Shifra Schectman
CS162-Lab 10
June 10, 2018
Analysis and Discussion of Results

There is a theorem that sates that anything that can be solved recursively, can also be solved iteratively. In this program, I will be running code that calculates the Fibonacci number at a certain spot. I will do so recursively and iteratively. I will record the run-time used by both functions and compare it. I remember reading about recursion that it can be inefficient for larger numbers as it uses a lot of memory. Therefore, I think that the iterative function will take less time to run.

Here is my chart which records the run time of both the recursive and iterative function used to find the Fibonacci number.

N	**Recursive Time**	**Iterative Time**
2	1.1065e-05 s	9.879e-06 s
20	0.000230363 s	1.8813e-05 s
32	0.0678012 s	2.0515e-05 s
41	3.26962 s	1.091e-05 s
50	206.996 s	1.0943e-05 s

I then tried running my code on numbers higher than 50, but the time taken to return results was too long. When I ran the code only iteratively, it would return rather quickly. I did some research as to why the recursive code was more time-costly. I learned that recursion repeats code for each step, and is therefore very inefficient. For the Fibonacci sequence, the code calculates the Fibonacci number for n-1 and n-2 and then adds them together. The big O for this algorithm is O(2^n). This means that the run time/memory used will increase exponentially with the number input.

Based on the large discrepancy in run time between the iterative and recursive function, I have learned the importance of analyzing the run time and memory used in an algorithm in order to make sure that a program runs efficiently.