Lab 10

Results: It is pretty clear by running each implementation of this program that the recursive implementation of calculating the Nth term of the Fibonacci sequence takes exponentially longer than the iterative implementation. Both implementations take less than 1 second up until about the 40th term of the sequence where the recursive implementation takes 1 second. From then on after the iterative implementation remains at 0 seconds for as far as I could see and the recursive implementation increases exponentially with each incrementation of the Nth term in the Fibonacci sequence.

For Big-O I would say that the recursive implementation is exponential and the iterative implementation is logarithmic.

I believe that the recursive implementation takes so much longer because it has to create new functions and variables each time it is recursively called whereas the iterative implementation does not require any new variable to take up memory space and does not have to call new functions which take up time and memory space.

Running Time in Seconds for Calculating the nth term of the Fibonacci Sequence in an iterative program and a recursive program

Nth term of Fibonacci sequence	**Iterative Implementation**	**Recursive Implementation**
5	0(s)	0(s)
10	0(s)	0(s)
15	0(s)	0(s)
20	0(s)	0(s)
25	0(s)	0(s)
30	0(s)	0(s)
35	0(s)	0(s)
40	0(s)	1(s)
42	0(s)	2(s)
43	0(s)	4(s)
44	0(s)	7(s)
45	0(s)	12(s)
46	0(s)	19(s)
47	0(s)	32(s)
48	0(s)	58(s)

49	0(s)	84(s)
50	0(s)	137(s)