ADS 2023 spring Week 2 Exercises

Exercises for Algorithms and Data Structures at ITU. The exercises are from *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne unless otherwise specified. Color-coding of difficulty level and alterations to the exercises (if any) are made by the teachers of the ADS course at ITU.

1.2.6 - Green A string s is a circular rotation of a string t if it matches when the characters are circularly shifted by any number of positions; e.g., ACTGACG is a circular shift of TGACGAC, and vice versa. Detecting this condition is important in the study of genomic sequences. Design an algorithm that checks whether two given strings s and t are circular shifts of one another.

1.3.7 - Green Describe how you could create a method/function peek() to Stack that returns the most recently inserted item on the stack (without popping it).

1.3.19 - Green Explain how you would create a method/function removeLast(), that removes the last node in a linked list whose first node is stored in the variable `first`. Making a drawing of what is going on is encouraged.

1.3.20 - Green Explain and draw how one could design a method/function delete() that takes an int argument k and deletes the kth element in a linked list, if it exists.

1.3.21 - Green Explain how one could desing a method find() that takes a linked list and a string key as arguments and returns true if some node in the list has key as its item field, false otherwise.

1.3.22 - Green Suppose that x is a linked list Node. What does the following code fragment do? (This question is also in the quiz, but necessary for the question below).

```
t.next = x.next;
x.next = t;
```

1.3.23 - Green Why does the following code fragment not do the same thing as in the previous question?

```
x.next = t;
t.next = x.next;
```

1.3.27 - Green Suppose that you get a reference for the first node in a linked list. Describe how you can find and output the maximum key in the list. Assume that all keys are positive integers, and return 0 if the list is empty.

1.3.24 - Yellow Explain how you would design a method/function removeAfter() that takes a linked-list Node as argument and removes the node following the given one (and does nothing if the argument or the next field in the argument node is null).

1.3.25 - Yellow Explain how you would design a method/function insertAfter() that takes two linked-list nodes (n1 and n2) as arguments. Assume that n1 is in a list, and insert n2 immediately after n1. Do nothing if either argument is null.

1.3.28 - Yellow Develop a recursive solution to the previous question (1.3.27).

1.3.31 - Red A doubly-linked list is a linked list, where each node contains a reference to the previous node in the list in addition to the reference to the next node. We define the nodes of this list to be of the class DoubleNode. Explain how the class DoubleNode differs from a regular Node class in a linked list, and make a drawing of a doubly-linked list data structure.

Next, describe how the following methods/functions work in a doubly-linked list: Insert at the beginning, insert at the end, remove from the beginning, remove from the end, insert before a given node, insert after a given node, and remove a given node.

1.3.48 - Red A double-ended queue or deque (pronounced "deck") is like a stack or a queue but supports adding and removing items at both ends, so where a regular stack has a push() function, a deque has both pushLeft() and pushRight() functions and similarly a deque has popLeft() and popRight() functions instead of just a pop() function.

How can two stacks be implemented using a single deque so that each operation takes a constant number of deque operations?