

**Task 2:**

The add item function that starts from the end is fairly simple and easy to implement. It starts by comparing the last value of the array to the item being inserted. If the item being inserted is less than the last item, then it bumps the last item back one place and continues through the list. This method is a lot more efficient than adding an item from the front, but it still uses a good amount of operations, especially if the item has to go all the way to the front of the array.

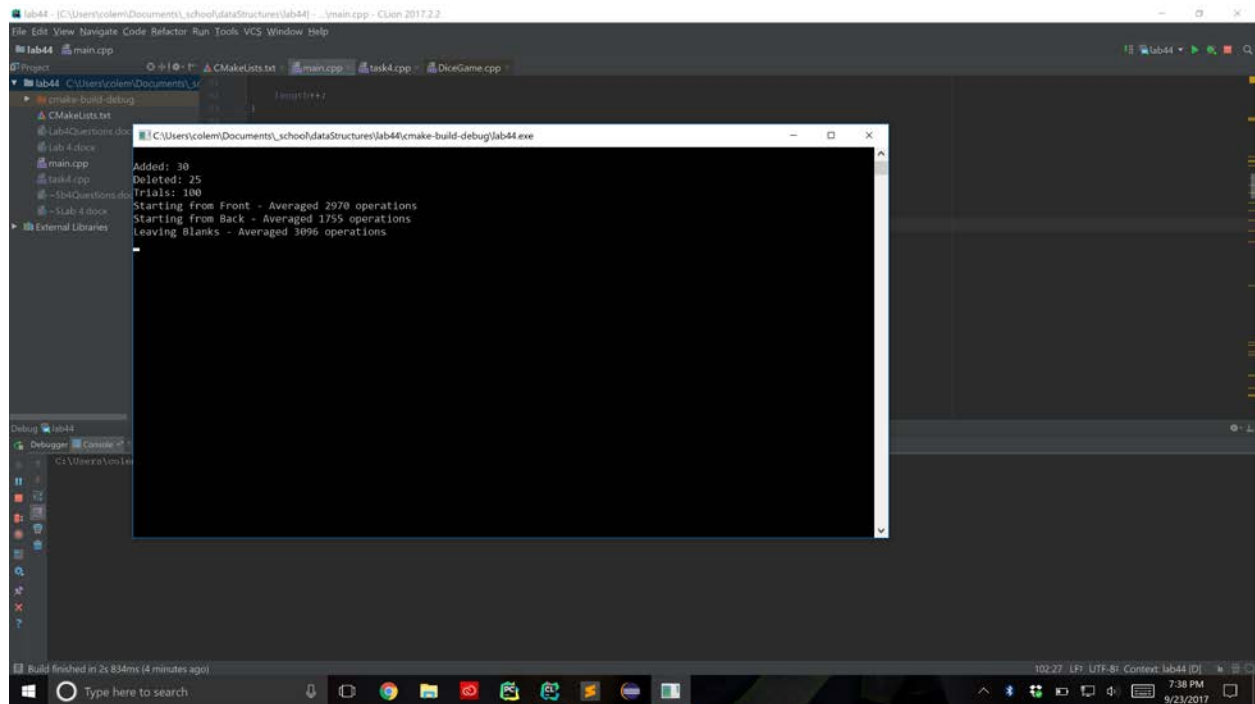
**Task 3:**

The add item halfway between method is a lot more efficient than the add item from the back method. This is because if there is an empty space (NULL) between the two objects that the item must be inserted between, then no other operations must be performed. This works by finding the position where the item must be inserted, checking to see if there is an empty space, and then finding the middle of the empty space.

The remove item function that leaves an empty space (NULL) is extremely efficient. The only operation that must be performed is the locating of the item being removed. After it is removed a new NULL item is put in its place. This works by going through the list until the item is located, and then replacing it with a NULL object.

