EECS 268

Lab2 write-up

Christopher Watkins

1. What benefit does having a test class provide versus simply checking the list manually with the interactive mode?

• The test class makes it easy to see the functionality of all the class's methods and locate a problem quickly. Without the test class, much time would be spent devising individual ad hoc strategies to check whether a certain method was performing correctly, which is inefficient.

2. Create tests to verify that your list is preserving the order of values.

1. Which LinkedList methods will be involved in this test?

i. I used addFront(), addBack(), insert(), getLength() and the constructor/destructor.

2. How will you go about verifying the order?

i. I commented my code as I was doing the test, but to recap: I created and populated a list with integer nodes storing values 1, 2, and 3 (in that order). I then inserted a 0 value in position 2 of the list. Hence, I knew if my function (inster()) worked properly and maintained the order, I would print the list at "1 0 2 3". To check this I simply put these values into an array (key) and then read in my lists values into an array (listArr) using a For loop. I then compared my listArr to the key array to see if they matched.

3. Describe (do not implement) how you would verify that your Node class can house any integer.

1. I would create three nodes, storing values of 1, 0, and -1. I would then create a for loop for the first node and change the m_value of the node to 1+m_value for each increase of my "I" parameter (i++). Hence, the loop would increase the Node's value incrementally by 1 integer until the specified maximum integer I defined in the For loop's condition (let's say 1000). I would then print this m_value. If it reads 1000 I can likely assume the Node could hold any integer higher, up to infinity. I would then do the same for the Node holding value -1, but of course in the negative direction approaching negative infinity.

2. This would give me strong likelihood to believe my node class can house any integer. The upper bound could be higher than 1000 but the practicality of increasing the bound would yield diminishing returns, as you start housing very large integers, at some point on must simply assume that the class with hold any integer larger. Hence, we cannot have 100% confidence, but we could have a strong indication from this test.

4. Discuss the ratio of code you're trying to test to test code. For example, how much code was isEmpty() to implement versus how much code you needed to test isEmpty()?

• The ratio of code to test code is high. The code needed to test a function like isEmpty() is much shorter than the actual code required to write the method. The add/remove Front & Back methods, for instance, took quite a bit of code compared to the brief methods used to test their efficacy. The only exception was test 16, which required more code than the other tests, but this is understandable as it checked a more complex outcome of a function.