Operating Systems
Spring Semester 2020

Greg Witt
greg.witt625@gmail.com

Lab-09:

Test Cases Outline:

%ll –c apple //this program creates a list apple and does work

%ll –p //this program prints a list and does work

%ll apple –a zebra //adds zebra to the list and does work

%ll apple –d zebra //deletes zebra from the list and does work

Methods for Testing:

-a ADD:

Initial Setup:

-Check to determine there are lists for testing with $**ll -c**. Use -c to create an additional list separate from the previous tests for control testing. $**ll -c test** Confirm that the list has been created by checking stack or Heap for the list by running$ **ll -p.**

Testing Development:

Confirm that test was created and determine which order the test list will add another node to the list. **Add** can either develop the pointer to the HEAD or TAIL of the Linked-list. It is important to determine this in order to properly implement adding elements to a linked list. As pointers and dereferences could be in question and if they are improperly referenced there could be memory leak issues in the system.

>$ll test -a subject

>$ll -p

Assertion Analysis:

If the test case was correctly executed it would be imported to print out the list in its entirety and determine which location the items to the test list were added. In this test the subject list was added to the list. It would be important to then test the location of the HEAD and Tail of both the subject list item and the test item nodes in the linked list

HEAD Case:

If the *subjec*t is pointing to head then the *subject* would be a clear indicator that it is the method's implementation to add the list as a parameter and that the location of the list

would be to set the pointer of its incoming node to HEAD and then move the references of the other nodes to the TAIL of the previous entry.

TAIL Case:

If the *subjec*t is pointing to *tail* then the *subject* would be a clear indicator that it is the method's implementation to add the list as a parameter and that the location of the list would be to set the pointer of its incoming node to TAIL of the previous existing node and then move the references of the other nodes to the NULL of the new incoming entry.

-d DELETE:

Initial Setup:

-Inital configuration for this method is crucial as testing the development of **ADD,** should assist in determining the outcome of the deletion method. As well as prove that there is another NODE element to point to in the linked-list. Check to determine there are lists for testing with $**ll -c**. Use -c to create an additional list separate from the previous tests for control testing. $**ll -c test** Confirm that the list has been created by checking stack or Heap for the list by running$ **ll -p.** To properly configure this Unit test there must be a clear indication for the testing parameters and the elements in the list should be added and the initial testing of add should of revealed the nature of the method, this will allow for quicker results cases and assertions. Adding elements to the list will be the first step necessary for testing the method.

Testing Development:

Confirm that test was created and determine which order the test list(s) for deletion **Delete** can either remove the pointer of the HEAD element or TAIL of the Linked-list. It

is important to determine this in order to properly implement deleting the elements from the linked list

Optional Configurations:

$>ll test -a subject

$>ll -p

$>ll test -d subject

$>ll -p

Additional Options:

$>ll test -a subject

$>ll test -a subject2

$>ll -p

$>ll test -d subject2

$>ll test -d subject

$>ll -p

Assertion Analysis:

With the testing implemented the assertions for the test will either result in a deletion in which the head element is somehow removed from the linked list and the second element of the list will be pointing to HEAD and the TAIL pointing to NULL or the second head will be the one clipped from the list and the Testing Element remaining,

In the case of the second Assertion tests there will most likely be a case where the test node remains untouched. However there could be instances where the deletion executed affects the head and improperly removes the remaining subjects without a First in First out or a Last In First Out (LIFO). This behavior after printing the elements with **$>ll -p** should yield more clues as to the behavior of this method on the linked lists