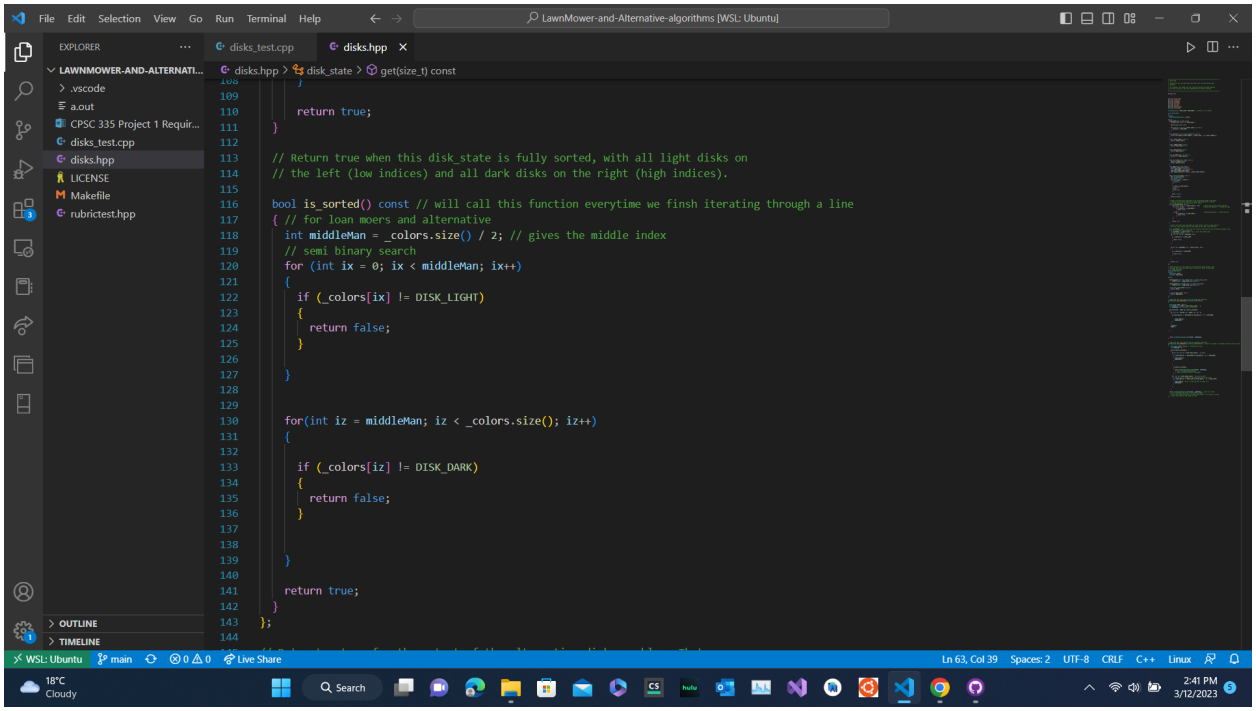
Ryan Haddadi: mochi456@csu.fullerton.edu

Naeem Khayat Albirkdar: arnaeem@csu.fullerton.edu

Code proof using Linux environment:

Is_sorted function:



Sort_alternate:

