个代表的字符分别是什么，总是这么空出第二个来从第一个移动到第三个，这是在汉诺塔移动所有盘子的共通的地方
要把 n 个从 A 通过 B 移到 C，就先要把上面 n-1 个从 A 移到 B【1】，然后把第 n 个移到 C。在把 n-1 个从 B 移动到 C【2】。（一）
那么首先要解决【1】这个问题，也是用同样的方法，和方才的大问题（一）一样的问题，所以相同的思路，只是把 B 和 C 交换位置而已（满足从第一个到第三个移动的共通点），也就是说把 n-2 个移到 C，方法用的和【1】一样，就这样递归实现。解决【2】时也同思路，把 A 和 B 互换（还是满足满足从第一个到第三个移动的共通点），因为要将位于 B 的 n-1 个移到已经有第 n 个的 C 上。然后就这么递归下去，也甭管 n 不断减 1 后 n 变得很小时盘子是怎么移动的，根据写的递归函数他自己就会算出来，没必要想那么多。
）
然后对于输出需要移动的步数，拿 n=3 的时候举例子，n=3 的前三步就是重复 n=2 时候，把小的两块移动到另一个一个柱子。然后就是将最大块移动到另一块柱子上，在继续重复 n=2 把小的两块移动到另一个（指存在 3 的一个）柱子上。扩展到 n，即 f（n）=2*f（n-1）+1。
参考代码：

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
        void hanoi(int ,char ,char ,char );
        int val (int);
```

```
        int m;

        while(cin>>m)
        {
                printf("%d\n",val(m));
                hanoi(m,'A','B','C');
        }

}


void hanoi(int n,char one,char two,char three)
{
        if (n == 1)
        {
             printf("%d %c->%c\n",n , one, three);
        }
        else
        {
               hanoi(n-1,one,three,two);
               printf("%d %c->%c\n", n, one ,three);
               hanoi(n-1,two,one,three);
        }
}

int val(int n)
{
      int c;

      if(n==1)
            c=1;
      else c=2*val(n-1)+1;
      return c;
}
```

F jhljx 学斐波那契数列(I)

Problem Description

jhljx 听说你们学了斐波那契数列，于是他想考考你们。他想问你斐波那契数列的第 n 项是多少。

有人不知道斐波那契数列？
它就是 1，1，2，3，5，8……这样的一个数列。

Input

输入多组数据。
每组数据输入一个 n。

Ouput

输出斐波那契数列中的第 n 个数。

Sample Input

1
2
3

Sample Output

1
1
2

解题思路：思路很简单，不必多说。f(n)=f(n-1)+f(n-2)。详情可见 c++课本 199 页。
参考代码：

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

int f(int n)
{
	if(n<=1)
		return n;
	else
		return f(n-1)+f(n-2);
}

int main()
{
	int n;
```

```
    while(cin>>n)
    {
        cout<<f(n)<<endl;
    }
}
```

G 找不到的孩子们

Problem Description

数列是一个神奇的存在。
现在定义一个数列:f(n)=|f(n-1)-f(n-2)|
给你 f(1)和 f(2)
//求 f(n)
年轻人好好刷题别做梦了，怎么会那么简单。
KamuiKirito 告诉你这个数列会出现的数字个数为有限个。
求该数列会出现的数字个数。

Input

输入多组数据。
每组数据为两个整数 a,b，代表 f(1)和 f(2)。(0<=a,b<=10^18)

Output

每组数据输出一行，为会出现的数字个数。

Sample Input

2 1
4 6

Sample Output

3
4

解题思路：想几个例子在纸上多演算即便找到规律。直接说规律不好说。我就拿 17 和 4 举个例子。（特殊情况 a=0 且 b=0 则只求一个数，a 和 b 有一个为 0 则有两个数，想想就知道）。对于 17 和 4，大小前后顺序不用管，若 17 在前，则是 17 4 13 9 4 5 1 4 3 1 2 1 1 0，若 4 在

前，则是 4 17 13 4 9 5 4 1 3 2 1 1 0。从中可以看出，首先是 17 不断减 4,到 13 到 9 到 5 ，减到最后保证比 4 大。减了 3 次，加上原来的 17，那么就有了 4 个数（a/b 的整数除法得到），之后再用 4（这是 17 减的那个 4，不是 a/b 得到的那个 4）来不断减 1（a%b 得到的），减到 3，到 2，到 1，到 0，减了 4 次（新 a（原 b）/新 b（原 a%原 b）整数除法得到）那么就又有了四个数，再加上原来的 4 就是 5 个，所以一共 9 个数。实际上出现 0 就意味着结束了（新 a 可以被新 b 整除），拿刚刚的到 1，到 0 时的情况举例子，减到最后就是 1 0 1 1 0 1 1 0 1 1 0 •••• 如此循环始终是 0 和 1 于是没有了继续讨论的意义。

参考代码：

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

int main()
{
    long long a, b;
    while(cin>>a>>b)
    {
        if(a==0&&b==0)
            cout<<"1"<<endl;
        else if(a==0||b==0)
            cout<<"2"<<endl;
        else
        {
            if(a<b)
            {
                long long t=b;
                b=a;
                a=t;
            }

            long long c=a/b;
            long long m=c;

            while(a%b!=0)
            {
                long long d=a%b;
                a=b;
                b=d;
                c=a/b;
                m+=c;
            }
            cout<<m+1<<endl;
        }
    }
}
```