Read the following resource about different algorithms to calculate Fibonacci numbers, their time & space complexities:

1. http://www.ics.uci.edu/~eppstein/161/960109.html
    - This article lists 5 algorithms, namely
        1. *Recursive*
        2. *Dynamic Programming*
        3. *Space optimized Dynamic Programming*
        4. *Recursive Powering (using the power of matrices)*
        5. *Time optimized (log n) Recursive Powering*
2. This article introduces another algorithm (let's call it algorithm 6), which is named *Intelligent Recursive Fibonacci Solution*.

Assignment

Write 6 programs, to calculate the first *n* fibonacci numbers using the 6 algorithms identified above, and print them in order. Your program(s) should get *n* as a user input. e.g., if *n = 5*, your program should output

F(1) = 1

F(2) = 1

F(3) = 2

F(4) = 3

F(5) = 5

- For each program, get the time taken by the program to print the first *100* Fibonacci numbers. (hint: you can use the *time* command in Linux to get the time used by the program)
    - For Algorithm 1 you will get a nice (easily comparable) value for this with any computer. For the rest, it may be better to use an old computer OR calculate up to a large number (e.g., 10,000). Note that Fibonacci numbers you'll get at larger values will be wrong.
- It's better to use *unsigned long* to calculate the Fibonacci numbers!