CPSC 335: Project 1

Student 1 —
Name: Dylan Wong
Email: dylannwong@csu.fullerton.edu

Student 2 —
Name: Nicklaus Marietta
Email: nick.marietta@csu.fullerton.edu

Algorithm 1 (Left-to-Right):

Pseudocode:

```
Given: n
Multiply list * n
Def left_to_right(some_list):
        For element in range of some_list length - 1: # avoid error
                For i in range of element to some_list length - 1:
                        If (some_list[i] and some_list[ next element (element+1)] == “light:):
                                Set temp to some_list[element]
                                Set some_list[i] to some_list[element+1]
                                Set some_list[element+1] to temp
        Print the list outside of the loop
```

Mathematical Analysis:

The overall time complexity of this algorithm is O(n^2). We will prove this using limits:
Proof:

T(n) our algorithm, is $O(\frac{(n-1)n}{2})$ —> step count

We are going to assume the efficiency class for f(n) is T(n)=O(n^2) for this algorithm

$$\lim_{n \to \infty} f(n)/T(n)$$

$$= \lim_{n \to \infty} \frac{\frac{(n-1)n}{2}}{n^2}$$

$$= \lim_{n \to \infty} \frac{n^2 - 2n}{2n^2}$$

$$= \lim_{n \to \infty} \frac{n^2}{2n^2} - \frac{2n}{2n^2}$$

$$= \lim_{n \to \infty} \frac{1}{2} - \frac{1}{n}$$

$= \frac{1}{2} - 0$

Since the outcome is $\frac{1}{2}$, which is constant, this satisfies case 2 which f(n) = T(n).

Algorithm 2 (Lawnmower):

Pseudocode:

```
Given: list of 2n size
Def lawnmower(some_list):
    Set one pointer called left to 0
    Set one pointer called right to 1
    Set n to the length of some_list
    For element in range(int(n divided by 2)):
        For x in range from left to the n size minus right:
            If some_list at x is equal to “dark” (1) and the next element is “light” (0):
            # this is switching the two elements to trade places
                Set temp to some_list at x
                Set some_list at x to the next element (light)
                Set the next element (light) to the temp
        Increment right by 1
        Increment left by 1 so that they both move through the loop
        For t in range from the size - right to left, reversing the traversal with -1
            If some_list at t is “light” (0) and the element before it is “dark”(1):
                Set temp to some_list at t # light
                Set some_list at j to the previous element # light = dark
                Set the previous element to temp # dark = light
        Increment right by 1
        Increment left by 1
```

Mathematical Analysis:

The overall time complexity of this algorithm is O(n^2). We will prove this using limits:

Proof:

T(n) our algorithm, is $O(\frac{n^2-2n}{2})$ —> step count

We are going to assume the efficiency class for f(n) is T(n)=O(n^2) for this algorithm

$\lim_{n \to \infty}$ f(n)/T(n)

$$= \lim_{n\to\infty} \frac{\frac{n^2-2n}{2}}{n^2}$$

$$= \lim_{n\to\infty} \frac{n^2-2n}{2n^2}$$

$$= \lim_{n\to\infty} \frac{n^2}{2n^2} - \frac{2n}{2n^2}$$

$$= \lim_{n\to\infty} \frac{1}{2} - \frac{1}{n}$$

$$= \frac{1}{2} - 0$$

$$= \frac{1}{2}$$

Since the outcome is $\frac{1}{2}$, which is constant, this satisfies case 2 which f(n) = T(n).