CS 0445 Spring 2026
Recitation Exercise 4

Introduction:

In Assignment 2 you will be implementing a StringBuilder utilizing a circular, doubly-linked list. In lecture we looked at an implementation of the author's ListInterface<T> utilizing a null-terminated, singly-linked list (LList<T>). To help you get more familiar with circular, doubly-linked lists, in this exercise you will modify the author's LList<T> class to make it a circular, doubly-linked list. The functionality of the class will not change – what will change is simply the form of linked-list utilized in the implementation of the interface.

The new class will be called DLList<T>, and the header will be as follows:

```
public class DLList<T> implements ListInterface<T>
```

To get you started I have already partially implemented this class in file DLList.java (located in Files/Recitation/Recitation4/). Carefully read over this file, including my comments. Note in particular the changed inner Node class and the places in the code where the circular doubly-linked list implementation differs from that of the null-terminated, singly-linked list. Look at the various cases in the code I have written – in particular note the special cases in the two add() methods.

Some specific things to consider include:

- The Node class now has an extra reference, prev, which will connect a Node to is predecessor in the list.
- When nodes are added to or removed from the list, both next and prev references of affected nodes must be updated.
- Other than the special case of an empty list, no Node references in the DLList<T> class will be null. The next value of the last Node will be a reference to the first Node in the list and the prev value of the first Node will be a reference to the last Node in the list. If the list has only a single Node in it, that Node will "point to itself" in both directions.
- For traversals of the list we can no longer have a loop that stops when null is reached. There are several ways of handling traversals - one way is to utilize the numberOfEntries value and a counter to determine when the traversal is complete and another is to check to see when a reference traversing the list has circled back to the front of the list.

The DLList<T> class provided to you is missing only 2 methods – which you must complete in this exercise. Those methods are:

```
public T remove(int givenPosition)

public boolean contains(T anEntry)
```

See the detailed requirements for these methods in ListInterface.java. We have discussed contains() in our linked bag implementation and it is the same in the LList<T> implementation. We also recently discussed the remove() method in lecture. **Consider how they will be different in the DLList<T> class.**

For help with this exercise look at the code I have written for some of the other methods in the DLList<T> class. Also look at the code for remove() and contains() in the LList<T> class. Think about what will be the same / similar in the DLList<T> class and what will be different.

Once you have completed your modified DLList<T> class, test it with the main program CS445Rec4.java. Your output should match that shown in file Rec4Out.txt. To run this program you will need the following files:

ListInterface.java (provided)
DLList.java (provided, but incomplete – you must complete the methods specified)
CS445Rec4.java (provided)

All of these files are on Canvas at Files/Recitation/Recitation4/.

As usual, if you are able, please volunteer to demonstrate and explain your solution to the rest of your class at the end of recitation. If you are unable to complete the exercise, you may view my solution after it is posted onto Canvas.