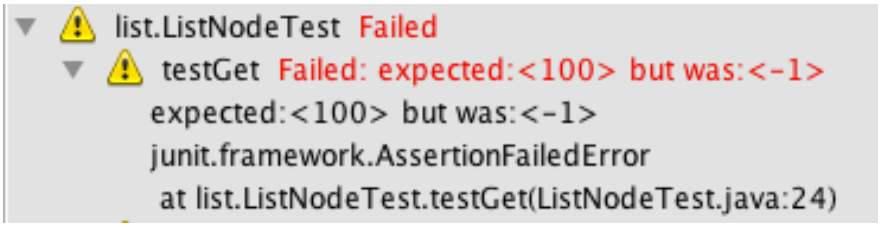
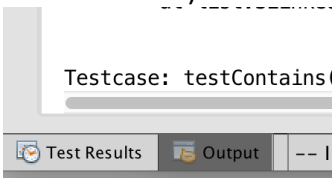
CS II: Data Structures
Lab 3: Testing and fixing a linked list

In the prelab, you ran `ListNode.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. `testGet` fails and provides a line number for the assertion that failed. You can find that line number from JUnit's output.

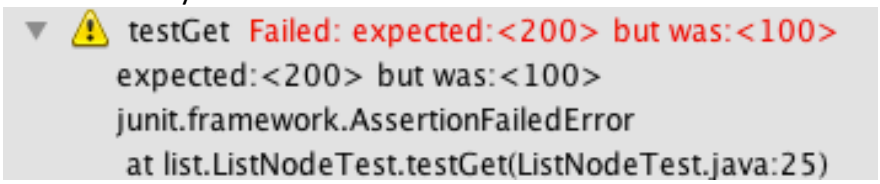


(If you can't see the test results you might need to click the Test Results button at the bottom of the console)



So, get() incorrectly returns -1 when the list contains [100,200]. **Write *just enough code*** in the ListNode.get method so that the first assertion passes. That is, implement get() so that it always returns the first element. We'll worry about arguments larger than 0 later.

2. Once you complete that step, you should see a different assertion error for testGet when you re-run the tests.



That message makes sense, since our current ListNode.get implementation always returns the first element, it will give the wrong answer when asked for index 1. **Finish the code for ListNode.get**. You are done when testGet passes.

If you are not sure how to start, you can try steps in this order.

a. Take a look at ListNode.add and ListNode.length for examples of how to traverse the linked list

b. Draw an abstract list (i.e., [200,300]) and follow the operations in `testGet` to make sure you could explain to someone what `get` is expected to do.
c. Write an algorithm for `get` in words.
d. Show the steps of the algorithm using boxes-and-arrows.
e. Write your first try at a runnable solution. Debug, debug, debug![1]

3. Complete the implementation of ListNode.contains. Do not continue until testContains passes.

   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 1.
   e. Write your first try at a runnable solution. Debug, debug, debug![1]

4. Complete the implementation of ListNode.removeLast. When all the tests pass, that is decent evidence that you have a correct implementation.

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.