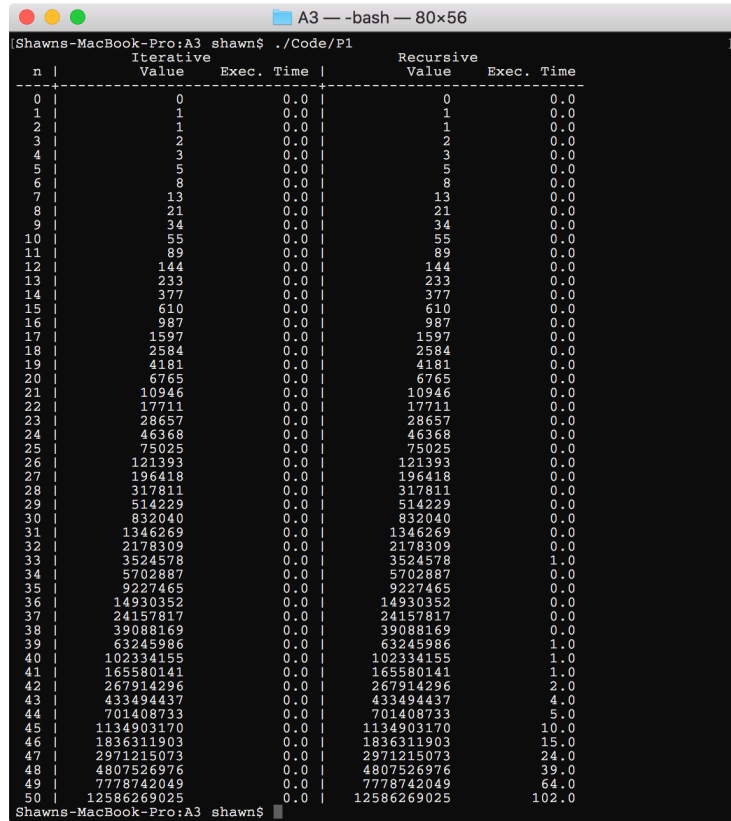# COMPSCI 2S03: Assignment 3 Report
## Shawn Esseltine | 400145888

## Problem 1 (Fibonacci)

Disparities between the recursive and the iterative approach to solving the Fibonacci series are made clear through execution time, size of code, and memory consumption. Figure 1 highlights the execution time required in seconds for each approach at a given value n. A value of zero represents an execution time of less than one second. In comparing the two Execution Time columns, the recursive approach shows a considerable increase in time when calculating large n values. When comparing size of code, both the recursive and iterative approach are close in length. In the tested code the recursive function is one line shorter than that of iterative approach, being eight lines and nine lines respectively. When comparing memory usage, the iterative approach has a constant memory usage of three variables with



*Figure 1*

type *long long int;* each have a memory utilization of 8 bytes and thus a total usage of 24 bytes for the function. However, when the Fibonacci function is called recursively, each base case stores a value of *long long int* with a corresponding memory utilization of 8 bytes. This means that the memory usage will increase linearly with the number of terms calculated in the sequence and can be represented using the following formula: $\# \text{ Bytes} = 8 \text{ Bytes} \cdot fibRecurse(n+1)$. The nth plus one value is used since the 0th base case is still assigned 8 bytes of storage. When considering the aforementioned factors, it is clear to see that unless searching for the Fibonacci sum where n is less than or equal to two, it is beneficial to use the iterative case.