Data structure
Workshop 2

I. Exercise

1.1 A linear list is stored in a singly linked list, where elements are arranged in strictly increasing order of their values. Design an efficient algorithm to delete all elements in the list whose values are greater than mink and less than maxk (if such elements exist). Ensure that the memory of the deleted nodes is freed. Analyze the time complexity of your algorithm.
(Note: mink and maxk are given parameters, and their values may or may not exist in the list.)

1.2 Write an algorithm to reverse a singly linked list in-place (i.e., without using extra space for another linked list).

II. Experiment

Problem Description

A parking lot has n parking spaces arranged in a narrow one-way lane, with the entrance/exit at the right end. Vehicles can only enter from the right and park in the first available space (i.e., the rightmost empty space). When the lot is full, no more vehicles can enter.

When a vehicle leaves, all vehicles parked to its right must first exit the lot (temporarily stored), then the target vehicle exits. After that, the temporarily exited vehicles re-enter the lot in their original order (right to left, so their relative positions are preserved).

Vehicles are identified by their license plates. There are q operations:

1. A vehicle with license plate x tries to enter the lot. If the lot is full, do nothing.
2. The vehicle with license plate x leaves the lot. Output its position (1-based from left to right) before leaving; if x is not in the lot, output -1.
3. Query the license plate of the vehicle in position x (1-based from left to right) in the lot; if the position is invalid, output -1.

Output Requirements:

- For entry/exit operations, output the current state of the parking lot as:

parking lot: license1 license2 ... (no trailing space).

- For query operations, output the license plate or -1.

Input Format:First line: two positive integers n, q. Next q lines: each line starts with an integer op (1-3) followed by a parameter x.

Example Illustration:If the lot has 3 spaces and the current state is A B C (A at left, C at right, near the gate):

- When B leaves:
    - Temporarily exit C (stored in a stack), then B exits (position 2).
    - C re-enters, so the new state is A C.