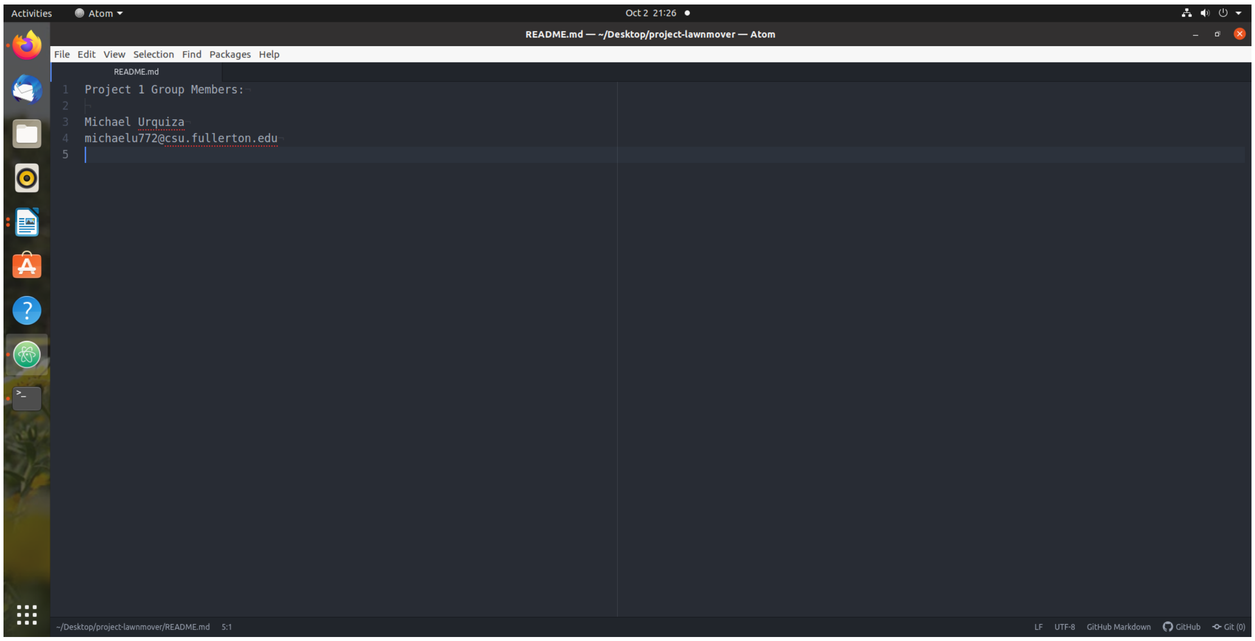
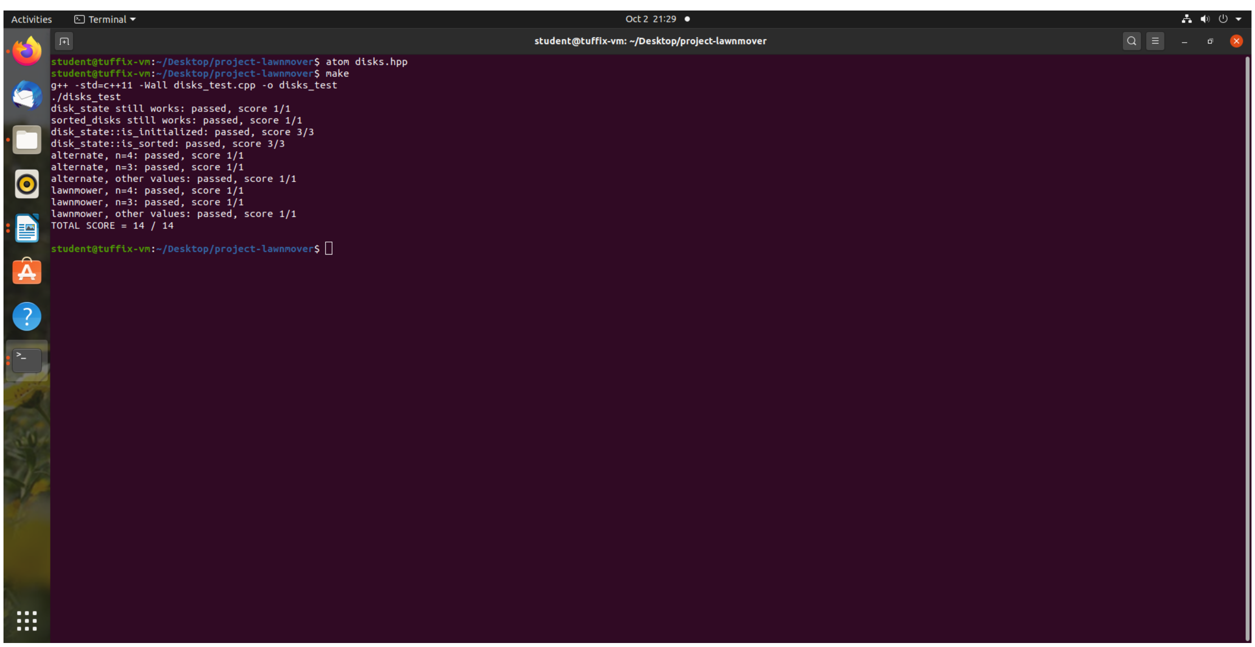
Project 1 Report

