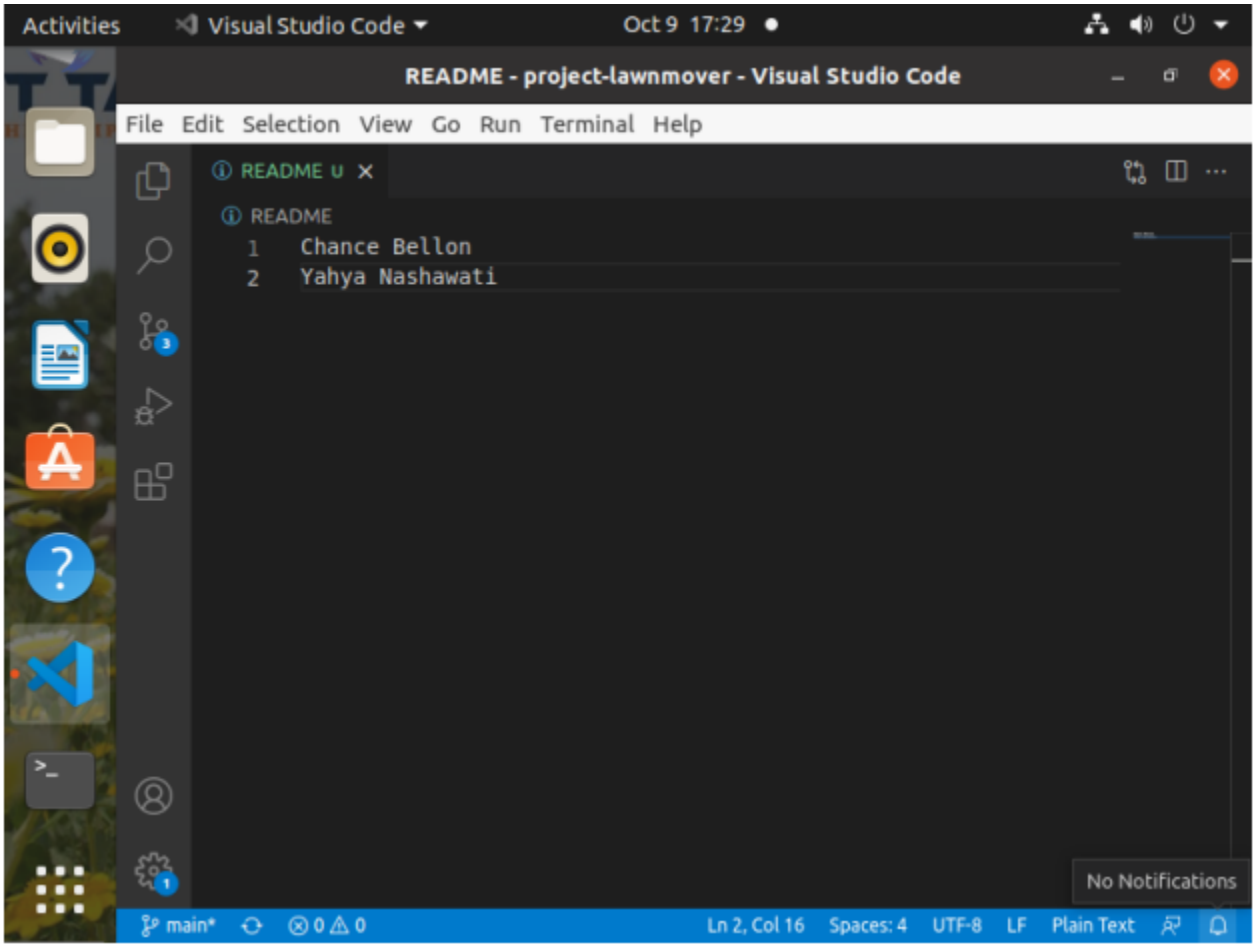
Project One Report

Group Members:

Yahya Nashawait yahyanashawati@csu.fullerton.edu

Chance Bellon chancebellon@csu.fullerton.edu




```
student@tuffix-vm:~/Desktop/Algorithm Engineering Projects/project-lawnmover$ g++ disks_test.cpp
student@tuffix-vm:~/Desktop/Algorithm Engineering Projects/project-lawnmover$ ./a.out
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
```

Project One Report

Pseudocode:

