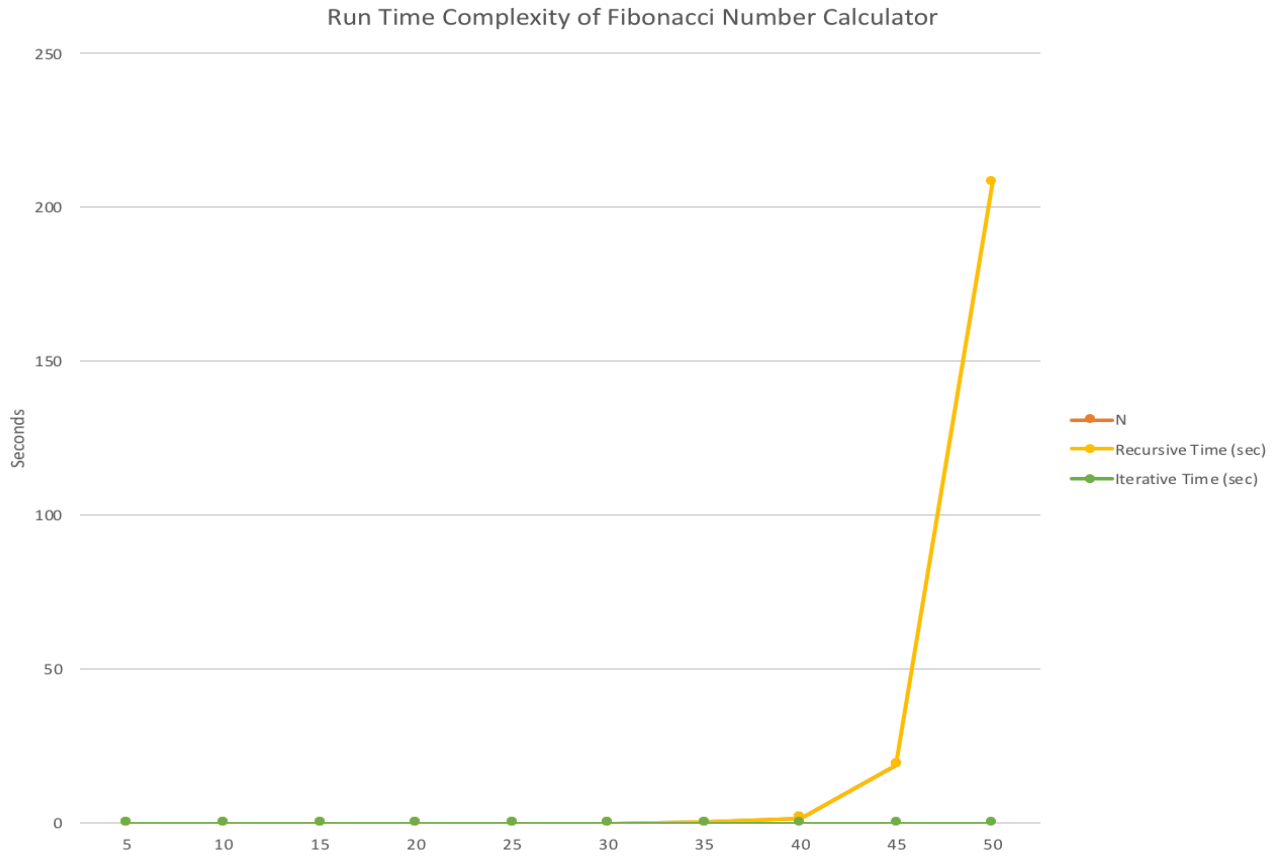
Devin Gendron
Lab 10 Write Up

Pre-Test Thoughts:
My understanding of algorithms involving time complexity is fairly limited. I do however remember that recursion should take much longer than an iterative approach when it comes to time complexity in runtime. With some research beforehand, this seems to depend on what language the recursion is occurring in, as well as a difference between head and tail recursion. Thus, I expect the iterative approach to have a shorter run time than the recursive approach.

