

### Warm-up

**Problem 1.** Suppose we implement a stack using a singly linked list. What would be the complexity of the push and pop operations? Try to be as efficient as possible.

**Problem 2.** Suppose we implement a queue using a singly linked list. What would be the complexity of the enqueue and dequeue operations? Try to be as efficient as possible.

### Problem solving

**Problem 3.** Given a singly linked list, we would like to traverse the elements of the list in reverse order.

- a) Design an algorithm that uses  $O(1)$  extra space. What is the best time complexity you can get?
- b) Design an algorithm that uses  $O(\sqrt{n})$  extra space. What is the best time complexity you can get?

You are not allowed to modify the list, but you are allowed to use position/cursors to move around the list.

**Problem 4.** Using only two stacks, provide an implementation of a queue. Analyze the time complexity of enqueue and dequeue operations.

**Problem 5.** We want to extend the queue that we saw during the lectures with an operation `GETAVERAGE()` that returns the average value of all elements stored in the queue. This operation should run in  $O(1)$  time and the running time of the other queue operations should remain the same as those of a regular queue.

- a) Design the `GETAVERAGE()` operation. Also describe any changes you make to the other operations, if any.
- b) Briefly argue the correctness of your operation(s).
- c) Analyse the running time of your operation(s).

**Problem 6.** Consider the problem of given an integer  $n$ , generating all possible permutations of the numbers  $\{1, 2, \dots, n\}$ . Provide a recursive algorithm for this problem.

**Problem 7.** Consider the problem of given an integer  $n$ , generating all possible permutations of the numbers  $\{1, 2, \dots, n\}$ . Provide a non-recursive algorithm for this problem using a stack.