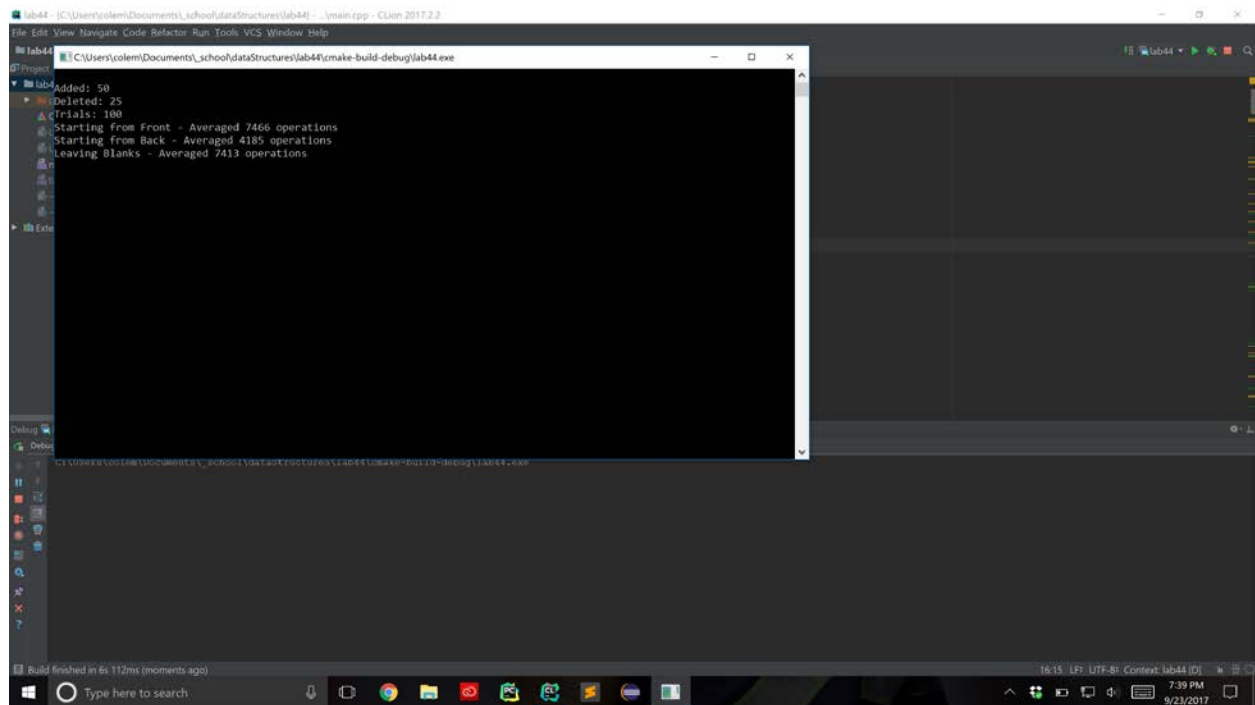
## Task 4:

### Add 30, Delete 25



```
lab44: C:\Users\colem\Documents\schooldataStructures\lab44 - main.cpp - Clion 2017.2.2
File Edit View Navigate Code Refactor Run Tools VCS Window Help
lab44 main.cpp
lab44 C:\Users\colem\Documents\schooldataStructures\lab44\cmake-build-debug\lab44.exe
Added: 30
Deleted: 25
Trials: 100
Starting from Front - Averaged 2970 operations
Starting from Back - Averaged 1755 operations
Leaving Blanks - Averaged 3096 operations
Build finished in 25.834ms (4 minutes ago)
10/27/17 7:38 PM 9/23/2017
```

