Logan Leavitt

CS 202.1103

Project 8 Documentation

To compile this project, you can type “make”, which produces an executable called *proj8*. Typing “make clean” will remove this executable and the object files. The code I personally wrote for this project can be found in *ArrayList.cpp*, *NodeList.cpp*, and *proj8.cpp* (the test driver). This project aims to provide two different implementations for the same type of list. The first class, ArrayList, implements this functionality using a dynamically allocated array. The first half of output produced by *proj8* is dedicated to testing this class and all its required functionality. The first few lines test the class constructors (and implicitly the << operator in order to print the lists to the terminal). List 1 uses the default constructor, list 2 uses the parameterized constructor, and list 3 uses the copy constructor to copy list 2. I then test the = operator by assigning list 2 to list 1, and print the lists again. To test the [] operator, I access specific elements from list 1 using [] and assign them specific values, as can be seen. The next section tests the front, back and find member functions by checking that they return the correct pointers. I then test the insertAfter and insertBefore by inserting values on the very front, very back, and the middle of the list to cover all test cases. I then test the erase function and erase the target value {3,3}. This next section tests the non-required functionality of the find function. Here I use the extra parameter “start” to recursively search list 1 for the second {5,5}, and demonstrate that it is the second {5,5} by printing out the prev pointer after each call of find. The last section of output for the ArrayList class tests the size, empty, and clear member functions by clearing the contents of list 2 and printing out the data for each list. The second class the test

driver tests is the NodeList class, which takes a different approach to the list by using dynamically allocated nodes which point to each other (creating a linked list) instead of an dynamically allocated array. This class has all the same functionality as ArrayList, so I ran it through the same test cases. The only difference in the test would be the extra functionality of find. While NodeList::find does start searching the list from start if it is given, I could not demonstrate this from the main function because while I could find the first {5,5}, searching from that element would give me the first {5,5} right back, and I had no way to access the next element while m_next was a private member. So, I did not demonstrate this functionality with the NodeList even though I did write the find function this way.