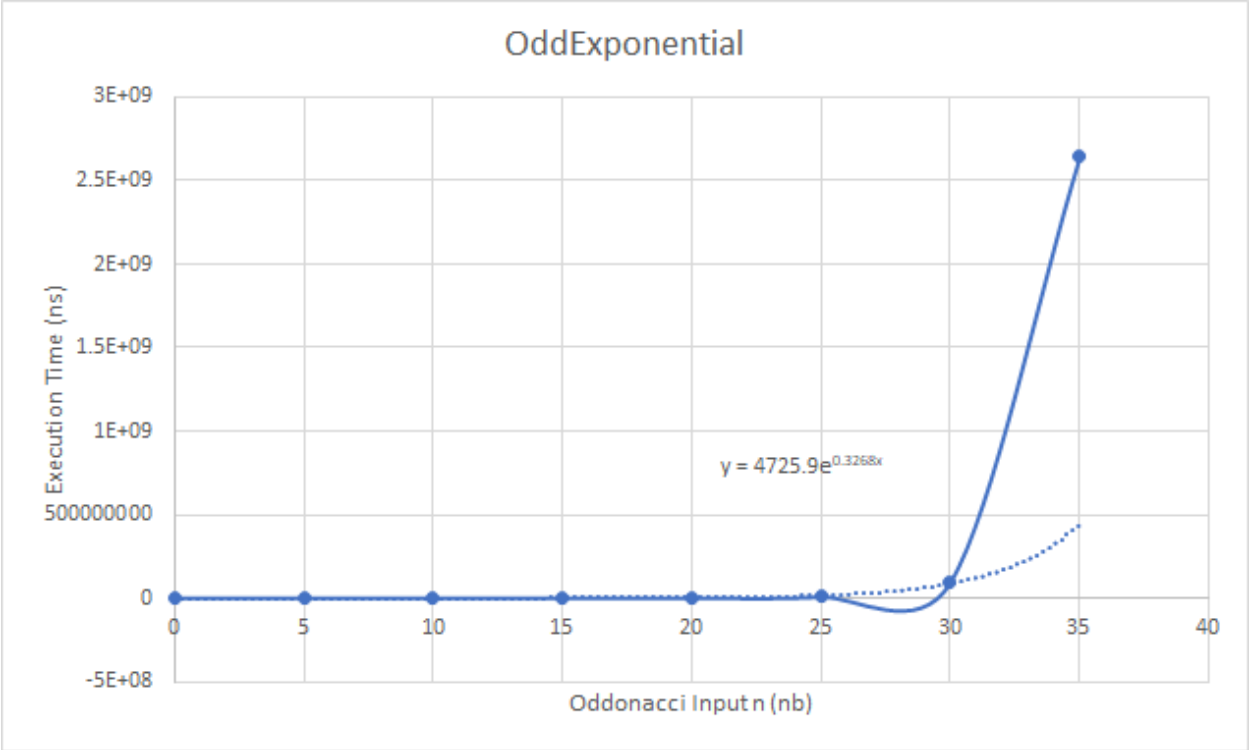
Sofia Valiante 40191897
COMP 352
Assignment 1
October 7th, 2022

