Lab 6 -- List Utilities

The goal of this lab is implement additional functionality for abstract data types and using higher order functions.

List Utilities

In order to complete an implementation of any collection, a class library will usually create a utility class consisting exclusively of static methods that operate on that collection. This utility class could include the following:

1. ListAdt<T> toList(T[] elements) that creates a new instance of the ListAdt that contains all of the specified elements in the order they appeared in elements. This method should throw a IllegalArgumentException if elements contains one or more null values.
2. void addAll(ListAdt<T> list, T... elements) that adds the specified elements to the end of the specified list. Elements should be added in the same order they appear in elements. This method should throw a IllegalArgumentException if elements contains one more more null values.
3. int frequency(ListAdt<T> list, T element) that returns the number of elements in the specified list equal to the specified element. More formally, it should return the number of elements in the list such that (o == null) ? e == null : o.equals(e))
4. boolean disjoint(ListAdt<?> one, ListAdt<?> two that returns true if the two lists have no elements in common. This method should throw a IllegalArgumentException if either list is null or if either list contains a null element.
5. boolean equals(ListAdt<?> one, ListAdt<?> two) that returns true if the two lists are equal. Two lists are equal if they have the same elements in the same order. If either list is null, or if either list contains a null element, this method should throw a IllegalArgumentException.

*Hint:* If you have never see the <?> syntax before, this is the [generic wildcard that you can read more about on the Oracle's website (Links to an external site.)](https://docs.oracle.com/javase/tutorial/extra/generics/wildcards.html).

What to do

**Package:** listadt

Start by downloading [this ListAdtUtilities class](https://northeastern.instructure.com/courses/63372/files/8190273?wrap=1)[download](https://northeastern.instructure.com/courses/63372/files/8190273/download?download_frd=1)that contains a stub for each of method that you need to implement. Using the ListAdt implementation of our sequential structure, implement each of the above methods in a ListAdtUtilities class. If your current implementation does not have a filter, map, and fold higher order function implementations, add them to your ListAdt interface and implementation:

ListAdt<T> filter(Predicate predicate)

<R> ListAdt<R> map(Function<T, R> converter)

T fold(T identity, BinaryOperator<T> accumulator)

. For each method listed above, consider whether you can leverage the higher order functions that you just added to implement the functionality.

If you cannot find a way to leverage the higher order functions, then add the implementation to the list by adding the necessary functionality into the ListAdt node classes. To be able to call these methods from inside the ListAdtUtilities class, you will need to add getHead and setHead methods with package level access in the ListAdt implementation (but not in the interface).

We recommend that you select an operation and implement it end-to-end before starting the next operation:

1. Add it to the interface
2. Add an empty implementation
3. Write test cases
4. Complete the implementation until all of the tests cases pass