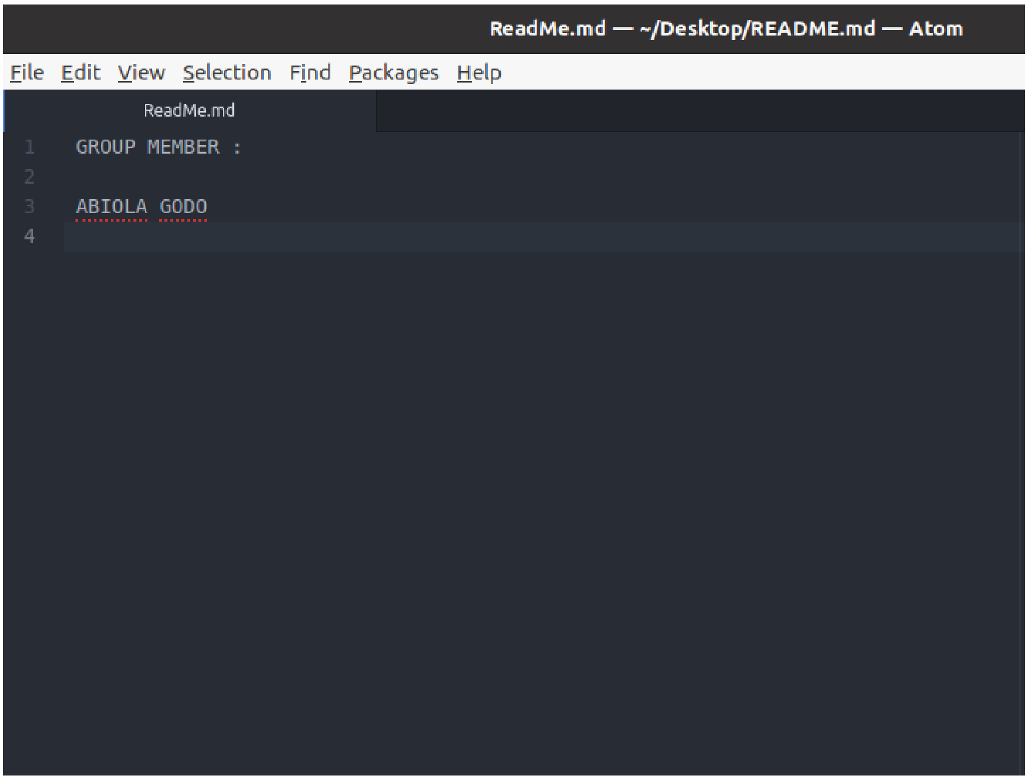
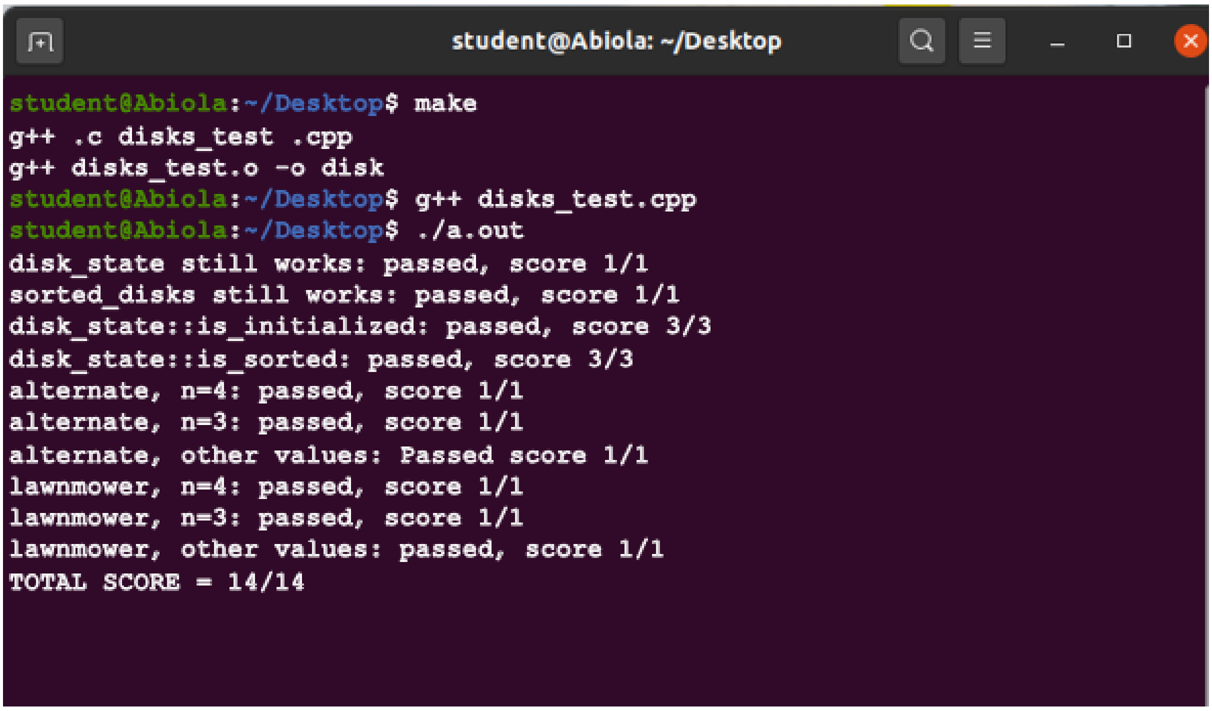
PROJECT 1: Implementing Algorithms

