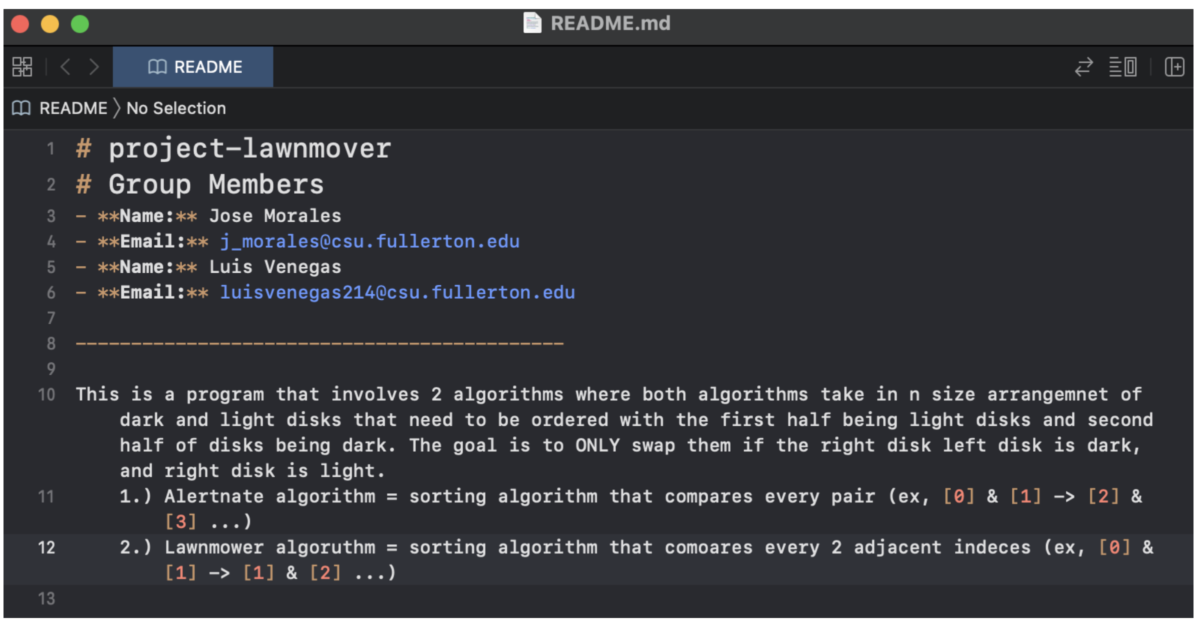
Report

Finally, produce a brief written project report ***in PDF format***. Submit your PDF by committing it to your GitHub repository along with your code. Your report should include the following:

1. Your names, CSUF-supplied email address(es), and an indication that the submission is for project 1.
2. A full-screen screenshot, inside Tuffix, showing the Atom editor or the editor you used, with your group member names shown clearly as below. One way to make your names appear in Atom is to simply open your `README.md`.
3. A full-screen screenshot showing your code compiling and executing.
4. Two pseudocode listings, for the two algorithms, and their step count.
5. A brief proof argument for the time complexity of your two algorithms.

