C++第五次上机解题报告

//“本次上机主要考察函数和递归的应用，同时涉及到了很多与最大公约数和最小公倍数相关的思路。”某助教如是说。

//总体上看本次题目难度适中，（最后一道神题不讨论Orz~），只要大家平时勤加练习，应该都能取得不差的成绩。在下是 coding 小白，故代码多少有些简陋，但都可以AC。

A. jhljx 学 gcd

要求出 n 个数的最大公约数 gcd 和最小公倍数 lcm。

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）。

分析：求两个数的最大公约数和最小公倍数相信大家都很熟悉了，递归的 gcd 函数并不难，这里的关键在于如何求出一组数的最大公约数和最小公倍数。比如四个数 a，b，c，d，可以先求出 a 与 b 的最大公约数，然后再求这个数

和 c 的最大公约数，以此类推。最小公倍数的求法与之大同小异。注意到 n 不超过 20，开个数组也很方便。

代码：

```
#include <iostream>

using namespace std;

int gcd(int a,int b)//定义一个 gcd 函数，使用递归方法。

{

    if(b==0)

        return a;

    if(a<b)

        return gcd(b,a);

    else

        return gcd(b,a%b);

}

int main()

{
```

```
int a[24];//把数组开得稍微大一点
int n;
while(cin>>n)
{
    int yueshu,beishu=1,c = 1;//yueshu 和 beishu 是我们最终要得出的结果，这里的 c 是为后面暂时存储数据而准备。
    for(int i=1;i<=n;i++)
    {
        cin>>a[i];
    }
    yueshu=a[1];//把第一个数的值给 yueshu
    beishu=a[1]; //把第一个数的值也给 beishu
    for(int i=2;i<=n;i++)//然后从第二个数开始运算
    {
```

```
            yueshu=gcd(yueshu,a[i]);这里就用到了分析中提到的思路，注意当第一次循环时这个式子就是求 a[1]和 a[2]的最大公约数
            c=beishu;//把我们需要的 beishu 暂存在 c 中
            beishu=c*a[i]/gcd(c,a[i]);//同上，当第一次循环时这个式子就是求 a[1]和 a[2]的最小公倍数
        }
        cout<<yueshu<<" "<<beishu<<endl;
    }
}
```

//某些同学可能误认为一组数的最大公约数就是每两个数的最大公约数再取最大的一个，可以在草稿纸上用几组数据验证一下，就能看出正确的解法。

//做题时注意题干中给出的数据范围，有时会很有用处。

B. jhljx 学素数

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!"。

分析：求一个数 a 是否是素数的方法很简单，从 1 开始循环到这个数 a 自身，只要能整除他的数只有 1 和 a 两个即可。为了节省时间，可以从 1 循环到根号 a，只要只有 1 整除 a 即可。

代码：

```
#include<iostream>

#include<cmath>

using namespace std;

int sushu(long long n)//用拼音起函数名果然方便（其实是因为不知道英文单词 Orz）

{

    int ctr = 0;//定义一个计数器

    if(n==1)//注意到 1 也不是素数
```

```
        cout<<"jhljx is sangxinbingkuang!"<<endl;
    if(n>=2)
    {
    for(int i = 1;i<=sqrt(n);i++)
    {
        if(n%i==0)
            ctr++;
    }
    if(ctr==1)
        cout<<"jhljx is good!"<<endl;
    if(ctr>1)
        cout<<"jhljx is sangxinbingkuang!"<<endl;
    }

}
```

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

C. jhljx 学下棋

一个 n*n 的棋盘。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。

分析：多画几个图答案就“跃然纸上”了，n 分奇偶讨论即可。注意数据可能会爆掉 int，所以 long long 还是保险点~/*因为爆 int 没看出来被坑了半小时我会乱说？*/

代码：

```
#include<iostream>

using namespace std;

int main()

{

    long long n;

    while(cin>>n)

    {

        long long ans = 1;

        if(n%2==0)

        {

            ans = n*(n/2);
```

```
        }


        if(n%2==1)
        {
            ans = n*(n-1)/2+(n+1)/2;
        }
        cout<<ans<<endl;
    }
}
```

D. jhljx 的强迫症

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”。

分析：jhljx 童鞋在上机结束前半分钟点出了这道题的本质：只要 n 与 m 不互质，那么 n 的倍数模上 m 的值就能取完从 0 到 m-1 的所有数。反之若互质，就不行。/*后悔当年没好好看数论书啊 Orz*/

代码：

```
#include<iostream>

#include<cstdio>

#include<iostream>

#include<cmath>

#include<iomanip>//无视这些没用的头文件吧 QAQ

using namespace std;

int gcd(int x,int y)//一个用递归来做的最大公约数函数

{

    if(y==0)

        return x;

    if(x<y)
```

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

}

E. 汉诺塔再度来袭

十分经典的递归应用，可以在网上找找关于汉诺塔的思路分析，对递归的理解应该会更加深入。

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。

分析：对于 n 个盘子的汉诺塔，最少需要 2 的 n 次方减 1 步，对每一步的输出则需要一个递归函数。参数较多，不要搞混了。

代码：

```
#include<iostream>

#include<cmath>

#include<cstdio>

#include<cstdlib>
```

```
//第一个塔为初始塔，中间的塔为借用塔，最后一个塔为目标塔


void action(int n,char chushigan,char mubiaogan) //将编号为 n 的盘子由 chushigan（初始杆）移动到 mubiaogan（目标杆）
{
    printf("%d %c->%c\n",n,chushigan,mubiaogan);
}
void hanoi(int n,char chushigan,char zhongjiangan,char mubiaogan)//将 n 个盘子由初始塔移动到目标塔(利用借用杆)
{
    if (n==1)
    action(1,chushigan,mubiaogan);//只有一个盘子则是直接将初塔上的盘子移动到目的地
   else
```

```
    {
        hanoi(n-1,chushigan,mubiaogan,zhongjiangan);//先将初始塔的前 n-1 个盘子借助目的塔移动到借用塔上
        action(n,chushigan,mubiaogan);                    //将剩下的一个盘子移动到目的塔上
        hanoi(n-1,zhongjiangan,chushigan,mubiaogan);//最后将借用塔上的 n-1 个盘子移动到目的塔上
    }
}
int main()
{

    int n;
    while(scanf("%d",&n)!=EOF)
    {
        int step = pow(2,n)-1;//偷懒 Orz~写个已有结论吧，实际上也可以通过计算递归次数来算出步数。
```

```
    char x='A',y='B',z='C';
     printf("%d\n",step);
     hanoi(n,x,y,z);//这里的 x,y,z 分别代表初始，中间和目标~
     }
}
```

F. jhljx 学斐波那契数列(I)

求斐波那契的第 n 项，循环或者递归都不难完成。

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。

代码如下：

递归版：

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

```
using namespace std;
int fibo(int n)
{
	if(n==1)
		return 1;
	if(n==2)
		return 1;
	if(n>=3)
		return fibo(n-1)+fibo(n-2);
}
int main()
{
	int n;
	while(cin>>n)
	{
```

```
        cout<<fibo(n)<<endl;


    }
}
```

循环版：

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
using namespace std;
int main()
{
    int n;
    while(cin>>n)
    {
        int a = 1,b = 1,c = 0;
        if(n==1)
```

```
            cout<<a<<endl;
        if(n==2)
            cout<<b<<endl;
        if(n>2)
        {
            for(int i = 3;i<=n;i++)
            {
                c = a+b;
                a = b;
                b = c;
            }
            cout<<c<<endl;
        }
    }
}
```

//斐波那契数列相当有趣，可以 google 或者 wiki 一下~