2014 第五次上机解题报告

--------by 14211065 于济凡

目录

写在前面：

上机题目有时候是长时间的训练，有时候是一瞬间的灵感，所以，不用过于在意结果，毕竟，我们的目的都在于增长自己的能力，都在于成长。

第一题：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

如果只是单纯地比较和计算两个数之间的最大公约数和最小公倍数，那对于已经学习了函数和递归的同学们来说并不是什么难题，但是计算多个数的最大公约数和最小公倍数可能对于一些同学就有些难度了，那我们不妨现在先来复习下三个最大公约数和最小公倍数的求法。

比如：24,60,18

求最大公约数可以采取先求 24 与 60 的最大公约数，然后再求这个公约数 12 与 18 的最大公约数，得到 6

求最小公倍数可以采取先求 24 与 60 的最小公倍数 120，再求 120 与 18 的最小公倍数，得到 180.

由此可见：我们求多个数的最大公约数与最小公倍数时，可以采用先求前两个数的最大公约数，再不断用这个数的公约数与下一个数求公约数，直到结束，最小公倍数同理。

于是，代码如下：

```
#include<iostream>

using namespace std;
```

```
int gcd(int m,int n)
{
	int r;
	r=m%n;
	if(r==0)
		return n;
	else
		return gcd(n,r);
}

int lcm(int p,int q)
{
	return p*q/gcd(p,q);
}

int main()
{
	int counter;
	int x,y,z,x1,y1,z1;
	int gcdx,lcmx;
	while(cin>>counter)
	{
		cin>>x>>y;
		x1=x;
		y1=y;
		gcdx=gcd(x,y);
		lcmx=lcm(x1,y1);
		for;counter>=3;counter--)
		{
			cin>>y;
			y1=y;
			x=gcdx;
			gcdx=gcd(x,y);
			x1=lcmx;
			lcmx=lcm(x1,y1);
		}
		cout<<gcdx<<" "<<lcmx<<endl;
	}
}
```

第二题：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!

Hint

本题会检查代码，不用函数实现的一律 0 分。
本题是课本原题，见课本 214 页 6.29

首先，这道题的确是书后的原题，第二呢，没做也没关系，毕竟我们的第一次练习赛也

曾经做过类似的素数判断（2014 级第一次练习 E）。

不过这个题目还有一点，那就是要求用函数实现

那么具体如何实现呢？

判断一个数 n 是不是素数，只需要判断从 1 开始，一直到根号下 n，只要没有除了 1 之外的数能够被整除即可（为什么不是一直到 n 呢？因为怕超时间）

所以规避了超时的陷阱之后，这道题可能就基本做出来了。

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

int sushu(long long x)
{
        if(x==1)
        {
            printf("jhljx is sangxinbingkuang!\n");
        }
        else
        {
        int sum=0;
        for(long long i=1;i<=sqrt(x);i++)
        {
            if(x%i==0)
            {
                sum=sum+1;
            }
        }
        if(sum==1)
        {
            printf("jhljx is good!\n");
        }
        else
        {
            printf("jhljx is sangxinbingkuang!\n");
        }
        }

}
int main()
{
    long long x;
    while(cin>>x)
    {
        sushu(x);
```

```
    }
}
```

这道题给我们一个非常重要的提示，这个提示将在最后一题依然有用处，那就是要注意在算法中为计算机简化，否则。。。计算机这个笨蛋会超时的

第三题：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

这道题的难度并不很大，但是它却值得思考，比如，这道题还是运用了常用的判断：奇偶判断。

应用了奇偶判断，然后再进行一些基本的计算，我们不难发现数字规律，那就是奇数时候为（n*n+1）/2;,偶数时为 n*n/2

但是这样就结束了吗？万万没那么简单，这回如果只用 int，则会出现 WA。

这是为什么呢？因为在两个 int 数值相乘的过程中，结果会超过 int 范围，所以这道题要求我们使用 long long ，这样才能做对。

代码如下：

```
#include<iostream>

using namespace std;
int main()
{
    long long n;
    long long k;
    while(cin>>n)
    {
        if(n%2==1)
        {
            k=(n*n+1)/2;
            cout<<k<<endl;
        }
        else
        {
            k=n*n/2;
            cout<<k<<endl;
        }
    }
}
```

这道题再次给了我们启示：在进行了算法的思考后，一定要想想是否数值即将超过定义数据的范围，否则将出错。

第四题：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!

这道题再次是一道考察数学想法的题目，怎么才能在两个数的倍数和求模中得到 1~m-1 呢？
可能有人会想到数组，保存一个又一个数据，然后判断。。。
但是，这么做会惊讶地发现，这一定会超时间很久。
所以一定有什么简单算法的。
我们来观察一下，如果是 3 和 8，会出现所有的数吗？
3%8==3,
3*2%8=6
3*3%8=1
3*4%8=4
。。。
貌似可以，这是为什么呢？
那我们再来看看，如果是 6 和 8 呢?
那么貌似是：
6%8=6
6*2%8=2
6*3%8=2
。。。
为什么一个奇数都没有呢？
哦！原来是因为 6 和 8 有最大公约数 2，所以每次得出的结果都至少是加 2！
所以，只要判断两个数是否互质就好啦！（互质不就是最大公约数为 1 嘛！）
于是，代码如下：

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

