German International University
Informatics and Computer Science
Prof. Dr. Slim Abdennadher

Data Structures and Algorithms, Winter semester 2021
Practice Assignment 7

Exercise 7-1

Using the `LinkList` class provided to you on the website, adding the following methods:

- `void insertLast(Object o)`: inserts an object `o` at the end of the linked list. Your implementation should be an **internal recursive** one.
- `void insertLast(Object o)`: inserts an object `o` at the end of the linked list. Your implementation should be an **external** one.
- `Object removeLast()`: removes and returns the last element from the linked list. Your implementation should be an **external** one.

Exercise 7-2 Reverse

Write a Java method `reverse` that takes as input a linked list of integers and reverses it. For example if the input list is:

$$\{1, 2, 3, 8, 10, 15\}$$

then `reverse` should return the list:

$$\{15, 10, 8, 3, 2, 1\}$$

a) Implement the method as an instance method of the class `LinkList`

b) Implement the method externally in a class `LinkListApp`

Exercise 7-3 Reverse in place

Write a Java method `reverse` that takes as input a linked list of integers and reverses it without using any external data structures, only use the input linked list. The reverse should be done in place. For example if the input list is:

$$\{1, 2, 3, 8, 10, 15\}$$

then `reverse` should return the list:

$$\{15, 10, 8, 3, 2, 1\}$$

a) Implement the method as an instance method of the class `LinkList`

Exercise 7-4 Remove Duplicates

Write an external Java method `removeDuplicates` that takes as input a linked list of integers sorted in ascending order and deletes any duplicate nodes from the list.

For example, if the input list is:

$$\{1, 2, 3, 3, 5, 7, 7, 15\}$$

then `removeDuplicates` should change the list to:

$$\{1, 2, 3, 5, 7, 15\}$$

Exercise 7-5 Mix elements of a List

Write an external iterative method `static LinkList mix(LinkList x)` that rearranges the list `x` as follows: the first element is taken from the front followed by the first element from the end, then the second element from the front followed by the second from the end and so on.
Example: The mix method invoked on the linked list $\{1, 2, 3, 4, 5, 6, 7\}$ should return the linked list $\{1, 7, 2, 6, 3, 5, 4\}$.

Exercise 7-6 Cut a List

Write a method `static LinkList cut(LinkList x)` that divides the list `x` into two (roughly) equal halves and puts the second half at the front of the first half.

Write an internal (instance) method `void cutList()` that performs the same operation.

Example: The `cut` method invoked on the linked list:

$$\{1, 2, 3, 4, 5, 6, 7\}$$

will return the linked list:

$$\{5, 6, 7, 1, 2, 3, 4\}$$

Exercise 7-7 Splitting a List

Write a java method `frontBackSplit` that takes as input a linked list of integers. The method should split the list into two sublists: one for the front half and one for the back half and print each. If the number of elements is odd, then the extra element should be placed into the front list. Note that your method should **not alter** the original list; instead, it should just print the contents of the two lists.

For example, if the input list is

$$\{2, 3, 5, 7, 11\}$$

then `frontBackSplit` should print:

```
First sublist:{2,3,5}, Second sublist:{7,11}
```

Note: Getting this right for all cases is harder than it looks. You should check your solution against a few cases (length = 2, length = 3, length = 4)to make sure that the list gets split correctly in all cases. If it works right for length = 4, then it will probably work right for length = 1000.

Exercise 7-8 Alternating Split

Write a Java method `alternatingSplit` that takes as input a linked list of characters and divides its nodes to make two smaller lists. The sublists should be made from the alternating elements in the original list. Your method should not alter the original list, just create two new lists and print them.

For example, if the input list is:

$$\{a, b, c, d, e\}$$

then `alternatingSplit` should print:

```
First sublist: {a, c, e}, Second sublist:{b, d}
```

Exercise 7-9

Implement each of the following methods both internally (using reference manipulation) and externally (using the methods available in the `LinkList` class).

- `merge` that merges two linked lists
- `equals` that returns true if the two lists consist of the same elements
- `intersection` that returns the intersection of two linked lists
- `difference` that returns the difference of two linked lists
- `union` that returns the union of two linked lists

Implement all methods both **internally** inside the `LinkList` class and **externally** inside `LinkListApp` class.