Data Structures and Algorithms I Thu 9.2.2017

Exercise 3

13. Write an algorithm that gets as parameters two unsorted listnode-based (*TraLinkedList*) lists (A, B), and which removes from list A all those elements that occur in list B. Algorithm returns the number of elements removed. Do not use ready *removeAll*() operation. What is the time complexity of your algorithm? How to improve the time complexity?

14. Write a linear time algorithm that gets as parameters two sorted listnode-based (*TraLinkedList*) lists (A, B), and which removes from list A all those elements that occur in list B. Algorithm returns the number of elements removed. Do not use ready *removeAll*() operation or set structures.

15. Write an algorithm that reverses the k topmost elements of a stack. If there are $\leq k$ elements, algorithm reverses the whole stack. Use either data structure library *LinkedStack* or Java API *LinkedList*. What it the time complexity of your algorithm?

16. A palindrome is a word/string that reads the same also when read backwards. Write an algorithm that finds out whether a string a palindrome or not. Hint: store the string to a deque character by character, and then do the checking. Take main program from course www-page. What is the time complexity of your algorithm?

The following task X1 is obligatory for all students. X-tasks must be done **oneself** by each student. Copies/versions of the same answer won't be accepted. Answers must be sent by Wed 8.2. 21:00 using the instructions below. You'll receive an automatic reply by email soon after successful submission. If you won't get the email reply, something went wrong. If the reply contains compiler errors, there is something wrong in the file. Then **resend a fixed version**. The answer must contain a short **self evaluation** where you evaluate the functionality, correctness, time complexity, and possible points of improvement of your solution. A correct self evaluation (for a full answer) is worth one point. The points of these tasks form a part of course evaluation.

Send your solution using a www-form, address and credentials of which you got by email. The solution should be a compilable Java source code file of name `userid.java` where `userid` is the first part of your email address. Also the **class name** of your solution must be `userid`. As the submission is Java source code, the self evaluation must be in comments of the program.

Take a skeleton from course www-page. Do not change the header (name, parameters) of the X-task method. Please make sure that the program is compilable as such, i.e., have whole answer in the same class and do not use a package.

X1. Write an algorithm that gets as parameter two sorted lists (ascending order) (A and B) and which returns a new sorted list (ascending order) which has all the elements that occur in both lists at least twice (at least twice in A *and* at least twice in B). Each element will, however, be in the result list only once.

    What is the time complexity of your algorithm? The time will impact the grading the the task. For full points, you need linear time complexity.

    It may be easier to approach the task by making first a version that makes an ordinary intersection, i.e., takes those elements that occur in both lists at least once. If you make only this ordinary intersection, maximum is 4 points (otherwise 6 p).

    You can use either *java.util.LinkedList* or *TraLinkedList*. Do not use set or map types as helping structure. Do not modify the input lists. Take a skeleton from course www-page, do not modify the header of method.