CS272 Lab Assignment #5.

Due: Thursday, 3/8 by 11:30pm.

In this assignment you will work with doubly linked lists. An implementation of a doubly linked list is provided:
DNode class represents a node of a doubly linked list;
DList class represents a doubly linked list with dummy head and tail nodes.

Part 1 (10 points). Plan to Graduation Assignment. If you are not a CS undergraduate major you need to do only item 1) on the assignment. If you are a CS undergraduate major you will need Plan to Graduation Spreadsheet.

Part 2 (90 points).

0: Read Chapter 4 from the textbook. (Doubly linked lists and dummy nodes are explained in Section 4.6.)

1: Use Eclipse. Create a new project with the name, say, lab5. Import the above files DNode.java and DList.java into the project. You will be just using these two classes in your project; their code should not be modified. Create a new class in the project with the name DLinkTester. Write the following methods in the class DLinkTester:

1. Write a Java method called *swap* to swap two nodes x and y (and not just their contents) in a doubly linked list L given references only to x and y. You need to change links in nodes x, y, and their neighbors; data elements in x and y do not change. Your method should work correctly when
    - nodes x and y are not next to each other in the list,
    - node x is the previous node of node y in the list,
    - node x is the next node after node y in the list,
    - x and y refer to the same node (in this case they should not be changed).

    Method *swap* should have two parameters of type DNode (references to nodes x and y). It should swap x and y and return true if none of x and y are null as well as none of their neighbors (previous of x, next of x, previous of y, next of y) are null. Otherwise (if any of these nodes are null), the method should not change anything and return false (the nodes are not swapped).

2. Write a Java method called *concat* for concatenating two doubly linked lists L and M, with head and tail dummy nodes, into a new single doubly linked list N. (In a concatenation the elements of the second list are appended to the end of those of the first list.) Lists L and M should remain unchanged. The method should work for all lists, including empty lists. The method *concat* should have two parameters of type DList (references to lists L and M). It should return the reference (of type DList) to the created list N.

3. Write a Java method called *reverse* to reverse the order of elements in a given doubly linked list. (Do not create a new list, reverse the elements in the existing list. Do not allocate any

new nodes.) The method should work for all lists, including empty lists. If the list is empty the method should so nothing. The method should have one parameter of type DList (reference to the list which elements are to be reversed). The method should not return anything (return type is void).

4. Write a Java method *merge* to create a new doubly linked list N that contains elements alternately from two given doubly linked lists L and M. If you run out of elements in one of the lists, then append the remaining elements of the other list to N. Lists L and M should remain unchanged. The method should work for all lists, including empty lists. If any one of the two lists is empty, then the method should return a copy of the second list.
   For example, if list L contains 4 elements "one", "two", "three", "four", and list M contains 6 elements "a", "b", "c", "d", "e", "f", then list N should contain the following elements: "one", "a", "two", "b", "three", "c" "four", "d", "e", "f".

   The method *merge* should have two parameters of type DList (references to lists L and M). It should return a reference (of type DList) to the created list N.

5. The *main* method of DLinkTester class should test all your methods. Make sure to fully exercise the code of the methods (see p. 28 of the textbook). (Note: DList class has toString method that you can use to output list contents.)

Note: Specifications for all the methods which you implement should be included as comments in your code. Also, please use inline comments, meaningful variable names, indentation, formatting and whitespace throughout your program to improve its readability.

What to submit:

- Submit your Plan to Graduation assignment on paper.
- Submit your source code (DLinkTester.java file) electronically on Canvas. (Files DNode.java and DList.java should not be changed; therefore, you do not need to submit them).