CS II: Data Structures
Lab 3: Testing and fixing a linked list

In the prelab, you ran the SLinkedList.java through some tests and saw that it was failing most of them. In this lab, you will fix the bugs--one by one.

1. testIsEmpty fails and provides a line number for the assertion that failed. You can find that line number from JUnit's output.

   ![JUnit output](image)

   (If you can't see the test results you might need to click the Test Results button at the bottom of the console)

   We know from looking at the test code that `isEmpty` incorrectly returns true when the list is not empty. Finish the SLinkedList.isEmpty method so that testIsEmpty passes.

2. testGet is failing on its first assertion statement. There is a bug in the SLinkedList.get method that makes it return data from the wrong ListNode. Fix it so that testGet passes.  

3. Finally, you are going to implement the contains method. This method returns true only if the given element is found in the list. Once you get testContains to pass, that is decent evidence your implementation is correct.

   If you are not sure how to start, you can try steps in this order.
   a. Draw an abstract list (i.e., [200,300]) and follow the operations in testContains to make sure you could explain to someone what contains is expected to do.
   b. Write an algorithm for contains in words.
   c. Show the steps of the algorithm using boxes-and-arrows.
   d. Write Java code for contains so it only works on lists of length 0 and 1.
   e. Write your first try at a runnable solution. Debug, debug, debug!

1. Debugging strategies
   - Add print statements to see what the values of the variables are at different times.
   - Draw a boxes-and-arrows diagram using the failing test case as your example input.