Subprime Fibs

The following is an excerpt from the biography: Genius At Play: The Curious Mind of John Horton Conway

Then he'd clear the spare chair of the day's debris — The New York Times, which he devours back-to-front with the morning's bagel and coffee, his page-a-day Sudoku calendar, and a landslide of loose paper on which he's been running columns of numbers, playing not just any pointless game but a pointless game he recently invented. Subprime Fibs, he calls it, after the subprime mortgage crisis and the Fibonacci numbers (the sequence of numbers that begins 0, 1, with all subsequent numbers being the sum of the previous two — 0, 1, 1, 2, 3, 5, 8, 13, and so on).
The one and only rule of Subprime Fibs goes like this: Take any two numbers and write them down. Then add them up. If the sum is a prime number — a number divisible only by 1 or itself — write it down as well. If the sum is not prime, divide it by the smallest prime divisor and write down that result. Then take the last two numbers written down, and repeat the process. And continue to repeat it.
Hoisting himself out of his armchair, Conway affably offers a blackboard demonstration, adding as a disclaimer,

> *I know you're not really interested but I'll show you anyway because you were foolish enough to ask.*
>
> *1 and 1 make 2, and that's prime. So I write it down.*
> *1 and 2 make 3, and that's prime.*
> *2 and 3 make 5, which is prime.*
> *3 and 5 make 8, which isn't prime, so I divide it by the smallest prime I can, which is 2, and I get 4*
> *5 and 4 make 9, which isn't prime, divide it by 3 and I'll get 3, which is prime*
>
> *That's what you do. And what you get is this sequence:*
>
> *1, 1, 2, 3, 5, 4, 3, 7, 5, 6, 11…*

What would Conway call this kind of number game?

> *A waste of time!*

However, what's fascinating about the rule governing the game, for the man who invented it anyway, is that it's totally stupid but yet it exists.
We can start a subprime Fibonacci sequence with any two positive numbers. You can see that such a sequence doesn't grow fast, because we divide the terms too often. We conjecture that no subprime Fibonacci sequence grows indefinitely, but they all start cycling.
Meanwhile, the sequences are a lot of fun and I suggest a couple of exercises for you:

- Prove that there are no cycles of length two or three.
- Prove that the maximum number in a non-trivial cycle is prime.
- Prove that the smallest number in a non-trivial cycle is more than one. You can prove that it is more than 6 for extra credit.

By the way, a *trivial cycle* is the boring thing that happens if we start a sequence with two identical numbers n bigger than one: n, n, n, n, ….

The programming problem begins on the following page

In this problem you will complete several methods in the `SubprimeFibs` class. You will implement the `isPrime` method, the `getNthTerm` method and the `getCycleIndex`.

The `isPrime(int n)` method returns true in its `int` parameter `n` is prime, otherwise it returns false.

The following table shows sample results of the `isPrime(n)` method.
You may assume `n > 0`.

The following code	returns
`SubprimeFibs sf = new SubprimeFibs(1, 1);`	
`sf.isPrime(2)`	`true`
`sf.isPrime(3)`	`true`
`sf.isPrime(11)`	`true`
`sf.isPrime(1783)`	`true`
`sf.isPrime(7919)`	`true`
`sf.isPrime(20639)`	`true`
`sf.isPrime(906767)`	`true`
`sf.isPrime(1)`	`false`
`sf.isPrime(4)`	`false`
`sf.isPrime(9)`	`false`
`sf.isPrime(7919 * 20639)`	`false`
`sf.isPrime(20639 * 48623)`	`false`
`sf.isPrime(906767 * 11)`	`false`

The tables on the following page show sample results of the `getNthTerm(n)` and the `getCycleIndex()` method.

The `getNthTerm(int n)` method returns the n^{th} of the Subprime Fibonacci sequence. The Subprime Fibonacci sequence is constructed with the first and second terms of the sequence

The following table shows sample results of the `getNthTerm(n)` method.
You may assume `1 <= n , 999`.

The following code	returns
`SubprimeFibs sf = new SubprimeFibs(1, 1);`	
`sf.getNthTerm(1)`	1
`sf.getNthTerm(2)`	1
`sf.getNthTerm(5)`	5
`sf.getNthTerm(6)`	4

Each of the Subprime Fibonacci sequence in this problem will start cycling. The `getCycleIndex()` method returns index where the Subprime Fibonacci sequence cycle starts.

For example, `sf = new SubprimeFibs(339, 113)` generates the following sequence:
`339, 113, 226, 113, 113, …`
With a Cycle index of 4.

Another example, `sf = new SubprimeFibs(17, 48)` generates the following sequence:
`17, **48**, **13**, 61, 37, 49, 43, 46, 89, 45, 89, 45, 67, 56, 41, 97, 83, 76, 53, 43, **48**, **13**, …`
With a cycle index of 2.

The following table shows sample results of the `getCycleIndex()` method.

The following code	returns
`sf = new SubprimeFibs(339, 113);` `sf.getCycleIndex();`	4
`SubprimeFibs sf = new SubprimeFibs(17, 48);` `sf.getCycleIndex();`	2
`sf = new SubprimeFibs(5, 5);` `sf.getCycleIndex();`	1
`sf = new SubprimeFibs(5, 11);` `sf.getCycleIndex();`	71