CS284: Exercise Booklet 3 - Lists

1 Single-Linked Lists

Implement the following public methods in the class `SingleLL<E>`. You may use helper methods if you feel the need to.

Exercise 1

`boolean isSingleton()` that returns a boolean indicating whether the recipient list is a singleton list or not.

Exercise 2

`boolean allNonNull()` that returns a boolean indicating whether all the elements in the list are non-null.

Exercise 3

`boolean mem(E el)` that returns a boolean indicating whether `el` belongs to the list.

Exercise 4

`boolean nonDuplicates()` that returns a boolean indicating whether the list has duplicates or not.

Exercise 5

`SingleLL<E> clone()` that creates a copy of the list.

Exercise 6

`SingleLL<E> append(SingleLL<E> l2)` that appends the two lists. Eg. Given `[1,2,3]` and `[4,5]` returns `[1,2,3,4,5]`.

Exercise 7

`void reverse()` that reverses the list. Provide two solutions. The first one returns a new list, the second reverses the recipient list (i.e. it does not create a copy of its elements). What does the return type `void` imply?

Exercise 8

`SingleLL<Boolean> areNull()` that returns a list of booleans indicating whether each element is null or not. Eg. given the list `[1,2,null]` it should return `[false,false,true]`.

Exercise 9

`SingleLL<E> repeatLN(Integer n)` that, returns a new list in which `n` copies of the original list have been juxtaposed, Eg. Given the list `[1, 2, 3]` and the number 3 it should return `[1, 2, 3, 1, 2, 3, 1, 2, 3]`.

Exercise 10

`SingleLL<E> stutterNL(Integer n)` that repeats each element in the list `n` times. Eg. Given `[1, 2, 3]` and the number 3, it should return `[1, 1, 1, 2, 2, 2, 3, 3, 3]`.

Exercise 11

`void removeAdjacentDuplicates()`. Eg. Given `[1,2,2,1,3,3,3]` it should return `[1,2,1,3]`.

Exercise 12

`void filterNonNull()` removes all null elements. Eg. Given `[1,null,3,null,5]` it should return `[1,3,5]`.

Exercise 13

`SingleLL<E> zipL(SingleLL<F> l2)`. Eg. Given : `[1, 3, 5]` and `[2, 4, 6]`), it should return `[1, 2, 3, 4, 5, 6]`. Provide a solution in which a new list is constructed. Then provide another solution where the two given lists are "weaved" appropriately.