```
int gcd(int x,int y)
{
	if(y==0)
	{
		return x;
	}
	else
	{
		return gcd(y,x%y);
	}
}
int main()
{
	int n,m,k;
	while(cin>>n>>m)
	{
		k=gcd(n,m);
		if(k==1)
		{
			cout<<"jhljxshidadoubi"<<endl;
		}
		else
		{
			cout<<"shuishuowoshidadoubi"<<endl;
		}
	}
}
```

那么，这道题有什么启示呢?不要着急，这只是一个小小的提示，最后一道题还没来呢。

第五题：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。

汉诺塔，这个演示递归最最经典的问题不出所料，再次出现在了这次上机题中。
如果之前刷过练习赛或者是去年的第四次上机的同学，对于汉诺塔应该比较熟悉了，在这里再次讲解下解决汉诺塔问题的基本原理：
首先有三个柱子：A，B，C
将 n 个盘子从初始柱子移到最终柱子，一共分三步：

1. 将前 n-1 个盘子从初始柱子 A 移到中间柱子 B
2. 将第 n 个盘子从初始柱子 A 移到最终柱子 C
3. 将前 n-1 个柱子从中间柱子移到最终柱子 C

注意：全过程递归，为了表示前 n-1 个盘子的移动，要先求 n-2。。。一直到只剩下一个盘子，就直接从 A 移到 C 即可。

这是最为经典的递归问题！一定要搞懂呀！

然后计算次数：

既然每一次都包括 n-1 个盘子的移动的两倍加一。。。那么。。。数学规律就来了。。。

移动 n 个盘子需要 2 的 n 次方减一次!!

于是，代码如下：

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

int f(int n,char start,char over,char middle)
{
	if(n==1)
	{
		printf("%d %c->%c\n",n,start,over);
	}
	else
	{
		f(n-1,start,middle,over);
		printf("%d %c->%c\n",n,start,over);
		f(n-1,middle,over,start);
	}
}
int main()
{
	char A='A';
	char C='B';
	char B='C';
	int n;
	while(cin>>n)
	{
		cout<<pow(2,n)-1<<endl;
		f(n,A,B,C);
	}
}
```

第六题：斐波那契数列

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

Hint

这道题用递归和迭代都可以过。纯签到题

思路：参见 Hint……

如果这个题不会。。。那就看书吧，书里讲的十分简单明了哦！

第七题：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

本次上机最终 boss 题来临！

这个题开做之前，希望你能看一下本次上机的第 A,B,C,D 题

得到了什么启示吗？

细细说来：

A 题：本题上来就将我们带入了最大公约数问题，这是否是提示呢？

B 题：这个题告诉我们小心超时，要优化算法，否则是会超时的哦。

C 题：这个题告诉我们，定义变量时，要小心计算过程中的超过数据范围的问题

D 题：再次使用了最大公约数的问题。

好，看看这道题怎么做！

可能会有不少同学选择上来就直接开一个数组，然后一个一个去判断，终于发现，超时啦！为什么超时！

因为要判断那么多数的话。。。4,6 这样的数据还可以接受，那么 100000000,1 呢？

电脑会累死的。。

所以，我们来看看有没有规律可言。

首先，先输入两个数，求差的绝对值，然后继续减，继续减。。。什么时候停止出现新的数据呢？

答案是，出现了 0 的时候。

那么怎么样才能出现 0 呢？我们不难发现，只有当之前的两个数相等时才会出现 0。

等等，这像不像是“更相减损之术”呢？

这是啥？

“更相减损之术”是最大公约数的另一种求法，至于怎么想到的，是因为最大公约数这个问题这次上机出现了好多次。。。（我就是这么想到的）

好，那么我们编一个直接利用更相减损，然后求循环次数不就好了？

结果发现：再次超时。

这说明我们的算法依旧不够简单。

怎么才能更加简单呢？在之前的高中学习中，我们知道了“更相减损之术”和“辗转相除法”而最后，选择这两种算法中的一种时，我们却选择了辗转相除，这是为什么呢？因为一次除法就相当于好多次减法了，所以我们是否可以按照这个来简化算法，我相信是可以。

以 100 和 40 为例，正常应该是以 100-40，得到 60，再 60-40 得到 20，再 40-20，最后 20=20，计算结束。

次数是什么呢？是不停地除法，直到等于 100%40，再继续，期间经历了 100/40 次。。。

等等，如果是倍数呢，比如 100 与 5 呢？

那么就是 100/5-1 次计算

好，综上我们可以设计一个代码，大概实现这道题：

代码如下：

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

long long f(long long x,long long y)
{
    long long sum=0;
    while(x!=y)
    {
        if(x>y)
```

```
    {
        if(x%y!=0)
        {
            sum+=x/y;
            x=x%y;
        }
        else
        {
            sum+=x/y-1;
            x=y;
        }
    }
    else
    {
        if(y%x!=0)
        {
            sum+=y/x;
            y=y%x;
        }
        else
        {
            sum+=y/x-1;
            y=x;
        }
    }
}
return sum;
sum=0;
```

```
}

int main()
{
	long long n,m;
	while(cin>>n>>m)
	{
		if(m==0)
		{
			if(n==0)
			{
				cout<<"1"<<endl;
			}
			else
			{
				cout<<"2"<<endl;
			}
		}
		else if(n==0)
		{
			if(m==0)
			{
				cout<<"1"<<endl;
			}
			else
			{
				cout<<"2"<<endl;
			}
```

```
            }
            else
            {
                cout<<f(n,m)+2<<endl;
            }
        }
}
```

通过观察代码，你可以发现我这条代码还有几个补充：

1. 我把数据范围改为 long long 了，这就是出自题干的 10^18 而改的，这是来自于 C 题的血泪经验。。。
2. 我考虑了 a,b 中上来就具有 0 的情况，这样才能防止在做除法时出现 Runtime Error，谨慎起见。
3. 利用函数让主函数小一点了。

如果对这个题自己能做出来，那么恭喜，你的思路一定让你有所提高~
（当然，希望大家积极说出自己的想法，大家都是最棒的）