1堆石子有n个,两人轮流取.先取者第1次可以取任意多个，但不能全部取完.以后每次取的石子数不能超过上次取子数的2倍。取完者胜.先取者负输出"Second win".先取者胜输出"First win".

Input

输入有多组.每组第1行是2<=n<2^31. n=0退出.

Output

先取者负输出"Second win". 先取者胜输出"First win".   
参看Sample Output.

Sample Input

2

13

10000

0

Sample Output

Second win

Second win

First win

典型的斐波那契博弈

有一堆个数为n(n>=2)的石子，游戏双方轮流取石子，规则如下：

1)先手不能在第一次把所有的石子取完，至少取1颗；

2)之后每次可以取的石子数至少为1，至多为对手刚取的石子数的2倍。

约定取走最后一个石子的人为赢家，求必败态。

结论：**当n为Fibonacci数的时候，必败。**

f[i]：1,2,3,5,8,13,21,34,55,89……

用第二数学归纳法证明：

为了方便，我们将n记为f[i]。

1、当i=2时，先手只能取1颗，显然必败，结论成立。

2、假设当i<=k时，结论成立。

     则当i=k+1时，f[i] = f[k]+f[k-1]。

     则我们可以把这一堆石子看成两堆，简称k堆和k-1堆。

    （一定可以看成两堆，因为假如先手第一次取的石子数大于或等于f[k-1]，则后手可以直接取完f[k]，因为f[k] < 2\*f[k-1]）

     对于k-1堆，由假设可知，不论先手怎样取，后手总能取到最后一颗。下面我们分析一下后手最后取的石子数x的情况。

     如果先手第一次取的石子数y>=f[k-1]/3，则这小堆所剩的石子数小于2y，即后手可以直接取完，此时x=f[k-1]-y，则x<=2/3\*f[k-1]。

     我们来比较一下2/3\*f[k-1]与1/2\*f[k]的大小。即4\*f[k-1]与3\*f[k]的大小，对两值作差后不难得出，后者大。

     所以我们得到，x<1/2\*f[k]。

     即后手取完k-1堆后，先手不能一下取完k堆，所以游戏规则没有改变，则由假设可知，对于k堆，后手仍能取到最后一颗，所以后手必胜。

     即i=k+1时，结论依然成立。

那么，当n不是Fibonacci数的时候，情况又是怎样的呢？

这里需要借助“Zeckendorf定理”（齐肯多夫定理）：**任何正整数可以表示为若干个不连续的Fibonacci数之和。**

关于这个定理的证明，感兴趣的同学可以在网上搜索相关资料，这里不再详述。

分解的时候，要取尽量大的Fibonacci数。

比如分解85：85在55和89之间，于是可以写成85=55+30，然后继续分解30,30在21和34之间，所以可以写成30=21+9，

依此类推，最后分解成85=55+21+8+1。

则我们可以把n写成  n = f[a1]+f[a2]+……+f[ap]。（a1>a2>……>ap）

我们令先手先取完f[ap]，即最小的这一堆。由于各个f之间不连续，则a(p-1) > ap  + 1，则有f[a(p-1)] > 2\*f[ap]。即后手只能取f[a(p-1)]这一堆，且不能一次取完。

此时后手相当于面临这个子游戏（只有f[a(p-1)]这一堆石子，且后手先取）的必败态，即先手一定可以取到这一堆的最后一颗石子。

同理可知，对于以后的每一堆，先手都可以取到这一堆的最后一颗石子，从而获得游戏的胜利。

部分参考：[http://yjq24.blogbus.com/logs/46150651.html](http://yjq24.blogbus.com/logs/46150651.html" \t "_blank)

#include <stdio.h>

#include <stdlib.h>

#include<iostream>

typedef long long ll;

using namespace std;

ll f[5010];

int main()

{

int n;

f[1]=f[2]=1;

for(int i=3;i<=5000;i++)

f[i]=f[i-1]+f[i-2];

while(cin>>n && n){

int flag(0);

for(int i=1;i<=5000;i++)

if(n==f[i]){

puts("Second win");

flag=1;

break;

}

if(!flag)puts("First win");

}

return 0;

}