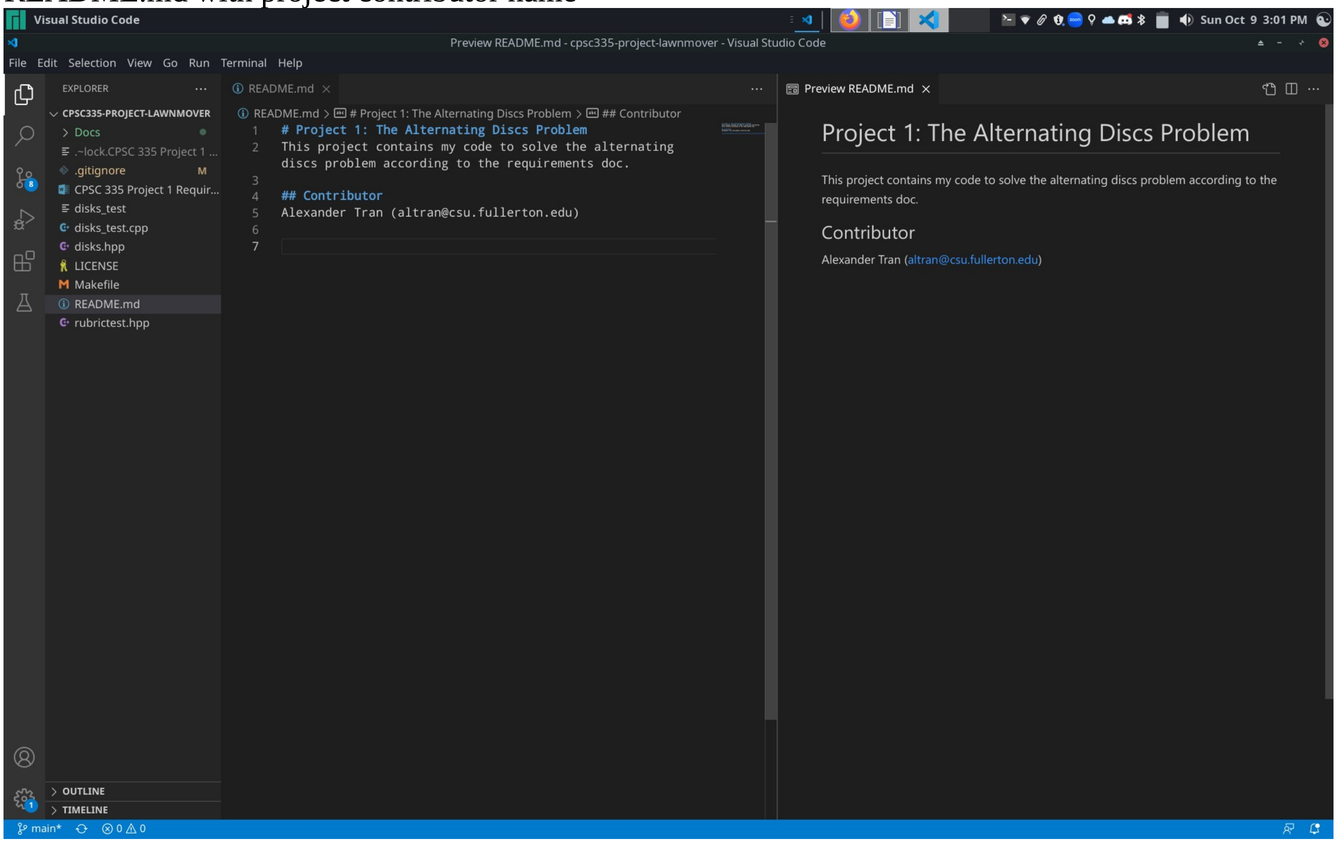
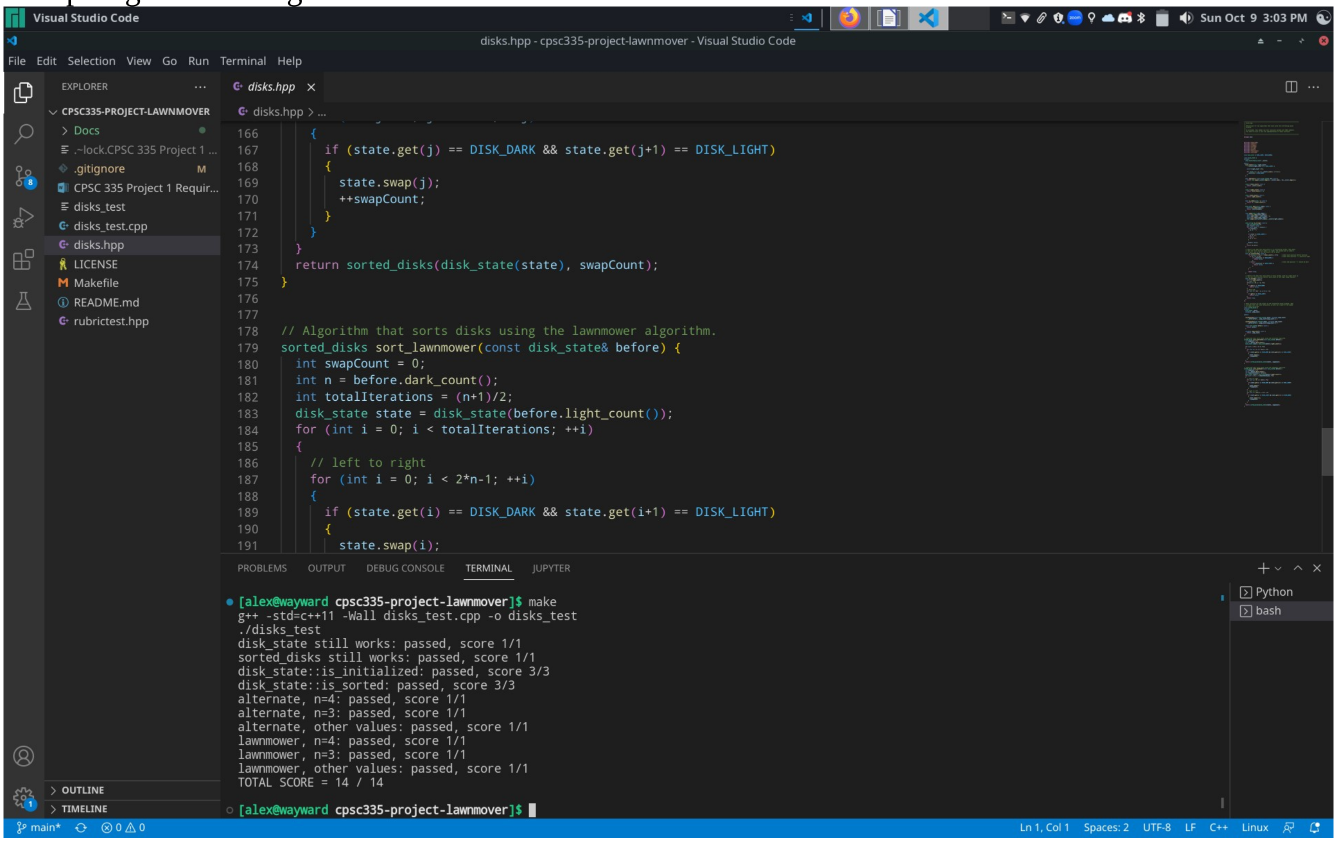
Alexander Tran
CPSC 335-01
Project 1 Report

