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

Data Structures and Algorithms, Winter semester 2021
Practice Assignment 8

Exercise 8-1 Count

a) Write an **iterative** method called `countIter()` that returns the total number of elements in a linked list, or 0 if the list is empty.

b) Write a **recursive** method called `countRec()` that returns the total number of elements in a linked list, or 0 if the list is empty.

Exercise 8-2 Max of a list

Write an iterative method `max` that takes a linked list as input, and returns the value of the maximum key in the linked list. Assume all keys are positive integers, and return `null` if the list is empty. Implement the method both externally and internally.

Exercise 8-3 Contains

Implement an **iterative internal** method `boolean contains(Object o)` inside class `LinkedList` which returns `true` if the linked list contains an object equal to `Object o`, and `false` otherwise.

Implement a **recursive internal** method `boolean containsRec(Object o)` which performs the same function.

Exercise 8-4 Circular LinkedList

a) A Circular Linked List is a linked list in which the last node points to the first node in the same list. Implement the `CircularLinkedList` class. **Note:** you can use the `LinkList` class on the website for assistance, but **do not** modify the `Link` class. Test your class using a main method.

b) Implement an instance method `void append` in class `CircularLinkedList` which appends the elements of one circular linked list to the end of the other.

Exercise 8-5 Doubly LinkedList

A Doubly Linked List is a linked list in which each node has a link to the previous node as well as the next node. The `DoublyLinkedList` class has the following methods:

- `isEmpty()`: To check whether a list is empty.
- `insertFirst()`: Inserting an item at the beginning of the list.
- `insertLast()`: Inserting an item at the end of the list.
- `removeFirst()`: Deleting an item at the beginning of the list.
- `removeLast()`: Deleting an item at the end of the list.

- The traversal methods: Iterating through the list to display its contents.

Augment the implementation provided on the website with the following methods:

a) `boolean insertAfter(Object key, Object dd)`: Inserting an item `dd` following element `key`. The method will return `false` if the `key` was not found and `true` if the item was inserted successfully. Assume that the list is not empty.

b) `boolean insertBefore(Object key, Object dd)`: Inserting an item `dd` before element `key`. The method will return `false` if the `key` was not found and `true` if the item was inserted successfully. Assume that the list is not empty.

c) `Object deleteKey(Object key)`: Deleting the first entry that is equal to `key`. Assume that the list is not empty.

d) `void insertToSorted(Comparable x)`: Inserting an item x to a sorted list;

e) `void insertionSort()`: Sorting a list using the insertion sort algorithm.
Note: You can use `insertToSorted(Comparable x)`.

f) `DoublyLinkedList insertionSort(DoublyLinkedList list)`: Sorting the `list` using the insertion sort algorithm without using reference manipulation (externally).

g) `void reverse()`: Reverse a list in place.

h) `void reverse2()`: Reverse a list in place using recursion.

i) `int countIter()`: that returns the total number of elements in a linked list, or 0 if the list is empty. (iterative)

j) `int countRec()`: that returns the total number of elements in a linked list, or 0 if the list is empty. (recursive)

k) `static int count(DoublyLinkedList l)`: that returns the total number of elements in a linked list, or 0 if the list is empty. (external)

Exercise 8-6 Queue using Linked List

Implement the Queue ADT using only the DoublyLinkedList implementation provided on the MET website.

- Implement the constructor and all the basic methods for queues: enqueue, dequeue, peek, isEmpty and size.
- Describe the time complexity in terms of Big O notation for all your methods.
- If the DoublyLinkedList implementation did not have the attribute last, we would have called it a singly-ended doubly-linked list. If that was the case, which queue methods would have had a different time complexity? Justify your answer.