CS 313 Spring 2020 Midterm Assignment

The following 4 coding problems are part of the midterm 1 assessment and in total are worth 20% of the course grade (the other 20% of midterm 1 will come from the exam). The following files related to these problems are provided:

- ArrayList.java
- CircularArrayList.java
- CircularList.java
- DoublyLinkedList.java
- LinkedIterator.java
- LinkedStack.java
- List.java
- Node.java
- Sortable.java
- Stack.java

To answer each of the questions you will need to modify some of the provided files. You should only modify the files as instructed by the problems, and you should not utilize any additional classes that are not already directly imported in the provided files. For each of the questions you may define additional helper methods if needed, but should not redefine existing methods (unless specified). You may also modify the **package** statements if it is convenient.

Your submission should be a compressed folder (preferably .zip) containing all of the above files with the required modifications made. Late submissions will not be accepted; for this reason it is suggested that you do not wait until the last minute to submit.

1. Complete **ArrayList** (5 points)
    a. The provided **List** interface has been modified to include the following methods: **void addRange(int index, E[] values)**, **Object[] removeRange(int fromIndex, int toIndex)**. Modify the provided **ArrayList** class so that these two methods are implemented in O(n) time.
2. Complete **CircularArrayList** (5 points)
    a. An interface called **CircularList** is provided that consists of all of the methods from the **CircularlyLinkedList** implementation from lecture (in other words, the **CircularlyLinkedList** class from lecture already implements the **CircularList** interface.) Add the missing methods from the **CircularList** interface to the resizing array-based **CircularArrayList** that has been provided. You should also complete the **private void resize(int newCapacity)** method so the structure can dynamically resize. Other than **resize(...)**, which must take O(n) time, each of the **CircularList** methods should be implemented so that they run in O(1) time. (To clarify, methods which add data to the list may need to call the **resize()** method - do not consider **resize()** calls in your runtime computation). It is suggested, but

not required, that you also implement a **toString()** method for debugging and testing.

3. Complete **DoublyLinkedList**'s missing method from the **Iterable** interface, complete **LinkedIterator**, and complete **LinkedStack** (5 points, +1 extra credit possible)
    a. The provided **DoublyLinkedList** class definition has been modified so that it implements **Iterable<E>**. Modify the **DoublyLinkedList** so that it implements the missing method required by the **Iterable<E>** interface; this method should return an instance of **LinkedIterator**. The **LinkedIterator** class is provided but incomplete; modify **LinkedIterator** so that it correctly implements the **Iterator<E>** interface. Extra Credit: implement the optional **Iterator<E>** method **void remove()** in the **LinkedIterator** class (as described in the **java.util.Iterator<E>** documentation). [Note that the **Node** class used by **DoublyLinkedList** has been provided in a separate file.]
    b. The provided **Stack** interface has been modified to include the method **int search(Object o)**, as described in the **java.util.Stack<E>** documentation. Utilize the completed **LinkedIterator** to implement this missing method in the provided **LinkedStack** class (the **LinkedStack** class is a **DoublyLinkedList**-based implementation of **Stack**.)
4. Complete **ArrayList**'s and **DoublyLinkedList**'s missing method from the **Sortable** interface (5 points)
    a. The provided **ArrayList** class definition has been modified so that it implements the provided **Sortable<E>** interface. Modify the **ArrayList** so that it implements the missing method required by **Sortable**; this method should sort the elements stored in the list using merge sort.
    b. The provided **DoublyLinkedList** class definition has been modified so that it implements the provided **Sortable<E>** interface. Modify the **DoublyLinkedList** so that it implements the missing method required by **Sortable**; this method should sort the elements stored in the list using insertion sort. [Note that the **Node** class used by **DoublyLinkedList** has been provided in a separate file.]