**CSCI 2010U - Data Structures - Fall 2025**

**Lab 2 - Building and Using Lists**

**Learning Objectives:**

In this lab, students will:

* develop their own custom list based on a doubly-linked list;
* add various methods to their list to access and manipulate elements;
* generalize their list to work on any object or data type; and
* demonstrate the use of their list by creating and manipulating lists at run time.

**Academic Integrity Reminders:**

* For this lab, you **must** work with a partner, however your group should not consult with other students or any other outside persons except for general guidance.
* The use of generative AI tools to produce any portion of your solution is not permitted.
* You may review tutorials and other sources online for ideas and syntax assistance, but you cannot copy any code from another source. As such, it is recommended that you avoid resources such as Stack Overflow and stick to tutorials and reference sites.
* All code must be written by you on your own computer.
* When in doubt, ask your lab instructor - it’s better to play it safe.

**Part 1 - Create Your Version of A Doubly-Linked List**

1. While Java has its own LinkedList class, for this lab you will create your own class to implement a doubly linked list (you can call it whatever you want), as well as a class for the elements/nodes in your list.
2. To begin, your list should work with **int** values - we’ll update it later.
3. For this step you are simply creating your Classes, their attributes, getters and setters, and constructors.

**Part 2 - Provide A Core Set of Methods to Your List**

1. Now that you have your List created, you will need to add the following required methods to provide functionality:
2. **getLength()** - returns the number of items in the list
3. **addToFront(int value)** - a method that adds an element to the beginning of the list
4. **removeFromFront()** - a method that removes the first element of the list
5. **addToEnd(int value)** - a method that adds an element to the the end of the list
6. **removeFromEnd()** - a method that removes an element from the end of the list
7. **toString()** - returns a **String** representation of the list in an easy-to-read manner (e.g. **[-1, 0, 1, 2]**)
8. You must also add one more method of your choosing that expands the functionality of your list - consider the standard operations you would expect from a list and add a method to implement one.
9. Make sure that you account for all the special edge cases (e.g. adding to empty list, adding to singleton list, removing from singleton list, removing from empty list, etc.).
10. Create a **ListTest**Class that has a **main**method, and use the main method to demonstrate that your List works as expected by creating, editing, and printing various lists.

**STOP:** Show the output from your console to the TA to receive your grade for this portion of the lab. They will need to see your code (properly commented) and then the output on the console. They may ask you to do additional steps to properly demonstrate your code. **This part of the lab is worth 2 points.**

**Part 3 - Allow Your List to Generalize to Any Object**

1. Your current implementation works only for the **int**data type, however, a list should be more robust than that, so it should be able to represent a list of any type (any Java Object in our case). Since each of the primitive data types have container classes, it will still work for regular int values, but also for any custom or default objects.
2. Update your list item class to hold **Object**data rather than **int**along with any appropriate methods and constructors.
3. Update your list class’s methods so that they accept general Objects as parameters rather than simple int values.

**Part 4 - Demonstrate The Various Applications of Your List**

1. You must demonstrate that the changes you made to your list did not remove any of its functionality - to do this, you should**run THE SAME main method as Part 1**, with no changes, to ensure it still works with simple int values.
2. You will then need to ensure that your **Burrito**class from last week’s lab is in the same folder as your current source code so that you can leverage this object.
3. You should then create a new **BurritoList** class and **main** method to test your list with **Burrito** Objects. Create several Burritos and add them to a list, ensure that it prints properly and all the functions work for a Burrito.

**STOP:** Show the output from your console to the TA to receive your grade for this portion of the lab. They will need to see your updated code (properly commented) and then the output on the console for both main methods. They may ask you to do additional steps to properly demonstrate your code. **This part of the lab is worth 1 point.**