CPS 151/Spring 2017 **Homework Assignment 3** **April 11, 2017**
Due: April 21, 11 pm **Cut-off: April 24, 11 pm**

Numbered List (no duplicates) Implementation using linked list technique

In this assignment you will re-implement a Numbered List class using Linked Lists. The assignment 2 document describes the functionalities of a Numbered List interface, please refer to that document. The client will continue to view the items in the list to have a sequence number, but in the linked list the sequence numbers do not need to be stored with the items.

No duplicates
The class should not allow duplicate values to be stored.

Add and remove by value
Like Java's ArrayList class, this class will allow adding an item without specifying a position, in which case the item gets added as the last item. A remove operation by specifying a value will remove it from whatever position it is in. Note that the item, if present, must be unique.

Size of the list
This class supports the notion of **size**. The **size** is the number of values currently stored in the list. The **size** is incremented for every successful **add** operation and is decremented for every successful **remove** operation. These are the only operations that modify the size.

Error detection
Operations with invalid position numbers should cause an error message and are to be rejected. If there are **n** items in the list, then valid add positions are in the range [1, **n** + 1] and valid remove positions are in the range [1, **n**] (Why?). Remember that class clients view list positions to start at 1.

A **find(item)** operation should be supported which returns the (logical) position number of an item if present, or returns 0 if the item is not present (can be done by a linear/sequential search). This is the complete list of operations for the class (defined as a Java Interface)

```
public interface NumberedListNoDup<T> {
    boolean add(T item);
    boolean add(int position, T item);
    boolean remove(T item);
    boolean remove(int position);
    T get(int position); // return null if invalid position
    boolean set(int position, T item);
    int size();
    void clear(); // empties out the list
    int find(T item);   // return 0 if item not in list
    String toString();
} // end interface
```

The client application
The class client would present the user with a menu to exercise all the operations. You can design this similar to the Bag implementations.

Error detection

Many of the NumberedList methods may fail because the method is called with an invalid position (also attempt to add a duplicate item, remove an item not in list). The boolean typed methods should return false when they fail and should return true when they succeed.

Note: `get()` should return null when it is given an invalid position
`find()` should return 0 when it is asked to find the position of an item not present in the list.

File organization

The client application should be in a file named userid_Asgn03.java. The NumberedList class should be in a file named NumberedListNoDup.java. Both files should include comments to identify you.

Some partial code files are to be posted on Isidore.

The submission template for this assignment has separate pages for the client code and the class/interface code. There is another page for pasting the screen shot(s) showing program output.