Week 4 **31.07.2025**

Implementation of Queue using array and linked list

1. Hospital Appointment System Using Array

In a hospital's outpatient department, patients are registered and seen by doctors in the exact order in which they arrive. To simulate this system, implement a linear queue using an array in C. Define an integer array queue[MAX] to store the patient IDs, and use two integer variables front and rear to manage the queue operations. Implement the following functions: enqueue(int patientID) to insert a patient ID; int dequeue() to remove a patient from the front once they complete their consultation; and display() to print all the current patient IDs in the queue from front to rear. As part of the task, insert 5 patient IDs into the queue, perform 2 dequeue operations to simulate consultations, and display the remaining patient IDs in the queue. Handle overflow and underflow conditions appropriately in your implementation.

---

2. Call Center Support System Using Linked List

A call center receives multiple support calls, which must be handled in the order they arrive. Design a queue using a singly linked list to simulate this real-time scenario. Define a structure Node with two fields: an integer callID representing each support request and a pointer *next pointing to the next node in the queue. Use two pointers, front and rear, to manage the head and tail of the queue. Implement the following functions: enqueue(int callID) to insert a new support call into the queue, int dequeue() to remove the call at the front (representing a resolved issue), and display() to show the current queue of unresolved support calls. Begin by enqueuing 4 calls, resolve 2 by dequeuing, and finally display the remaining call IDs to show the current call center queue. Ensure that memory is properly allocated and deallocated in all operations.

---

3. Reverse First K Elements

You are tasked with simulating a ticket counter that processes customers standing in a queue. In this scenario, you are given a queue of integers representing customer tokens and a value K. Your goal is to reverse the order of the first K elements in the queue, while leaving the remaining elements in their original order. For example, if the queue contains 1 2 3 4 5 and K = 3, the output should be 3 2 1 4 5. To achieve this, implement the logic using both a queue and a stack: first, dequeue the first K elements and push them onto a stack to reverse their order; then, pop elements from the stack and enqueue them back; finally, move the remaining N-K elements to the back to maintain their relative order.

****