Data Structures

Exercise Booklet 3: Basic List Exercises

Single-Linked Lists

You are asked to implement the following private and static methods. These methods are added to the class `SingleLinkedList<E>`. Unless otherwise stated, you may freely copy the list provided as argument. Also, you may use helper methods if you feel the need to.

Exercise 1

`boolean isSingleton(Node<E> node)` that returns a boolean indicating whether the list that starts at `node` is a singleton list or not.

Exercise 2

`boolean allEven(Node<Integer> node)` that returns a boolean indicating whether all the integers in the list are even.

Exercise 3

`Integer sumL(Node<Integer> node)` that returns the sum of all the integers in the list.

Exercise 4

`boolean nonDuplicates(Node<Integer> node)` that returns a boolean indicating whether the list starting at `node` has duplicates or not.

Exercise 5

`Node<E> copyL(Node<E> node)` that creates a copy of the list it is given.

Exercise 6

`Node<E> append(Node<E> node1, Node<E> node2)` that appends the two lists. Eg. Given `[1,2,3]` and `[4,5]` returns `[1,2,3,4,5]`.

Exercise 7

`Node<E> reverse(Node<E> node)` that reverses the given list. Provide two solutions. The first one returns a new list, the second reverses the given list (i.e. it does not create a copy of its elements).

Exercise 8

`Node<Integer> doubleL(Node<Integer> node)` that returns a list that is like the one that starts at `node` but where all the integers have been doubled. Eg. given the list `[1,2,3]` it should return `[2,4,6]`.

Exercise 9

`Node<E> repeatLN(Node<E> node, Integer n)` that, given a list that starts at `node` returns a new one in which `n-1` 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

`Node<E> stutterNL(Node<E> node, 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

`Node<Integer> removeAdjacentDuplicates(Node<Integer> node)`. Eg. Given `[1,2,2,1,3,3,3]` it should return `[1,2,1,3]`.

Exercise 12

`Node<Integer> filterEven(Node<Integer> node)` removes all odd numbers. Eg. Given `[1,2,3,4,5]` it should return `[2,4]`.

Exercise 13

`Node<Integer> zipL(Node<Integer> l1, Node<Integer> 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.