Project	**1**
Jose Morales	**j_morales@csu.fullerton.edu**
Luis Venegas	**luisvenegas214@csu.fullerton.edu**




```
# project-lawnmover
# Group Members
- **Name:** Jose Morales
- **Email:** j_morales@csu.fullerton.edu
- **Name:** Luis Venegas
- **Email:** luisvenegas214@csu.fullerton.edu

--------------------------------------------

This is a program that involves 2 algorithms where both algorithms take in n size arrangemnet of dark and light disks that need to be ordered with the first half being light disks and second half of disks being dark. The goal is to ONLY swap them if the right disk left disk is dark, and right disk is light.
    1.) Alertnate algorithm = sorting algorithm that compares every pair (ex, [0] & [1] -> [2] & [3] ...)
    2.) Lawnmower algoruthm = sorting algorithm that comoares every 2 adjacent indeces (ex, [0] & [1] -> [1] & [2] ...)
```

```
j-morales:project-lawnmover-main j_morales$ make
g++ -std=c++11 -Wall disks_test.cpp -o disks_test
./disks_test
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: passed, score 1/1
lawnmower, n=3: passed, score 1/1
lawnmower, other values: passed, score 1/1
TOTAL SCORE = 14 / 14
```

```cpp
// Algorithm that sorts disks using the alternate algorithm.
sorted_disks sort_alternate(const disk_state& before) {
    int numOfSwap = 0;          //record # of step swap
    for (size_t i = 0; i < state.dark_count(); ++i)
    {
        for (size_t j = 0; j < state.total_count() - 1; ++j)
        {
            if (state.get(j) == Disk_Dark && state.get(j + 1) == DISK_LIGHT)
            {
                state.swap(j);
                numofSwamp;
            }
        }

    }

  return sorted_disks(disk_state(state), numOfSwap);
}


// Algorithm that sorts disks using the lawnmower algorithm.
sorted_disks sort_lawnmower(const disk_state& before) {
    int numOfSwap = 0;
    disk_state state = before;
    for (size_t i = 0; i < state.total_count(); i++) {
        for (size_t j = i; j < state.total_count() - 1; j++) {
            if (state.get(j) == DISK_DARK and state.get(j + 1) == DISK_LIGHT) {
                const_cast <disk_state&> (state).swap(j);
                numOfSwap++;
            }
        }
        for (size_t k = state.total_count() - 1; k > 0; k--) {
            if (state.get(k) == DISK_LIGHT and state.get(k - 1) == DISK_DARK) {
                const_cast <disk_state&> (state).swap(k - 1);
                numOfSwap++;
            }
        }
    }
    return sorted_disks(disk_state(state, numOfSwap);
}
```

```
    // Return true when this disk_state is fully sorted, with all light disks on
    // the left (low indices) and all dark disks on the right (high indices).
    bool is_sorted() const {
        for(size_t i = 0; i < total_count(); i++){
            for(size_t j = i; j < total_count() - 1; j++){
                if(_colors[j] == DISK_DARK and _colors[j+1] == DISK_LIGHT){
                    return false;
                }
            }
            for (size_t k = total_count() - 1; k > 0; k--){
                if(_colors[k] == DISK_LIGHT and _colors[k-1] == DISK_DARK){
                    return false;
                }
            }
        }
        return true;
    }
};
```

Pseudocode for Alternate Algorithm with step-count:

sorted_disks sort_alternate(const disk_state& before) {
Let numOfSwap = 0 -> **1 time unit**
For(size_t i = 0; i < state.dark_count();i++) **->** $\frac{n}{2} \textit{time unit}$
For(size_t j=0; j < state.total_count() -1; j++) -> **n time unit**
If element j of before is dark disk AND element j+1 of before is light disk -> **2 time unit**
Then swap j -> **1 time unit**
Increment numOfSwap ++ -> **1 time unit**
End if
End for
End for
Return sorted_disks with before and numOfSwap as parameters
}

SC: (n/2 * n * 4) +1= 2n^2 +1
—---

Pseudocode for Lawnmower Algorithm with step-count:
sorted_disks sort_lawnmower(const disk_state& before) {
Let numOfSwap = 0 -> **1 time unit**
For size_t i = 0 to length of before
For size_t j = i to length of before - 1
If element j of before is dark disk AND element j+1 of before is light disk-> **2 time unit**
Then swap element j of before -> **1 time unit**
Increment numOfSwap -> **1 time unit**

For size_t k = length of before - 1 to first element in before by decrement of 1 **-> 2-(n-1) / -1**

If element k of before is light disk AND element k - 1 of before is dark disk **-> 2 time unit**

Then swap element k - 1 of before3e2 **-> 1**

Increment numOfSwap -> **1 time unit**

Return sorted_disks with before and numOfSwap as parameters

}

- SC for dep loops = finalvalue_outerloop E initial value outerloop finalvalue inner E initalvalueinnderloop * SCnestingloop

SC for first for loop =

$\sum_{i=0}^{n} \sum_{j=i}^{n-1} \cdot 4$ ..... $\sum_{j=i}^{n-1} \cdot 4 = 4n - 4$ ... $4\left(\frac{n(2n)}{2}\right) = 4n^2$

SC for last for loop = 4(2-(n-1) = -4n+12 / -1 = 4n - 12

SC for lawnmower function = 1 + 4n^2 + 4n-12 = 4n^2 + 4n - 11

—---

Proof for Alternate Algorithm:

$\lim_{n \to \infty} \frac{2n^2+1}{n^2}$

= $\lim_{n \to \infty} \frac{4n}{2n}$

$\lim_{n \to \infty} \frac{4}{2}$

= 2

Proof for Lawnmower Algorithm:

$\lim_{n \to \infty} \frac{4n^2+4n-11}{n^2}$

$\lim_{n \to \infty} \frac{8n+4}{2n}$

$\lim_{n \to \infty} \frac{8}{2}$

= 4