CPS 151/Spring 2017 **Homework Assignment 2** **March 28, 2017**
Due: April 6, 2017

Numbered List (no duplicates) Implementation using an array for storage

In this assignment you will implement a Numbered List class. A Numbered List has zero or more items of a specific type (This is the type parameter for a generic class). Each item has a sequence number starting with 1 and going up to the size of the list. Items can be added to the list, or deleted from the list **by specifying a position**. This is very similar to ArrayList class of Java library, except that ArrayList positions start from 0. Consider the following numbered list (used to store a grocery list):

1. Bread
2. Milk
3. Cereal
4. Juice

After the operation **add(3, "Butter")** the list becomes

1. Bread
2. Milk
3. Butter
4. Cereal
5. Juice

Notice how two items now have their position numbers incremented. Now if we do the operation **remove(2)**, the list will become

1. Bread
2. Butter
3. Cereal
4. Juice

Once again some items change their position numbers.

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.

How to manage the data storage
The storage can be managed in a manner similar to the ArrayBag class.

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_Asgn02.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.