Project Contributor
Alexander Tran (altran@csu.fullerton.edu)

Screenshots
README.md with project contributor name



Compiling and running



The Algorithms

Let *l* be the given list of size 2*n* in any pseudocode presented.

<u>Lawnmower</u>

```
def lawnmower(n, l):
  iterations = (n+1)/2
  for _ from 0 to iterations:
    # LEFT TO RIGHT
    for i from 0 to 2*n-1:
      if l[i] == 1 and l[i+1] == 0:
        swap(l[i], l[i+1])
    # RIGHT TO LEFT
    for i from 2*n-1 to 0 in steps of -1:
      if l[i] == 0 and l[i-1] == 1:
        swap(l[i], l[i-1])
```

Step count formula: $6n^2+3n-3$

Proof: Lawnmower algorithm has time complexity of $\boldsymbol{O(n^2)}$

Let:
$f(n)=6n^2+3n-3$,
$g(n)=n^2$,
$c=|6|+|3|+|-3|=12$,
and $n_0=5$ (an arbitrary value).

By definition, $f(n)$ is in the order of $g(n)$ if
$f(n)\leq c*g(n)$ where $n\leq n_0$.

$$\begin{aligned} f(n) &\leq c*g(n) \\ 6n^2+3n-3 &\leq 12*n^2 \\ 6(5)^2+3(5)-3 &\leq 12*(5)^2 \\ 150+15-3 &\leq 300 \\ 162 &\leq 300 \end{aligned}$$

Thus, $6n^2+3n-3$ has a time complexity of $O(n^2)$.

Alternate

```
def alternate(n, l):
  for i in range(n+1):
    for j in range(i, 2*n-1):
      if l[j] == 1 and l[j+1] == 0:
        swap(l[j], l[j+1])
```

Step count formula: $\frac{3}{2}(3n^2+n-2)$

Proof: the alternate algorithm has time complexity of $\boldsymbol{O(n^2)}$

Let:

$f(n)=\frac{3}{2}(3n^2+n-2)$,

$g(n)=n^2$,

$c=\left|\frac{9}{2}\right|+\left|\frac{3}{2}\right|+|-3|=9$,

and $n_o=5$ (an arbitrary value).

By definition, $f(n)$ is in the order of $g(n)$ if $f(n)\leq c*g(n)$ where $n\leq n_0$.

$$\begin{aligned} f(n) &\leq c*g(n) \\ \frac{3}{2}(3n^2+n-2) &\leq 9*n^2 \\ \frac{3}{2}(3(5)^2+(5)-2) &\leq 9*(5)^2 \\ \frac{3}{2}(75+5-2) &\leq 225 \\ 117 &\leq 225 \end{aligned}$$

Thus, $\frac{3}{2}(3n^2+n-2)$ has a time complexity of $O(n^2)$.