Lab 3 February

Goal: This lab will demonstrate how a sentinel node can simplify a doubly-linked list implementation.

Getting Started

Please make sure you have a partner for this lab.

The files in the lab4 directory contain classes for two different types of doubly-linked list. The DList1 class does not use a sentinel, whereas the DList2 class does. The DList1 class is not circularly linked, but the DList2 class is (through the sentinel). Compile DList1.java and DList2.java (using "javac -g DList1.java DList2.java".)

Your task is to implement two insertFront() and two removeFront() methods--one of each for each list class. insertFront() and removeFront() insert or remove an item at the beginning of a list. Make sure your implementations work for empty lists, one-node lists, and larger lists.

The main() methods of DList1 and DList2 include test code, which you can run with "java DList1" and "java DList2".

Part I: insertFront in DList1 (1 point)

Write a method called DList1.insertFront() that inserts an int at the front of "this" DList1.

Part II: removeFront in DList1 (1 point)

Write a method called DList1.removeFront() that removes the first item (and node) from "this" DList1.

Part III: insertFront in DList2 (1 point)

Write a method called DList2.insertFront() that inserts an int at the front of "this" DList2. Your code should NOT use any "if" statements or conditionals.

Part IV: removeFront in DList2 (2 points)

Write a method called DList2.removeFront() that removes the first item (and non-sentinel node) from "this" DList2. Your code should not require separate branches for the one-node case and the more-than-one-node case. (You will need one "if", to handle the zero-node case.)

1 point: DList1.insertFront().
1 point: DList1.removeFront().
1 point: DList2.insertFront().
1 point: DList2.removeFront().