Ryan Hoffman
CS202 Project 8 Documentation
4/15/19
Compile Instructions: ***make -> ./proj8 *there are warnings but the program still works fine****

The purpose of this program was to practice making and using list data structures. The two kinds of list data structures we used were array-based and node-based. Array based lists uses standard memory allocation with the bracket operator to dynamically allocate an array. The project also included classes, pointers, and iostream. We were given a data type class to use as a data type for our lists. Each list performed basically the same functions, except implementing them into the different list types was different. The two list classes performed functions like getting size, inserting before/after, finding specific data types, getting the beginning/end pointers, and deleting/clearing the lists.

In my design I pretty much just did whatever I could to get the program to work. I didn't have enough time to optimize the program and cover all possible error causing conditions. The ArrayList class implementation was fairly easy because we've had experience with basic dynamic memory allocation before and it wasn't too much different, except we had to dynamically allocate arrays instead of single data members. The Node based list was very hard because the project directions were hard to follow and there were so many different conditions that had to be met for each function. For example, what if the list was already empty? What if the list is full? What if they list isn't full? Trying to implement all these conditions caused many errors during compile-time.

I ran into a lot of problems during this project. I got lots of core dump errors because of improper memory allocation or improper ways to access memory. I wasn't familiar with the syntax for linked lists so I was unsure how to use curr to advance to the next item in the list. It would have been helpful if we went over more examples in the lecture. The insertAfter(), insertBefore(), and find() functions gave me the most trouble mostly because they had the most conditions that had to be met. There were a lot of if else statements that I often got lost in. The test driver took me a long time because I had to figure out how to test all the functions I just created and that usually led to more errors I had to fix.

If I had more time I would try to better optimize the program and also add additional testing to the driver (proj8.cpp) file. I didn't have enough time to add testing for the ArrayList functions although I'm pretty sure everything is correctly implemented in that class. I also didn't get to test every single function in the NodeList class because I wasn't sure how to test everything. I think almost everything works in that class as well. It would have been nice if a test driver was provided so we could know what was being tested. I didn't gain much better of an understanding of linked lists due to the length of the program. It would have been better if we skipped the Array-based list class altogether so we could focus on the Node-based lists.