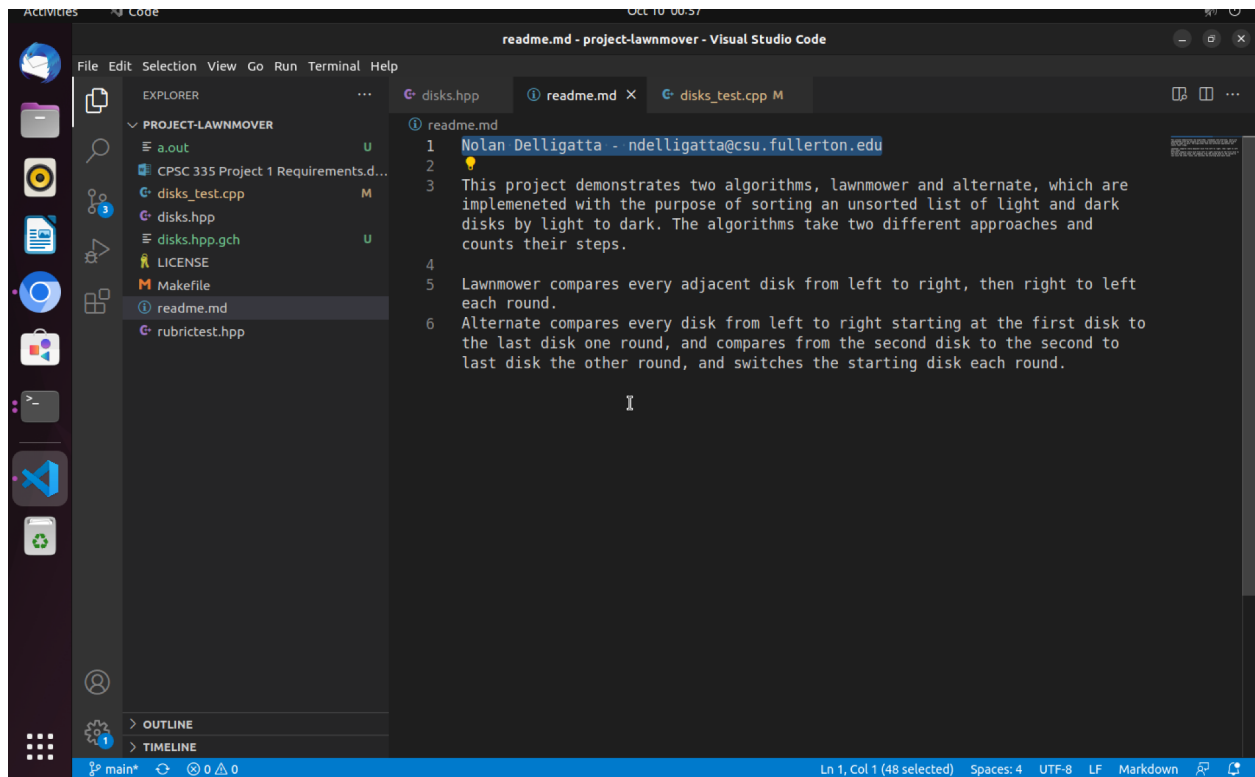


Project 1 PDF

1) Nolan Delligatta - ndelligatta@csu.fullerton.edu

2) Name shown in editor:



The screenshot shows the Visual Studio Code interface with the 'project-lawnmover' workspace. The Explorer sidebar on the left lists files: a.out, CPSC 335 Project 1 Requirements.d..., disks_test.cpp, disks.hpp, disks.hpp.gch, LICENSE, Makefile, readme.md, and rubrictest.hpp. The main editor area displays the 'readme.md' file with the following content:

```
1 Nolan Delligatta - ndelligatta@csu.fullerton.edu
2
3 This project demonstrates two algorithms, lawnmower and alternate, which are
4 implemented with the purpose of sorting an unsorted list of light and dark
5 disks by light to dark. The algorithms take two different approaches and
6 counts their steps.
7
8 Lawnmower compares every adjacent disk from left to right, then right to left
9 each round.
10
11 Alternate compares every disk from left to right starting at the first disk to
12 the last disk one round, and compares from the second disk to the second to
13 last disk the other round, and switches the starting disk each round.
```

The status bar at the bottom indicates 'Ln 1, Col 1 (48 selected)', 'Spaces: 4', 'UTF-8', 'LF', and 'Markdown'.

3)

```

157 sorted_disks sort_alternate(const disk_state& before) {
158     bool oddFlag = false;
159     int start = 0;
160     int numOfSwap = 0;
161
162
163
164 // Algorithm that sorts disks using the lawnmower algorithm.
165 sorted_disks sort_lawnmower(const disk_state& before) {
166     int numOfSwap = 0;
167     disk_state state = before;
168     while(!state.is_sorted()) {
169         if(oddFlag) {
170             start = 1;
171             for(i = start; i < len(S); i += 2) {
172                 if(S[i] > S[i+1]) {
173                     S.swap(i, i+1);
174                     numOfSwap++;
175                 }
176             }
177             oddFlag = !oddFlag;
178         } else {
179             start = 0;
180             for(i = start; i < len(S); i += 2) {
181                 if(S[i] > S[i+1]) {
182                     S.swap(i, i+1);
183                     numOfSwap++;
184                 }
185             }
186             oddFlag = !oddFlag;
187         }
188     }
189     return state;
190 }