N	Recursive Time (sec)	Iterative Time (sec)
5	0.000014	0.000006
10	0.000014	0.000006
15	0.000024	0.000006
20	0.000129	0.000006
25	0.000907	0.000007
30	0.014361	0.000006
35	0.151771	0.000008
40	1.706334	0.000007
45	18.961725	0.000006
50	208.390991	0.000007



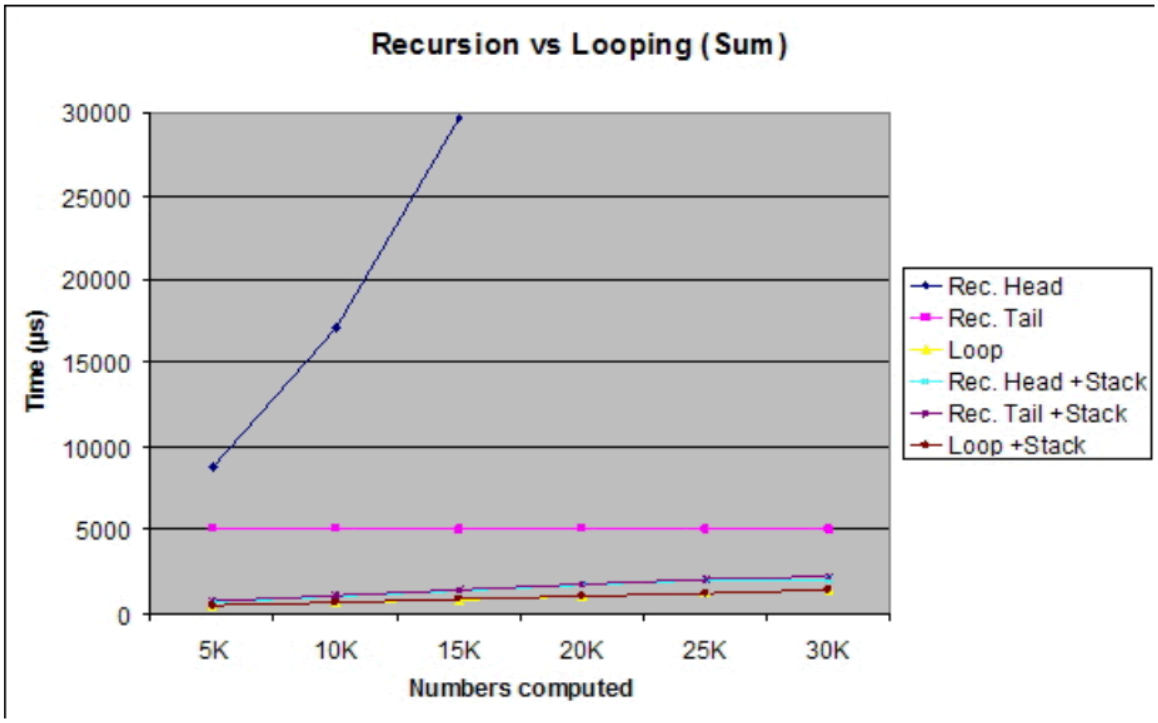
Post-Test Thoughts:
As can be seen in the chart and visualized in the table, the recursive implementation and the iterative implementation are about equal until about a size of 40 for Fibonacci's number calculation. After that, the time for the recursive implementation spikes up, whereas the iterative implementation stays at the same run time.

While reading more information on recursion, I came across this tech journal from IBM:

https://www.ibm.com/developerworks/websphere/techjournal/1307_col_paskin/1307_col_paskin.html

In this journal, IBM has a that table the seemingly resembles my chart. (See below). With this table in mind, one can clearly see that the data represents head recursion. While checking my source code, head recursion can be seen in the implementation borrowed. Thus, my initial speculation was correct – recursion (in certain circumstances due to all the function calls in the stack) is slower than an iterative (looped) approach.

Figure 1. Summation test case



NOTE : Size of N was limited due to IDE not being able to complete larger sizes of N.