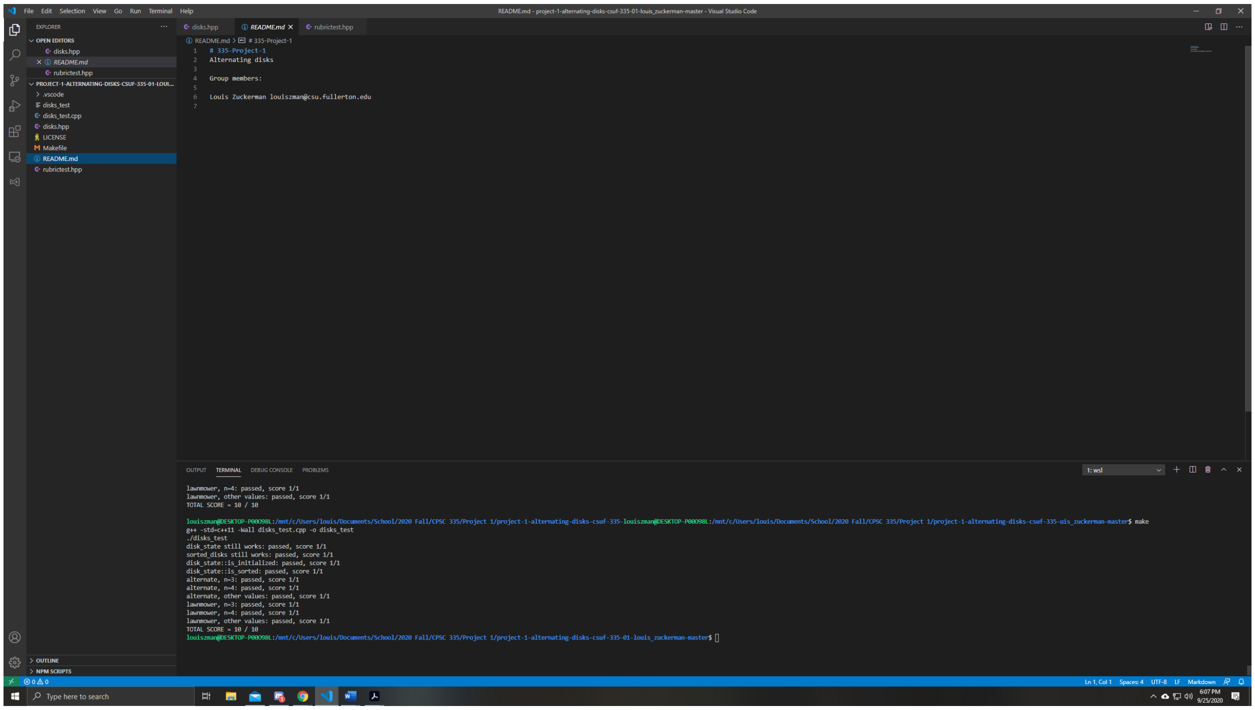
CPSC 335 – Algorithms
Louis Zuckerman
Project 1
Due: 09/25/2020
Email: louiszman@csu.fullerton.edu