Part 2

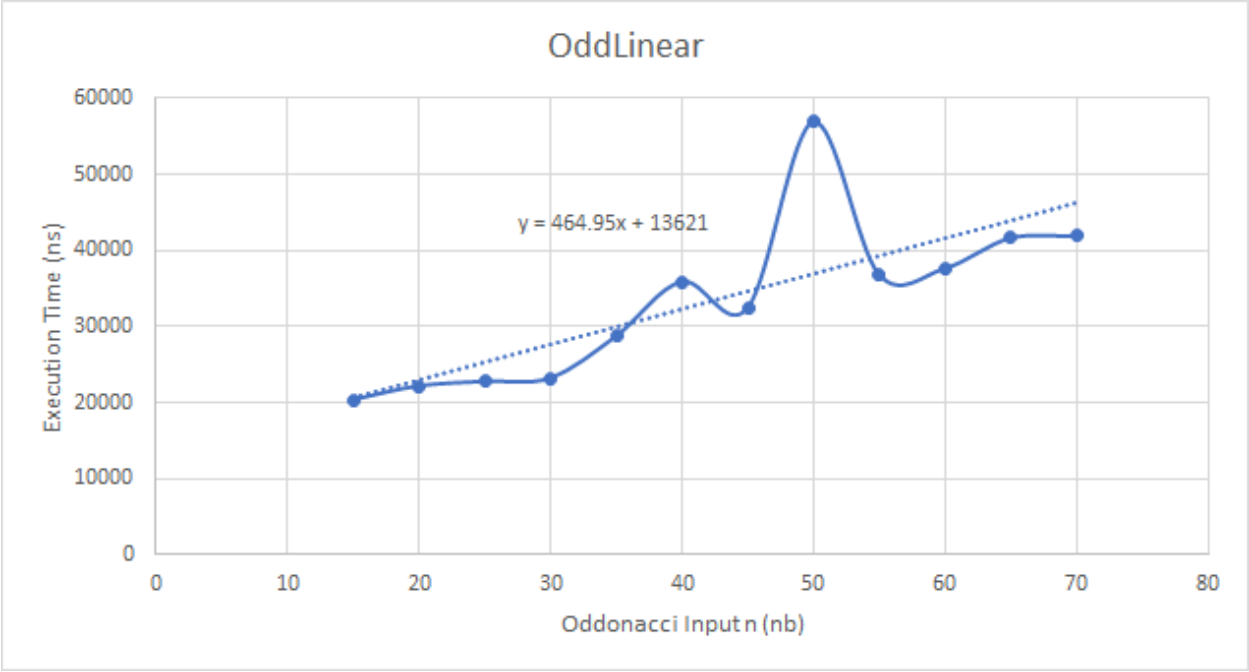
Question 1

a)
```
Algorithm oddLinear(nb, x, y, z)
        Input int nb, long x, long, y, long z
        Output long x
        if nb == 0 then
                return 0
        else if nb == 1 then
                return x
        else
                temp ← x
                x ← y
                y ← z
                z = x + y + temp
                return oddLinear(nb-1, x, y, z)

Algorithm oddExponential(nb)
        Input int nb
        Output long
        if nb == 0 then
                return 0
        else if nb <= 3 then
                return 1
        else
                Return oddExponential(nb-1) + oddExponential(nb-2) + oddExponential(nb-3)
```



The execution time observed during runtime of the OddExponential function has been graphed against the nb input to the functions made in the for-loop. It can be noted that the time of the function increases exponentially close to nb = 25, with a trendline $f(n) = 4725.9e^{0.32n}$, though the time complexity should be close to $O(n) = n^3$.



The execution times obtained by the program were graphed above, and yielded a linear trendline $f(n) = 464.95n + 13621$, as fit of the function OddLinear. The function noticeably has an outlier at

OddLinear(50). Overall, it ran much quicker than OddExponential, and by the graph, does the same input nb at a faster speed.

b) The first function oddExponential is of exponential time complexity because within the method body, it calls itself three times, thus making its time complexity $O(n^3)$. As such, with given input, it takes a method with exponential time complexity n^3 longer to produce results than one of linear time complexity n. The second function oddLinear is of time complexity O(n) since it only calls itself once in the method body, and as such it is much faster at processing input than $O(n^3)$. Whereas $O(n^3)$ only has one parameter, O(n) has 4 parameters set that allow it to solve any bottlenecks of the exponential function. The additional parameters, x, y and z allow the values in the current block of the oddonacci sequence of three values to be carried over and preserved into the next recursive method call, eliminating the need for multiple method calls to obtain the values of the previous three numbers in the oddonacci sequence.

c) Yes, the method oddLinear(nb, x, y, z) uses tail recursion. The values of x, y and z are used as accumulatory parameters, and they're values are preserved through the next method call to be used in the next instance, and so forth. The method is tail recursive because its return value is calculated as the call to itself, whereas oddExponential is not tail recursive because it performs the extra calculation of addition between each of the three calls in its return statement.