Abraham Meza

Christos Papachristos

CS 202 – 1001

April 17, 2019

Project 8 Documentation

**Description:**

For this project you will create List classes. There will have to be two separate List-based implementations, one Array-based, and the other Node-based.

Continuing through Computer Science II (202), our instructor assigned us a similar project that demonstrates our ability to complete a fragment of code; however, the project was designed to allow student to create List classes in order to test the implementation of array-based and node-based lists through a constructed test driver. By providing students with given header files that illustrate the structure of the ideal class skeletons, our instructor wants students to demonstrate their abilities to execute effective class implementations. . Compared to previous projects, we are instructed to create a sample driver that checks all required specifications. The purpose of this project tis to create lists that can be modified accordingly with predetermined data in the form of nodes and array manipulation (indexing). In doing so, students will be able to modify listed data accordingly to incorporate accurate information and efficient dynamic memory handling. As students continue to polish their knowledge and different implementations with classes, they are also being evaluated to be able to prove the functionality of their program through driver testing.

For my design, I referred to the header files provided in order to structure the programming implementation order and methods. Based on the information provided, students had to implement the given class skeleton into a test driver. Firstly, I began to devise the implementation of the array-based list. Given the many parameters, I was able to structure the program without much difficulty. I began to create the constructors and initialized/assigned the given variables to their corresponding members. As an array environment, I had to manipulate the data through the utilization of *for* loops for access to specific elements. Once I had assigned the data accordingly and established the dynamic memory in each section, I moved onto the individual method and operators. Most of the methods incorporated into the class skeleton revolved around the idea of array access: i.e. front () manipulates pointers to the first element, back () manipulates pointers to the last element, find() manipulates pointers to a specified position in the array, etc. As a result, the implementation of the positioning methods was self-explanatory as it required the concept of array indexing. From thereafter, I began to focus on the array modification methods. These methods incorporated the idea of locating/modifying specified positions in the array by modifying the order. At first, I utilized array manipulation through array indexing to keep track of where to insert/modify corresponding data. However, once I began to experiment with other methods such as erase(), clear(), and empty(), I did not understand how to handle the dynamic memory within the array. Unfortunately, I did not include those methods into the implementation since they could be handled through Nodes. Once I handled the methods, I constructed the overload operators. Luckily, the instructor provided us with two operators: assignment and insertion. The assignment operator closely mimicked the copy constructor, while the insertion operator handled printing the present array data through array indexing (loops).

After concluding and testing the array-based list, I began to construct the node-based programming. Though both lists were to execute similar functionalities, their programming utilized different manipulating. While array-based lists utilized the idea of array indexing, node lists utilize pointer manipulation to access/modify specific data. Through my knowledge of pointer manipulation, I simply referenced my created array-based list and modified it to be compatible with pointer arithmetic. By creating several pointer variables that would track and transfer data, I was able to successfully implement the same methods. Once I modified the similar implementations (constructors, position methods, and operators) and recorded it into the node programming file, I began to work on the modification methods that I could not figure out in the previous file. As a result of using pointer arithmetic (arrow operators, dot operators, etc.), I could simply point to the next object. By accessing the data of the pointed object, I could manipulate it accordingly without much trouble. Through the utilization of new pointer declarations and counting variables, I was able to modify and control the data and dynamic memory. For methods such as erase(), clear(), and empty(), I simply began from the beginning of the Node and traveled until there was nothing next. Through each passed element, the data and memory was deleted and cleared efficiently with previously taught methods.

After finishing the both programing files, I moved onto working on my test driver. Similar to the previous project, I looked onto previous assignments and samples in order to construct the test driver. In doing so, I was able to structure my main file correctly, and tested my implementations with pre-determined data. I implemented additional confirmation statements and organizers in order to clearly read the output and track the execution on the terminal screen, then went back and deleted the placeholders after the debugging process.

Subsequently after a few tests, I was satisfied with my results. Though the instructor did not provide a sample output, he was very clear to what the functionality was supposed to be for each method. By comparing the driver to the previous projects, I was able to map out and correct any possible mistakes. In doing so, I was able to reliably test my code and modify it accordingly. In addition, my confirmation statements allowed me to make sure that every specification was completed and functional.

All in all, the project was very efficient in teaching students about lists and the difference between arrays and nodes when manipulating data, while allowing them to further master class manipulation and implementation through the use of their own test drivers. In doing so, students were able to reflect and strengthen their utilization of pointers and array indexing. Students are now able to see the strengths and weaknesses of each structure. If I were to have more time with the project, I would revisit and attempt to figure out how to manipulate dynamic memory removal in an array environment. In addition, I would attempt to see if there is a more efficient way to organize array data without step-by-step implementations. Overall, the project was a success and was a good example to learn from.