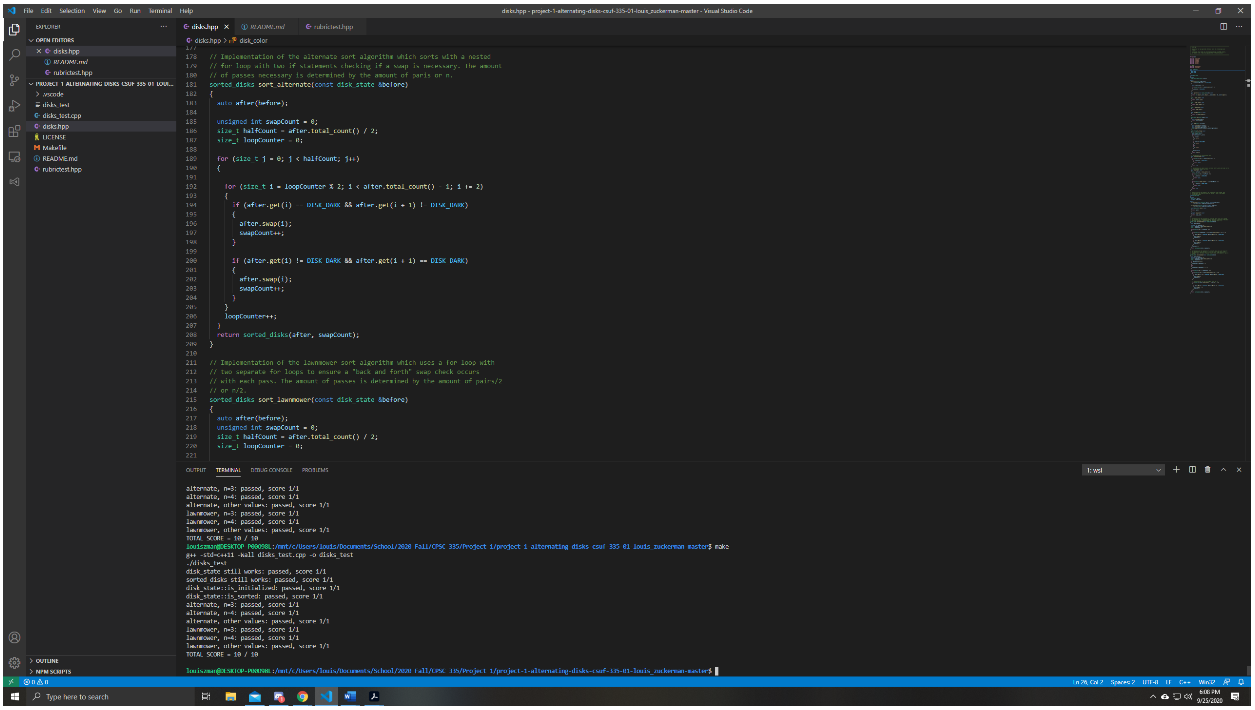
Screenshot Proof of Project

Screenshots of coding environment with name



(Visual Studio Code on Windows 10)

CPSC 335 – Algorithms
Louis Zuckerman
Project 1
Due: 09/25/2020
Email: louiszman@csu.fullerton.edu
Screenshot of make file execution showing TOTAL SCORE = 10 / 10



Algorithm Pseudocode

Input: a positive integer n and a list of $2n$ disks of alternating colors light-dark, starting with light

Output: a list of $2n$ disks, the first n disks are dark, the next n disks are light, and an integer m representing the number of swaps to move the light ones after the dark ones

Lawnmower algorithm:

sorted_disks lawnmower(array[])

{

Initialize swapCounter;

Initialize countofPairs;

Initialize loopCounter;

For (I =0; I < n/2; i++){

For (j = 0; j < 2n-1; j+=2){

CPSC 335 – Algorithms
Louis Zuckerman
Project 1
Due: 09/25/2020
Email: louiszman@csu.fullerton.edu

```
If (array[j+1] == Dark disk and array[j] != Dark disk)

{

Swap array[j] with array[j+1];

Increment swapCounter;

} endif

} endfor

For (k = 2n-2; k>0; k-=2)

{

If (array[k] == Dark disk and array[k-1] != Dark disk)

{

Swap  array[k-1] with array[k];

Increment swapCounter;

} endif

} endfor

} endfor

Return sorted_disks(array[], swapCounter);

} end function
```

Alternate algorithm for sorting 2n disks of alternating colors, light and dark

```
Sorted_disks alternateSort(array[])

{

Initialize swapCounter;

Initialize countofPairs = 2n/2;

Initialize loopCounter;


For (I = 0; I < countofPairs; i++)
```

CPSC 335 – Algorithms
Louis Zuckerman
Project 1
Due: 09/25/2020
Email: louiszman@csu.fullerton.edu

```
{
For (j = loopCounter %2; j < 2n-1; j+=2)
{
If (array[j] == Dark disk and array[j+1] != Dark disk)
{
Swap array[j] with array[j+1];
Increment swapCounter;
} endif
If (array[j] != Dark disk and array[j+1]== Dark disk)
{
Swap array[j] with array[j+1];
Increment swapCounter;
} endif
} endfor
Increment loopCounter;
} endfor
Return sorted_disks(array[], swapCounter);
} end function
```

