time_fact_recur.cc

- Interesting find: The time elapsed nearly doubles once the code is asked to compute 13 factorial
- I am curious if this is related to how long recursion takes in Big O notation

time_fact_iter.cc

- I realized that int does not mean int64_t. This was because when I completed my iterative code and set the return type to just int, I was getting back negative numbers as the factorial numbers increased

time_fib_recur.cc

- Interesting find: the time elapsed to calculate the various N starting points for the fibonacci sequence actually jump around. It varies from 1 second to up to 9 seconds.
- Not sure why though

time_fib_iter.cc

- I understand why we try to use recursion now because this process to use iteration seemed tedious and more complicated than recursively solving it

Compare the results for Fibonacci and factorial, and note any interesting observations or patterns.

- Fibonaci took longer to compile, this is likely because the math is a bit more complex and requires more time to complete, especially once the numbers get larger
- Both results showcase random jumps and falls (in seconds required to complete) between the next number, it is still unknown to me why that is
- Factorial is more exponential than fibonacci, jumping to a very large number by 19 factorial

Adjust one of your files to compute the first 25 factorials. Note any interesting observations or patterns.

- We start to get random negative numbers in the last few factorials (21, 22, 24). In my opinion, this is because the numbers are too big for even the data type ‘int64_t’ to even hold them