Submitted by:

ABIOLA GODO
Abiola@csu.fullerton.edu

Both screenshots:





Following is a pseudocode of the Lawnmower Algorithm for the given problem.

Algorithm1: Lawnmower Algorithm
Input: n ← a positive integer n
L ← a list of 2n disks of alternating colors dark-light, starting with dark
Output: L’ ← a list of 2n disks, the first n disks are dark, the next n disks are light
m ← an integer representing the number of swaps required

```
1.   m ← 0
2.   L’←L
3.   For i = 1, 2, ..., n/2:
4.           For j = 1, 2, ..., 2n - 1:
5.                   If L’[j] == white and L’[j+1] == black
6.                           swap(L’[j], L’[j+1])
7.                           m := m + 1
8.                   End If
9.           End For
10.          For j = 2n, 2n - 1, ..., 2:
11.                  If L’[j-1] == white and L’[j] == black
12.                          swap(L’[j-1], L’[j])
13.                          m := m + 1
14.                  End If
15.          End For
16.   End For
17.  Return L’, m
```

Complexity Analysis of the Lawnmower Algorithm:

Here the first loop (line 3) takes n / 2 iterations. Inside the first loop, there are two more loops (line 4, line 10). Both of them take 2n - 1 iterations. The contents inside those loops have complexity of O(1).
So the total complexity becomes:

$$(n / 2) * ((2n - 1) + (2n - 1))$$
$$=> (n / 2) * (4n - 2)$$
$$=> n * (2n - 1)$$
$$=> 2n^2 - n$$

So we can say that the complexity of the Lawnmower Algorithm is **$O(n^2)$.**

Following is a pseudocode of the Alternate Algorithm for the given problem.

Algorithm2: Alternate Algorithm
Input: n ← a positive integer n
L ← a list of 2n disks of alternating colors dark-light, starting with dark
Output: L' ← a list of 2n disks, the first n disks are dark, the next n disks are light
m ← an integer representing the number of swaps required

```
1.   m ← 0
2.   L'←L
3.   For i = 1, 2, ..., n:
4.       If i % 2 == 1
5.           For j = 1, 3, ..., 2n - 1:
6.               If L'[j] == white and L'[j+1] == black
7.                   swap(L'[j], L'[j+1])
8.                   m := m + 1
9.               End If
10.          End For
11.      End If
12.      Else
13.          For j = 2, 4, ..., 2n - 2:
14.              If L'[j] == white and L'[j+1] == black
15.                  swap(L'[j], L'[j+1])
16.                  m := m + 1
17.              End If
18.          End For
19.      End Else
20.  End For
21.  Return L', m
```

Complexity Analysis of the Alternate Algorithm:

Here the first loop (line 3) takes n iterations. Inside the first loop, there are two more loops (line 5, line 13). Both of them take n and n - 1 iterations correspondingly. The contents inside those loops have complexity of O(1).
So the total complexity becomes:

$$n * (n + (n - 1))$$
$$=> n * (2n - 1)$$
$$=> 2n^2 - n$$

So we can say that the complexity of the Alternate Algorithm is **$O(n^2)$.**