disks.hpp > disk_state > get(size_t) const

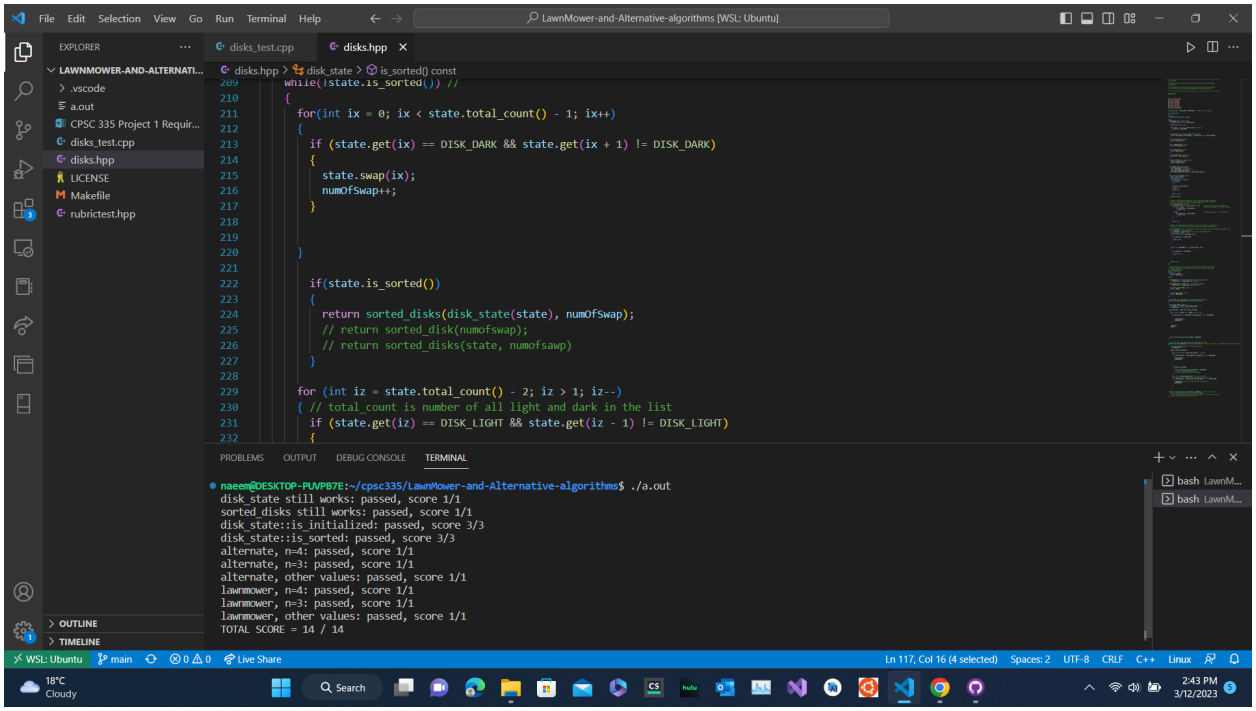
```
  const disk_state& after() const {
    return _after;
  }

  unsigned swap_count() const {
    return _swap_count;
  }
};

// Algorithm that sorts disks using the alternate algorithm.
sorted_disks sort_alternate(const disk_state& before)
{

  disk_state state = before;
  int startAt = 0, endAt = state.total_count() - 1;
  int numOfSwap = 0; //record # of step swap

  while(startAt < endAt && !state.is_sorted())
  {
    for (int ix = startAt; ix < endAt; ix = ix + 2)
    {
      if(state.get(ix) == DISK_DARK && state.get(ix + 1) != DISK_DARK)
      {

          state.swap(ix);
          numOfSwap++;
      }
    }

    startAt++;
    endAt--;


  }
```

sort_lawnmoer:

disks.hpp > disk_state > is_sorted() const

```
// Algorithm that sorts disks using the lawnmower algorithm.
sorted_disks sort_lawnmower(const disk_state& before) // before is already in the member function of disk state class
{
    disk_state state = before; // accessing the class
    int numOfSwap = 0;

    while(!state.is_sorted()) //
    {
      for(int ix = 0; ix < state.total_count() - 1; ix++)
      {
        if (state.get(ix) == DISK_DARK && state.get(ix + 1) != DISK_DARK)
        {
          state.swap(ix);
          numOfSwap++;
        }


      }

        if(state.is_sorted())
        {
          return sorted_disks(disk_state(state), numOfSwap);
          // return sorted_disk(numofswap);
          // return sorted_disks(state, numofsawp)
        }

      for (int iz = state.total_count() - 2; iz > 1; iz--)
      { // total_count is number of all light and dark in the list
        if (state.get(iz) == DISK_LIGHT && state.get(iz - 1) != DISK_LIGHT)
        {
          state.swap(iz - 1);// to swap the next to index iz-2
          numOfSwap++;
        }

      }

    }
```

output:



sort_alternate: pseudocode, time unit, time complexity, and limits

project 1 submission

```
#sort_alternate : pseudocode

set start_at to first disk
set end_at to last_disk - 1

while start_at less than end_at
for every other disk in the list from start_at to end_at
        if disk isn't identical to adjacent disk (and its
        DARK)
            call swap function
        end if

end for
increment start_at
decrement end_at

end while
```

Step count:

$$SC = \sum_{i=0}^{n} \sum_{j=0}^{n} 6 + \max(3,0)$$

inner loop:

$$\sum_{j=0}^{n} 9 \quad \text{constant series}$$

$$\sum_{j=1}^{n} 9n$$

outer loop

$$\sum_{i=0}^{n} 9n \quad \text{constant series} \Rightarrow \sum_{i=1}^{n} 9n$$

$SC = 9n^2 \Rightarrow$ time complexity is $O(n^2)$

- test the efficiency to see if the $9n^2 \in \Theta(n^2)$:

- <u>Using limit theorem</u>

$$\lim_{n \to \infty} \frac{f(n)}{g(n)}$$

$$\Rightarrow \lim_{n \to \infty} \frac{9n^2}{n^2}$$

$$= 9 \Rightarrow f(n) \geq 0 \text{ therefore } f(n) \in g(n)$$

sort_lawnmoer: pseudocode, time unit, time complexity, and limits

```
#Sort_lawnMoer: pseudo code

while not is_sorted

        for each disk from left to right
                if disk is black and adjacent[+1] dis isn't black
                        call swap function
                end if
        end for


        if IS_Sorted
                return

        for each disk from right to left
                if disk is white and adjacent[-1] disk isn't
                white
                        call swap function
                end if

        end for

end while

return
```

Step count: WL: while loop, WLB: while loop block

$$SC = 2 + SC_{WL}(n * (SC_{WLB}))$$

$$SC_{WLB1} = ((n-1)-0+1) * (6 + \max(2,0))$$

$$= n(8) \Rightarrow SC_{WLB1} = 8n$$

: WLB1 is the first for loop

$$SC_{WLB2} = \left(1 - \frac{(n-2)}{-1} + 1\right) * (6 + \max(3,0))$$

$$= 2 + (1 - (-n+2) + 1) * (6+3)$$

$$= (\cancel{2} + 1 + n - \cancel{2} + 1) * 9$$

$$= \cancel{11+n}(2+n) * 9$$

$$SC_{WLB2} = 18 + 9n$$

: WLB2 is the second for loop Block

entire lawn mower function

$$SC = n + 2 + n * (18 + 9n + 8n))$$

$$= n + 2 + n * (17n + 18)$$

$$\boxed{SC = n + 2 + 17n^2 + 18n}$$

time complexity is highest power because the other terms are lower ~~power~~ power which won't affect the n^2 so Time complexity is $O(n^2)$

- test efficiency to see that $17n^2 + 19n + 2 \in O(n^2)$:

- Using limit theorem:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{17n^2 + 19n + 2}{n^2}$$

$$= \lim_{n \to \infty} \frac{17\cancel{n^2}}{\cancel{n^2}} + \frac{19\cancel{n}}{\cancel{n^2}} + \frac{2}{n^2}$$

$$\lim_{n \to \infty} 17 + \frac{19}{n} + \frac{2}{n^2}$$

is_sort: pseudocode, time unit, time complexity

$= 17 + \frac{19}{\infty} + \frac{2}{\infty}$

$= 17 + 0 + 0$

$= 17$

because $\overset{=17}{f(n)} \geq 0$ such that $f(n) \in g(n)$

is_sorted: pseudocode

~~for dis is~~
for first half of disks
    if disk is not light
        fail

for second half of disks
    if disk is not dark
        fail

return success

- Step count:

$SC = 2 + (SC_{FL1}) + (SC_{FL2})$: FL1, FL2 are first and second for loop in is_sorted

$SC_{FL1} = (\#\text{iterations}) * SC_{FLB1}$ for loop block

$\#\text{iterations} = \frac{n}{2} - 0 + 1$
$= \frac{n}{2} + 1$

$SC_{FL1} = \left(\frac{n}{2} + 1\right) \times SC_{FLB1}$

$SC_{FLB1} = 1 + Max(0,0)$

$SC_{FLB1} = 1$

$SC_{FL1} = (\frac{n}{2} + 1) * 1$

① $\boxed{SC_{FL1} = \frac{n}{2} + 1}$

- $SC_{FL2} = (\#iterations) * (SC_{FLB2})$

$\#iterations = n - \frac{n}{2} + 1$

$= \frac{n}{2} + 1$

$SC_{FLB2} = 1 + Max(0,0)$

$= 1$

$SC_{FL2} = (\frac{n}{2} + 1) * 1$

② $\boxed{SC_{FL2} = \frac{n}{2} + 1}$

- Now, the sc of the entire is_sorted function:

$SC = 2 + (\frac{n}{2} + 1) + (\frac{n}{2} + 1)$

$= 4 + \frac{n}{2} + \frac{n}{2}$

$SC = 4 + n$

time complexity is $O(n)$