## Add 50, Delete 25

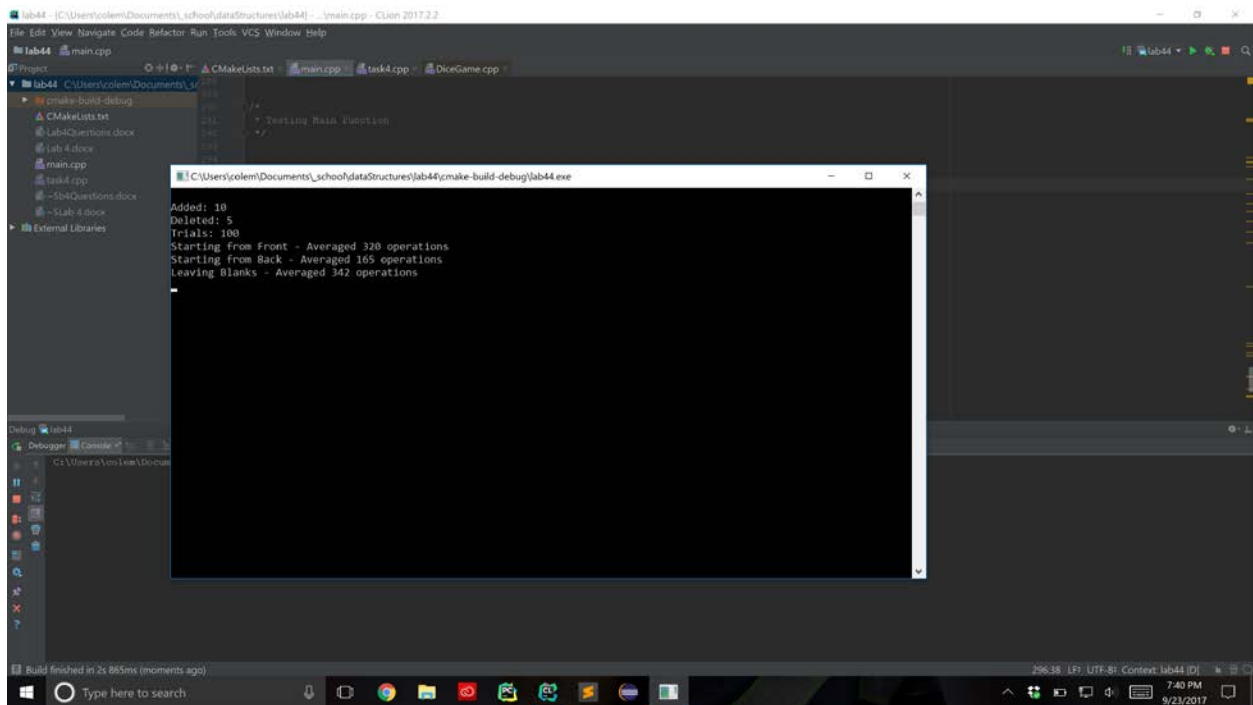


The screenshot shows a C++ IDE with a terminal window displaying the output of a program. The program is titled 'lab44' and is located at 'C:\Users\cole\Documents\school\dataStructures\lab44\cmake-build-debug\lab44.exe'. The output shows the following results:

```
Added: 50
Deleted: 25
Trials: 100
Starting from Front - Averaged 7466 operations
Starting from Back - Averaged 4185 operations
Leaving Blanks - Averaged 7413 operations
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help), a toolbar, and a status bar at the bottom showing the build time (6s 112ms) and the current file (lab44.D).

## Add 10, Delete 5



The screenshot shows a C++ IDE with a project named 'lab44'. The file explorer on the left lists files: 'cmake-build-debug', 'CMakeLists.txt', 'lab4Questions.docx', 'lab4.docx', 'main.cpp', 'task4.cpp', and 'lab4Questions.docx'. The main editor displays 'main.cpp' with a function 'Testing Basic Function'. A console window in the foreground shows the following output:

```
Added: 10
Deleted: 5
Trials: 100
Starting from Front - Averaged 320 operations
Starting from Back - Averaged 165 operations
Leaving Blanks - Averaged 342 operations
```

The bottom status bar indicates 'Build finished in 2s 865ms (moments ago)' and the system clock shows '7:40 PM 9/23/2017'.

For all three trials, starting from the back was the quickest, starting from the front was the second fastest, and leaving blank spaces was a close third place. This was not what I expected. I thought that leaving the blank spaces would be the best method followed by starting from the back. Even though leaving blank spaces is very efficient, calculating and searching for the middle takes a lot of separate operations.