



# T-201-GSKI Data Structures

## Programming Homework2: Linked Lists and ADTs (5%)

### Objective

The objective of the assignment is to gain hands-on experience implementing linked list data-structures, both single- and double-linked, as well as implementing ADTs using lists.

### Logistics

- You are expected to work on the assignment in a **group of 1-3 persons** (max).
- *Ensure that you test your implementations thoroughly* (it is easy to make mistakes or miss some important edge cases).
- Do not change the names of the provided Python files; they must remain as is.
- **Submit your code in Canvas.** Archive the folder where you keep your Python files to create either as a *zip* or a *tar/tgz* file. Submit only that single archived file.

### Description

You are provided with incomplete classes for a *single-linked list*, *double-linked list*, *stack*, *queue*, and *deque*. You are *not allowed* to change the interface of the provided class methods (you can add *private* methods, if you want).

1. (20%) Implement the four incomplete methods in the *single-linked list class* (in a way that meets the time-complexity requirements stated in their docstring).

The interface to our single-linked list allows us to manipulate only the front and back of the list. The only file you need to change here is *sll.py*.

2. (40%) Implement the incomplete methods in the *double-linked list class* (in a way that meets the time-complexity requirements stated in their docstring).

**First implement** the methods in the *fundamental section*, then implement the remaining methods by calling the ones in the *fundamental section* (do not unnecessarily duplicate code). The only file you need to change here is *dll.py*.

Unlike for the single-linked list, we have augmented the interface of our double-linked list to permit insertion and removal of elements at any *position* in the list (a so-called *positional list interface*, along the lines of that is shown in chapter 7.4 in the textbook).

3. (20%) You are provided with an example *stack* implementation that uses either a linked list (can be either single or double) as an underlying data structure for implementing the *stack* abstract data type (ADT).

Similarly implement the *queue* and *deque* ADTs. *Be careful in your implementation to use only class methods that are common to both the single- and double-linked list classes. The list you use is passed to the ADTs constructor.* The only files you need to change here are *queue.py* and *deque.py*.

4. (20%) Finish implementing the provided *match\_brackets.py* program, using the most appropriate ADT (from the ones you implemented above) for solving the task.

**Good luck!** 😊