Sort_Alternate:

```
initialize numOfSwap to zero
initialize disk_state state to before

for loop (initialize I to zero; I < count+1; increment I)
  if statement (I modular by two equals to zero)
    for loop (initialize index to zero; compare index to count-1; set index equal to index+2)
      if statement (current disk != next disk)
        if statement (current disk equals dark disk & next disk equals light disk)
          swap disks
          increment numOfSwap
        endif
      endif
    endfor
  endif
  else
    for loop (initialize index to one; index < count-2; set index equal to index+2)
      if statement (current disk != next disk)
        if statement (current disk equals dark disk & next disk equals light disk)
          swap disks
          increment numOfSwap
        endif
      endif
    endfor
  endelse
endfor
```

Sort_Lawnmower:

```
initialize numOfSwap to zero
initialize disk_state state to before

for loop (initialize I to zero; I < n/2; increment I)
  initialize index to zero
  while (index+1 < n)
    if statement (current disk != next disk)
      if statement (current disk equals dark disk & next disk equals light disk)
          swap disks
          increment numOfSwap
      endif
    endif
    increment index
  endwhile
  while (index > 0)
    if statement (previous disk != current disk)
      if statement (previous disk equals dark disk & current disk equals light disk)
          swap disks
          increment numOfSwap
      endif
    endif
    reduce index
  endwhile
endfor

return sorted disks
```

Project One Report

```
// Algorithm that sorts disks using the alternate algorithm.
sorted_disks sort_alternate(const disk_state &before)
{
  int numOfSwap = 0;                              //record # of step swap        1 tu
  disk_state state = before;                                                     1 tu

  for (int i = 0; i < state.total_count() + 1; i++)                              n+1 tu
  {
    if (i % 2 == 0)                                                              2 tu
    {
      for (int index = 0; index < state.total_count() - 1; index = index + 2)    n-1 tu
      {
        if (state.get(index) != state.get(index + 1))                            4 tu
        {
          if (state.get(index) == DISK_DARK && state.get(index + 1) == DISK_LIGHT) 6 tu
          {
            state.swap(index);                                                   1 tu
            numOfSwap++;                                                         1 tu
          }
        }
      }
    }
    else
    {
      for (int index = 1; index <state.total_count() - 2; index = index + 2)     n-2 tu
      {
        if (state.get(index) != state.get(index + 1))                            4 tu
        {
          if (state.get(index) == DISK_DARK && state.get(index + 1) == DISK_LIGHT) 6 tu
          {
            state.swap(index);                                                   1 tu
            numOfSwap++;                                                         1 tu
          }
        }
      }
    }
  }

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

12n^2 + 2n – 8, where n is any positive integer representing the number of iterations from total_count()

The time complexity behind the Sort Alternate Algorithm is $O(n^2)$. The Sort Alternate Algorithm is n^2 time complexity because it clearly has 2 nested loops within each other.

Project One Report

```
// Algorithm that sorts disks using the lawnmower algorithm.
sorted_disks sort_lawnmower(const disk_state &before)
{
  int numOfSwap = 0;                                  //record # of step swap          1 tu
  disk_state state = before;                                                           1 tu

  for (int i = 0; i < state.total_count() / 2; i++)                                    0 tu
  {
    int index = 0;                                                                     1 tu
    while (index + 1 < state.total_count())
    {
      if (state.get(index) != state.get(index + 1))                                    4 tu
      {
          if (state.get(index) == DISK_DARK && state.get(index + 1) == DISK_LIGHT)     6 tu
          {
            state.swap(index);                                                         1 tu
            numOfSwap++;                                                               1 tu
          }
      }

      index++;                                                                         1 tu
    }
    while (index > 0)                                                                  index tu
    {
      if (state.get(index - 1) != state.get(index))                                    4 tu
      {
          if (state.get(index - 1) == DISK_DARK && state.get(index) == DISK_LIGHT)     6 tu
          {
            state.swap(index - 1);                                                     1 tu
            numOfSwap++;                                                               1 tu
          }
      }

      index--;                                                                         1 tu
    }
  }

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

13n^2-6n+2, where n is any positive integer representing the number of iterations from total_count()

The time complexity behind the Lawnmower Algorithm is $O(n^2)$. Although it gets a little messy with the internal side by side while loops it must be noted that the outer loop had a nested loop inside of it, this means that it is n^2.