CS 260
Programming Assignment 3

This lab is worth a total of 100 points; 10 points are the self-evaluation, 50 points for the basic lab, 30 points for the advanced lab, and 10 points for a solution to the thinking problem.

Base Lab Specification

Create a test processing class named TextClass. This should be implemented by using a double-linked list of chars.

This should be a class that allows you to input a single character at a time, list the resulting characters, find a character, and delete the first example of a character. You can base your code off the pseudo code in the double linked list document in Moodle.

This lab should follow the course coding requirements and be split into multiple files as per your chosen language. There is a driver provided for this lab, you should test each part of your code as it is developed. The exceptions raised or thrown when trying to read from an empty list should be ***InvalidOperationException*** for C#, ***out_of_range*** for C++ and ***ValueError*** for Python.

The base methods required are:

- addHead(value) – adds value to the head of the list
- addTail(value) – adds value to the tail of the list
- getHead() – return the value from the head of the list (throw an exception if list is empty)
- getTail() – return the value from the tail of the list (throw and exception if the list is empty)
- removeHead() – removes the value at the head of the list
- removeTail() – removes the value at the tail of the list
- find(value) – returns true if value is present in the list, returns false if not
- findRemove(value) – returns true and removes the value if present, returns false if not
- displayList() – return a string containing the contents of the list from head to tail, should insert a space between each value.

Advanced Lab Specification

For the advanced lab, start with your base lab class and add the following functionality.

- append(otherList) – append the contents of otherList to the tail of this list. Note that otherList is an object, so should be passed by const reference (except for Python which does not know about references).
- findNext(value) – like find, but if findNext is called for the same value after a success, it should find the next instance of that value, wrapping if necessary.

  This function saves a reference (pointer) to the link that was found. If no such link is found, it should set the reference (pointer) to an empty state (nullptr, None, null).

- removeLast() – removes the link that was saved by the last call to findNext. If the saved value is empty (nullptr, None, null) then do nothing. After removing the link, it should set the referent(pointer) to its empty state.
- insertLast(value)—insert value before the link that was saved by the most recent call to findNext. If the saved value is missing (nullptr, None, null) then do nothing.

Thinking problem

Given the two strings "This is a cat" and "This is a dog", write the necessary function calls to create a single string "This is a cat and that is a dog". Your solution needs to use append as one of the steps. The dog string must not be modified as part of this solution.

You need to display both strings after you are done to show that you did not modify the dog string during the solution.

Examples of findNext

Consider a TextClass containing the following list of letters:

Head-> A -> B -> A -> C -> D -> nullptr/NULL/none

and the following list of calls to findNext()

findNext(A) should find the first A

findNext(B) should find the B

findNext(E) should return false

findNext(A) should find the first A

findNext(C) should find the C

findNext(C) should wrap and find the C

findNext(A) should find the first A

findNext(A) should find the second A

findNext(A) should wrap and find the first A

In other words, after a successful find, when called for the same value, look for the next instance of that value, wrapping if necessary.

After a successful find, when called for a different value, start at the beginning of the list, and do not wrap if the value is not found.