ENSC 151 – Assignment 2

Fall 2022 – Due November 9 – max 1 partner

Using the template eclipse project provided for the assignment and information provided in Chapters 2 through 5 of the textbook (and focusing on Chapter 5) and in addition some information provided in lectures, in this assignment you will work on a recursive program in C++ to calculate Fibonacci numbers. (Our Fibonacci numbers start with $f_0 = 0$, $f_1 = 1$). The program starts by outputting a specific line of instruction text to standard output. Do not modify this line. Afterwards, there are two major sections to your program just like in Assignment 1. In the first section, the program should input numbers from standard input. The numbers should be Fibonacci indices strictly increasing in value, or -1 to indicate the end of such input. After each index has been input, the Fibonacci number for that index should be calculated using a recursive strategy, and both the index and the Fibonacci number should be output in that order, each on a separate line. Note that you can skip as many indices as you want when entering indices. So you could enter

```
2
5
9
14
-1
```

as input and this would be perfectly fine.

The second major section of the program continues calculating Fibonacci numbers with a recursive strategy, and should calculate as many additional Fibonacci numbers as possible within a 2-second period. If a number is being calculated when the 2-second period elapses, it is allowed to be finished. In any case, the index of the last Fibonacci number calculated should be output on standard output on a separate line. Make sure that this last line is ended by using either "\n" or *std::endl* – there should be no blank lines that are output.

1) Run the program inputting -1 immediately. How many numbers can you generate in 2 seconds? Please switch the "alias declaration" fibonacciType1 from long to BigNumber and run your program again. Now how many numbers can you generate in 2 seconds? Between long and BigNumber, which is more efficient for integers with a smallish magnitude?

2) Comparing with the results reported for Assignment 1, note that the recursive function provided by the textbook authors is very inefficient. Let's say that we want to calculate fibonacci(40). This requires calculating fibonacci(39) and fibonacci(38). Consider that fibonacci(38) is calculated first and then fibonacci(39) is calculated. Well, calculating fibonacci(39) requires

calculating fibonacci(38) all over again. How inefficient. Instead of recursing down to fibonacci(0) or fibonacci(1) each time, we could remember some of the higher fibonacci numbers that we have calculated and that should speed things up immensely. Use either static variables or global variables to remember some of the higher fibonacci numbers. Please try implementing this. You can use either one recursive function or a set of mutually recursive functions. Recall that, in C++, a compiler is free to evaluate the operands for addition in any order that it wants. If you care about the order of evaluation of addition operands, do something to take this into account. We would like your code to work very well when using any compiler no matter how it handles evaluation of addition operands.

3) If you were successful in making the recursive fibonacci function more efficient, in 2 seconds you were probably able to calculate past fibonacci(92), the point at which fibonacci numbers become larger than we can store in a long variable. If you have fibonacci return a long, does fibonacci(93) return a correct fibonacci value?

4) For larger objects like some BigNumbers, working with a reference to an object can be more efficient than creating a copy of an object. Where they would make sense, try to work one or more references into your fibonacci code.

5) So for small fibonacci numbers, using long is faster than using BigNumber. Whether or not you were able to make the fibonacci function more efficient, get rid of the definition of the fibonacci function currently in assign2.cpp, and put some templates in the file templates.hpp. Your templates should certainly include a template to resolve a call such as

```
fibonacci<fibonacciType1>(index)
```

but can also include other function templates as well as global variable templates. Try using type alias fibonacciType1 for Section 1 of the program, and fibonacciType2 for Section 2 of the program. Make sure that you do thorough testing for Section 2, as the code as distributed outputs very little for Section 2.

If you have any questions about this assignment, please ask them on Piazza.