Data Structures in Java: Programming Assignment 3
ArrayLists and LinkedLists

1. Objective

Your goal is to implement additional methods found in the MyList interface. These methods will be implemented in both MyArrayList and MyLinkedList classes. In doing so, you will gain a better understanding of the similarities and differences between the two data structures.

2. Problem

Download the following files from CourseWorks:

- `MyArrayList.java`
- `MyLinkedList.java`
- `MyList.java` *(found in the assignment description itself)*

There are 6 new methods in the `MyList` interface. You need to implement them in `MyArrayList.java` and `MyLinkedList.java`.

```
/**
 * Returns a string representation of the list. The string will begin with
 * a '[' and end with a ']'. Inside the square brackets will be a comma-
 * separated list of values, such as [Brian, Susan, Jamie].
 * @return a string representation of the list.
 */
@Override
String toString();

/**
 * Inserts the specified element at the specified position in this list.
 * Shifts the element currently at that position (if any) and any subsequent
 * elements to the right (adds one to their indices).
 * @param index    index at which the specified element is to be inserted
 * @param element  element to be inserted
 * @throws IndexOutOfBoundsException if the index is out of range
 *         (index < 0 || index > size())
 * The exception message must be:
 * "Index: " + index + ", list size: " + size
 */
void add(int index, E element);

/**
 * Removes the element at the specified position in this list.
 * @param index  the index of the element to be removed
 * @return the element that was removed from the list
 * @throws IndexOutOfBoundsException if the index is out of range
 *         (index < 0 || index >= size())
 * The exception message must be:
 * "Index: " + index + ", list size: " + size
 */
E remove(int index);

/**
 * Returns the index of the first occurrence of the specified element in
 * this list, or -1 if this list does not contain the element. More
 * formally, returns the lowest index i such that Objects.equals(o, get(i)),
 * or -1 if there is no such index.
 * @param element element to search for
 * @return the index of the first occurrence of the specified element in
```

```
     * this list, or -1 if this list does not contain the element
     */
    int indexOf(E element);

    /**
     * Returns an array of indexes of each occurrence of the specified element
     * in this list, in ascending order. If the specified element is not found,
     * a non-null empty array (not null) is returned.
     * @param element element to search for
     * @return an array of each occurrence of the specified element in this
     * list
     */
    int[] indexesOf(E element);

    /**
     * Reverses the data in the list.
     * For MyArrayList, the data inside the underlying array is moved. For
     * MyLinkedList, the tail must become the head, and all the pointers are
     * reversed. Both implementations must run in Theta(n).
     */
    void reverse();
```

3. Tips

You must test your work thoroughly. Junit tests have been provided for you, but they are not comprehensive. You are responsible for ensuring your code works for edge cases. If your code does not compile, you will earn a 0. You will earn points for test cases that pass and follow the specifications described in the Javadoc comments.

4. Submission

Create a zip file called hw3.zip containing:

- `MyArrayList.java`
- `MyLinkedList.java`
- `MyList.java`

Do not put your code in a Java package or submit extraneous files or folders. Upload your submission to Canvas. Only the final submission will be graded.