Name: Alexander Zavaleta Alejandro Ramos
Email: A_zavaleta@csu.fullerton.edu alejandroramosh27@csu.fullerton.edu

Github:

https://github.com/zaavaleta33/project-lawnmover-main

Pseudocode and Step Count:

```
sorted_disks sort_alternate(const disk_state & before) do
int numOfSwap = 0; SC -> 1tu
disk_state af = before;  SC -> 1tu
for i = 0 in range total_count SC -> n+1 tu
   for j = 0 in range total_count step 2 SC ->  n/2 + 1 tu
     if j == DISK_DARK && j + 1 == DISK_LIGHT do SC -> 4 tu
       swap(t); SC -> 1 tu
       numOfSwap++; SC -> 1 tu
     end if
   end for
   for t = 1 in range total_count step 2 do SC -> (n-1)/2 + 1tu
     if a == DISK_DARK && a + 1 == DISK_LIGHT do SC -> 4 tu
       swap(a); SC -> 1 tu
       numOfSwap++; SC -> 1 tu
     end if
   end for
end for
return sorted_disks(disk_state(af), numOfSwap);

SC = 1 + 1 + n((n(6)+1)/2)+ n(6)/2+1/2))
SC = 2 + n((n(6)+1)/2 + (n(6)/2 + ½))
SC = 6n^2+n/2 + ((6n^2)/2 + n½) + 2
SC = 12n^2 + 2n/2 + 2

SC = 2 + ( (n+1) * ( (n/2) + 1 + (n-1)/2 + 1) )
SC = n^2 + 5n/2 + 31/2


sorted_disks sort_lawnmower(const disk_state & before) do
   disk_state aft = before; SC -> 1 tu
  int num_swap = 0; SC -> 1 tu
   for i = 0 in range total_count do SC -> (n - 0)/1 + 1 tu -> n + 1 tu
   for j = 0 in range total_count do SC -> (n - 0)/1 + 1 tu -> n + 1 tu
     if (j) == DISK_DARK &&j + 1 == DISK_LIGHT do SC -> 4 tu
       aft.swap(j); SC -> 1 tu
       num_swap++;   SC -> 1 tu
```

```
    end if
  end for
  for t = total_count to 0 do SC -> ((0 - n)/-1) + 1  tu -> n + 1 tu
    if t == DISK_LIGHT && t - 1 == DISK_DARK do SC -> 4 tu
      swap(t - 1); SC -> 1 tu
      num_swap++; SC -> 1 tu
    end if
  end for
 end for
return sorted_disks(disk_state(aft), num_swap);
```

SC = 2 + (((n + 1) + (n + 1)) * n) +(4 + (max(0,2)) + (4 + (max(0,2))
SC = 2 + ((2n + 2) * n)
SC = 2n^2 + 2n + 14

Mathematical Analysis:

SC for sort_alternate:
Let f(n) = n^2 + 5n/2 + 31/2
Let g(n) = n^2
We must prove that f(n) <= c * g(n) for all n >= n0
Let c = 10, let n = 5, then plug into the inequality
(5)^2 + 5(5)/2 + 31/2 <= (10) * (5)^2
51 <= 250
The inequality is true, therefore we can say that yes n^2 + 5n/2 + 31/2 does belong 0(n^2)

SC for sort_lawnmower:
Let f(n) = 2n^2 + 2n + 14
Let g(n) = n^2
We must prove that f(n) <= c * g(n) for all n >= n0
Let c = 10, let n = 15, then plug into the inequality
2(15)^2 + 2(15) + 14 <= (10) * (15)^2
494 <= 2250
The inequality is true, therefore we can say that yes 2n^2 + 2n + 14 does belong 0(n^2)

Video:

Screen Recording 2023-03-18 at 6.56.41 PM.mov

Screenshots:

```
116
117   // Return true when this disk_state is fully sorted, with all light disks on
118   // the left (low indices) and all dark disks on the right (high indices).
119   bool is_sorted() const {
120     for (size_t i = 0; i < total_count() - 1; i++) { //iterate through whole array
121       if (i < total_count() / 2 && get(i) != DISK_LIGHT) { //check's to see if i is
      less than totalcount/2 and if light is not a light disk
122         return false;
123       }
124       if (i > total_count() / 2 && get(i) != DISK_DARK) { //check's to see if i is
      greate then half the total_count and sees if it's a dark disk
125         return false;
126       }
127     }
128     return true;
```

```cpp
// Algorithm that sorts disks using the alternate algorithm.
sorted_disks sort_alternate(const disk_state & before) {
  int numOfSwap = 0; //record # of step swap
  disk_state af = before; //copies everything from before into af
  for (size_t i = 0; i < af.total_count() - 1; i++) { //iterate thorugh each index
    for (size_t j = 0; j < af.total_count() - 1; j += 2) { //iterate through all the odd
      //!this first iteration goes through each and every iteration
      if (af.get(j) == DISK_DARK && af.get(j + 1) == DISK_LIGHT) {
        af.swap(j); ////swaps a and the next index
        numOfSwap++; ////add's to numswap counter
      }
    }
    for (size_t a = 1; a < af.total_count() - 1; a += 2) { //iterats through all the even
      if (af.get(a) == DISK_DARK && af.get(a + 1) == DISK_LIGHT) {
        af.swap(a); //swaps a and the next index
        numOfSwap++; //add's to numswap counter
      }
    }
  }
  return sorted_disks(disk_state(af /*state*/ ), numOfSwap);
}
```

```cpp

// Algorithm that sorts disks using the lawnmower algorithm.
sorted_disks sort_lawnmower(const disk_state & before) {
  //the only move we can make is when one disk is next to another one
  //second algorightm for this seems to be easir
  disk_state aft = before;
  int num_swap = 0;
  //Todo:start from the left and go to each iteration
  for (size_t i = 0; i < aft.total_count() - 1; i++) {
    for (size_t j = 0; j < aft.total_count() - 1; j++) { //iterates from left to right
      if (aft.get(j) == DISK_DARK && aft.get(j + 1) == DISK_LIGHT) {
        aft.swap(j); //swaps from left to right
        num_swap++; //increases the numswap
      }
    }
    for (size_t t = aft.total_count() - 1; t > 0; t--) { //iterates from right to left
      if (aft.get(t) == DISK_LIGHT && aft.get(t - 1) == DISK_DARK) {
        aft.swap(t - 1); //swaps from right to left
        num_swap++; //increases the numswap
      }
    }
  }
  //Todo: Once you reach the end you go from right to left
  return sorted_disks(disk_state(aft), num_swap);
}
```

```
main          Makefile
main-debug    'project-lawnmover-main REAL'
main.o        replit.nix
> cd 'project-lawnmover-main REAL'
> ls
'CPSC 335 Project 1 Requirements.docx'
 disks.hpp
 disks_test
 disks_test.cpp
 LICENSE
 Makefile
 rubrictest.hpp
> g++ 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

>
```