Group Members:

Michael Urquiza
michaelu772@csu.fullerton.edu





Alternative Pseudocode

```
swapCount = 0; - 1
For int x = 0 to n - 1: - n
   For int i = 0 to 2n - 2; step 2: - n + 2
      If vector[i + 1] is dark AND vector[i + 2] is dark: - 3
         vector.swap(i + 1); - 3
         ++swapCount; - 1
         endIf
      //endFor
   For int j = 1 to 2n - 2: - 2n - 1
      If vector[j + 1] is dark AND vector[j + 2] is NOT dark: - 3
         vector.swap(j + 1); - 3
         ++swapCount; - 1
         //endIf
   //endIf
//endFor

return vector;
```

Step Count

S.c = 1 + n [(3 + 3 + 1) * (n + 2) + (3 + 3 + 1) * (2n - 1)]

S.c = 1 + n [7 * (n + 2) + 7 * (2n - 1)]

S.c = 1 + n [7n + 14 + 14n - 7]

S.c = 1 + n [21n - 7]

S.c = 1 + 21n^2 - 7n

S.c = 21n^2 -7n + 1

Time Complexity Proof

$$\lim_{n \to \infty} \frac{21n^2 - 7n + 1}{n^3}$$

$$\lim_{n \to \infty} \frac{21}{n} - \frac{7}{n^2} + \frac{1}{n^3}$$

= 0 - 0 - 0

Therefore, the time complexity of this algorithm is O(n^3)

Lawnmower Pseudocode

```
swapCount = 0; - 1
For int x = 0 to n / 2: - n/2 + 1
  For int i = 0 to 2n: - 2n + 1
    If vector[i] is dark AND vector[i + 1] is NOT dark: - 3
      vector.swap(i); - 3
      ++swapCount; - 1
    //endIf
  //endFor
  For int j = 2n - 1 to 0: - 2n
    If vector[j] is dark AND vector[j + 1] is NOT dark: - 3
      vector.swap(j); - 3
      ++swapCount; - 1
    //endIf
  //endFor
//endFor

return vector;
```

Step Count

S.c = $1 + \left(\frac{n}{2} + 1 \right) * [(3 + 3 + 1) * (2n + 1) + (3 + 3 + 1) * (2n)]$

S.c = $1 + \left(\frac{n}{2} + 1 \right) * [(7) * (2n + 1) + (7) * (2n)]$

S.c = $1 + \left(\frac{n}{2} + 1 \right) * [(14n + 7) + (14n)]$

S.c = $1 + \left(\frac{n}{2} + 1 \right) * [(28n + 7)]$

S.c = $1 + \left(14n^2 + \frac{63n}{2} + 7 \right)$

S.c = $14n^2 + \frac{63n}{2} + 8$

Time Complexity Proof

$$\lim_{n \to \infty} \frac{14n^2 + \frac{63n}{2} + 8}{n^3}$$

$$\lim_{n \to \infty} \frac{14}{n} - \frac{63}{2n^2} + \frac{8}{n^3}$$

= 0 - 0 - 0

Therefore, the time complexity of this algorithm is O(n^3)