10^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

11 October 2008

Problem B - Yellow Balloon
Fibonacci triangles

Problem Description

Fibonacci numbers have been claimed to represent everything from the rate at which rabbits multiply (though this claim appears to have little support in real life, and it can be construed as being speciest), to the number of florets in the outermost row of sunflower heads. Before we discuss yet another property of the Fibonacci sequence, a formal definition is in order:

$$F_n = F_{n-2} + F_{n-1}, \quad \forall n \in \mathbb{N} \mid n \geq 2$$

where $F_0 \equiv 0$ and $F_1 \equiv 1$.

One of the lesser known properties of Fibonacci numbers is that some of them form the hypotenuse of a right-angled triangle with integer sides. For example, $F_7 \equiv 13$, which is the hypotenuse of a right-angled triangle with sides 13, 12, and 5, since $13^2 = 12^2 + 5^2$.

One of your friends is working on his thesis linking the incidence of certain architectural styles to the predisposition of the owner to liking certain sports, or driving an SUV of some kind. Your friend has asked you to write a program to check whether a given number is a Fibonacci hypotenuse, as described above.

After a bit of research, you have identified the following useful test:

> A positive integer z is a Fibonacci number if and only if one of $5z^2 + 4$ or $5z^2 - 4$ is a perfect square.

A perfect square is an integer of which the square root is also an integer, such as the number 25, for example. Using this test, it would be straightforward to solve your friend's problem.

There is one minor complication, though. The list of numbers that you received from your friend are in the range $[0, 2^{30}]$. Since your Fibonacci test involves squaring the input numbers, your intermediate

results will be in the range $[0, 2^{60}]$, which exceeds the range of 32-bit integer variables. It also exceeds the 52-bit precision of typical floating point numbers, so you will have to perform all your tests using 64-bit integers.

In the C/C++ language, you can gain access to 64-bit integers by including the directive
`#include <stdint.h>`
and then using the `uint64_t` type. In Java, the `long` type should suffice.

You may also have to implement your own square root routine. The following pseudocode should get you started:

Algorithm 1 Calculate the integer square root r of x, such that $r^2 = x$

```
n ← 2^24
r ← (n + x/n) /2
while |r − n| > 1 do
    n ← r
    r ← (n + x/n) /2
end while
while r^2 > x do
    r ← r − 1
end while
return r
```

Input

Your input will consist of an arbitrary number of records, one record per line, with each record consisting of a single positive integer smaller than 2^{30}.

The end of the input is indicated by a line containing the value -1.

Output

For each input number, your program must produce the appropriate output. There are three possible scenarios:

1. The input value `n` is not a Fibonacci number. Your program prints

   ```
   n is not a Fibonacci number
   ```

2. The input value `n` is a Fibonacci number, but it is not the hypotenuse of a right-angled triangle with integer sides. Your program prints

   ```
   n is not the hypotenuse of a Pythagorean triple
   ```

3. The input value `n` is a Fibonacci number, and it is the hypotenuse of a right-angled triangle with integer sides `a` and `b`, with `a` $\geq$ `b`. Your program prints

```
a^2 + b^2 = n^2
```

Note that there may be more than one pair of values `a` and `b` that result in $\texttt{n}^2$ when their squares are summed. Your program must output *only* the pair corresponding to the largest overall value of `a` for each input value `n`.

Sample Input

```
100267
9227465
2
-1
```

Sample Output

```
100267 is not a Fibonacci number
9219273^2 + 388736^2 = 9227465^2
2 is not the hypotenuse of a Pythagorean triple
```

Time Limit

50 seconds