```

```

bolab@xix:~/Documents/CPSC/335/project-lawnmower$ g++ disks.hpp
disks.hpp:12:9: warning: #pragma once in main file
12 | #pragma once
    | ~~~~~
bolab@xix:~/Documents/CPSC/335/project-lawnmower$ g++ disks_test.cpp -o disk.out
bolab@xix:~/Documents/CPSC/335/project-lawnmower$ ./disk.out
L D
disk_state still works: passed, score 1/1
sorted disks still works: passed, score 1/1
disk_state::is_initialized: passed, score 3/3
disk_state::is_sorted: passed, score 3/3
alternate, n=4: passed, score 1/1
alternate, n=3: passed, score 1/1
alternate, other values: passed, score 1/1
lawnmower, n=4: L L L L D D D D
passed, score 1/1
lawnmower, n=3: passed, score 1/1
lawnmower, other values: passed, score 1/1
TOTAL SCORE = 14 / 14
bolab@xix:~/Documents/CPSC/335/project-lawnmower$

```

4)

Alternate algorithm pseudocode:

Input: A list L of $2*n$ elements consisting of n light disks and n dark discs

Output: A list S containing the elements of L sorted in order from light disks to dark discs

def alternate(L):

 swaps = 0 <- 1 tu

 start = 0 <- 1 tu

 oddFlag = false <- 1 tu

$S = L$ <- 1 tu

 while(S is not sorted): <- ($n + 1$) tu

 if(oddFlag): <- 1 + max(1, 1) tu

 start = 1 <- 1 tu

 else:

 start = 0 <- 1 tu

 for($i = \text{start}; i < \text{len}(S); i += 2$): <- $n / 2$ tu

 if($i+1 < \text{len}(S)$): <- 2 + max(2 + max(2, 0), 0) tu

 if($S[i] > S[i + 1]$): <- 2 + max(3, 0) tu

$S.\text{swap}(i, i + 1)$ <- 2 tu

 swaps++ <- 1 tu

 oddFlag = !oddFlag <- 1 tu

return S

Step Count = 4 + SC(While)

SC(While) = (n + 1) * SC(If/Else)

SC(If/Else) = 1 + max(1,1) = 1 + 1 = 2

SC(While) = (n + 1) * (2 + SC(For))

SC(For) = (n / 2) * SC(If) = (n / 2) * 7

SC(If) = 2 + max(2 + max(3, 0), 0) = 2 + max(2 + 3, 0) = 2 + max(5, 0) = 2 + 5 = 7

SC(For) = (n / 2) * 7

SC(While) = (n + 1) * (2 + ((n / 2) * 7) + 1) = (n + 1) * (((n / 2) * 7) + 3)

Step Count = 4 + (n + 1) * ((($\frac{n}{2}$) * 7) + 3) = $\frac{7}{2}n^2 + \frac{13}{2}n + 7$

Lawnmower algorithm pseudocode:

Input: A list L of 2*n elements consisting of n light disks and n dark discs

Output: A list T containing the elements of L sorted in order from light disks to dark discs

def lawnmower(L):

swaps = 0 <- 1 tu

T = L <- 1 tu

while(T is not sorted): <- n / 2 tu

for(i = 0; i < len(T); i++): <- n tu

if(i + 1 < len(T)): <- 2 + max(2 + max(3,0),0))

if(T[i] > T[i + 1]): <- 2 + max(3, 0)

T.swap(i, i+1) <- 2 tu

steps++ <- 1 tu

for(j = len(T); j > 0; j--): <- n tu

if(j - 1 > 0): <- 2 + max(2 + max(3,0),0))

if(T[j - 1] > T[j]): <- 2 + max(3,0)

T.swap(j - 1, j) <- 2 tu

steps++ <- 1 tu

return T

Step count = 2 + SC(While)

SC(While) = (n / 2) * SC(For)

SC(For) = n * SC(If) = n * 7

SC(If) = 2 + max(2 + max(3,0),0)) = 2 + max(2 + 3, 0) = 2 + 5 = 7

SC(While) = (n / 2) * SC(For) = (n / 2) * (7n + SC(For))

$$SC(For) = n * SC(If) = n * 7$$

$$SC(If) = 2 + \max(2 + \max(3,0),0) = 2 + \max(2 + 3, 0) = 2 + 5 = 7$$

$$SC(While) = (n / 2) * SC(For) = (n / 2) * (7n + 7n) = (n / 2) * (14n)$$

$$\text{Step count} = 2 + SC(While) = 2 + (n / 2) * (14n) = 7n^2 + 2$$

5)

Alternate algorithm proof:

$$\frac{7}{2}n^2 + \frac{13}{2}n + 7 \in O(n^2)$$

$$\lim_{n \rightarrow \infty} \frac{\frac{7}{2}n^2 + \frac{13}{2}n + 7}{n^2}$$

$$LH = \lim_{n \rightarrow \infty} \frac{7n + \frac{13}{2}}{2n}$$

$$LH = \lim_{n \rightarrow \infty} \frac{7}{2} = \frac{7}{2} \geq 0$$

$$\text{By limit theorem, } \frac{7}{2}n^2 + \frac{13}{2}n + 7 \in O(n^2)$$

Lawnmower algorithm proof:

$$7n^2 + 2 \in O(n^2)$$

$$\lim_{n \rightarrow \infty} \frac{7n^2 + 2}{n^2}$$

$$LH = \lim_{n \rightarrow \infty} \frac{14n}{2n}$$

$$LH = \lim_{n \rightarrow \infty} \frac{14}{2} = 7 \geq 0$$

$$\text{By limit theorem, } 7n^2 + 2 \in O(n^2)$$