Mathematical Analysis of Algorithms

Lawnmower algorithm:

```
sorted_disks lawnmower(array[])
{
Initialize swapCounter;         // 1 step;
Initialize countofPairs;        // 1 step;
Initialize loopCounter;         // 1 step;
```

CPSC 335 – Algorithms
Louis Zuckerman
Project 1
Due: 09/25/2020
Email: louiszman@csu.fullerton.edu

```
For (I =0; I < n/2; i++){            // (n/2)-1 steps

For (j = 0; j < 2n-1; j+=2){         // (2n-1-1)/2 steps

If (array[j+1] == Dark disk and array[j] != Dark disk)          // 4 steps

{

Swap array[j] with array[j+1];          //        7 steps

Increment swapCounter;        // 1 step

} endif           // if block1 = 12 steps

} endfor          // forBlock1 = 12*(2n-2)/2 * (n/2-1) = (12n-12)*(n/2-1) = 6n^2-12n-6n+12= 6n^2-
18n+12 steps

For (k = 2n-2; k>0; k-=2)          // (-2n+2)-1)/-2 steps = n-1/2 steps

{

If (array[k] == Dark disk and array[k-1] != Dark disk)          // 4 steps

{

Swap  array[k-1] with array[k];          // 7 steps

Increment swapCounter;        // 1 step

} endif  //ifblock2 = 12 steps

} endfor          // (n-1/2)*12 = 12n-6 steps

} endfor          // forblock1*forblock2 = (6n^2-18n+12)*( 12n-6) = 72n^3-36n^2-216n^2+108n-
+144n-72+3 = 72n^3-252n^2+252n-69 steps

Return sorted_disks(array[], swapCounter);

} end function
```

72n^3-252n^2+252n-69 steps while removing constants and assuming that n^2, n, and constants exist within the domain of n^3 we can say that the lawnmower algorithm has a big O complexity of:

$$O(n^3)$$

Alternate algorithm for sorting 2n disks of alternating colors, light and dark

CPSC 335 – Algorithms
Louis Zuckerman
Project 1
Due: 09/25/2020
Email: louiszman@csu.fullerton.edu

```
Sorted_disks alternateSort(array[])

{

Initialize swapCounter;          // 1 step;

Initialize countofPairs = 2n/2;          // 1 step;

Initialize loopCounter;          // 1 step;


For (I = 0; I < countofPairs; i++)          // (n-1) steps

{

For (j = loopCounter %2; j < 2n-1; j+=2)          // ((2n-1)-(1/2))/2 = 2n-1/2 /2 = n-1/4 steps

{

If (array[j] == Dark disk and array[j+1] != Dark disk)          // 4 steps

{

Swap array[j] with array[j+1];          // 7 steps (depends on swap function implementation)

Increment swapCounter;          // 1 step

} endif          // ifblock1 =  12 steps

If (array[j] != Dark disk and array[j+1]== Dark disk)          // 4 steps

{

Swap array[j] with array[j+1];          // 7 steps

Increment swapCounter;          // 1 step

} endif          // ifblock2 = 12 steps

} endfor          //ifblock1+ifblock2 * forblock2 = 12+12 *n-1/4 = 24n-8 steps

Increment loopCounter;          // 1 step

} endfor          1+forblock1*forblock2= 1+(n-1) * (24n-8) = 1+24n^2-8n-24n+8 = 24n^2-32n+9
steps

Return sorted_disks(array[], swapCounter);

} end function
```

CPSC 335 – Algorithms
Louis Zuckerman
Project 1
Due: 09/25/2020
Email: louiszman@csu.fullerton.edu
24n^2-32n+9 steps while removing constants and assuming that n, and constants exist within the domain of n^2 we can say that the alternating algorithm has a big O complexity of:

$$O(n^2)$$