Assignment 1

Q1) Create a **Hospital Management System** that handles **dynamic patient records** using **nested structures, dynamic memory allocation (DMA), structure pointers, and functions.**

Problem Statement:

A hospital wants to maintain a record of admitted patients. The system should:

1. Use a **nested structure**:
    - Hospital → Contains an **array of Patient structures** and hospital details.
    - Patient → Contains **a nested Doctor structure** with doctor details.
2. Use **dynamic memory allocation (DMA)** to store n patients dynamically.
3. Implement **functions** to:
    - **Admit a patient** (store patient details).
    - **Display all patients** and their assigned doctors.
    - **Discharge a patient** (remove patient details from memory).
4. Pass **structure pointers** to functions for efficient modifications.
5. Ensure proper **memory management** (use delete[] to free memory).

Q2) A university wants to develop a Course Management System that handles dynamic course enrollment, instructor assignments, and student records. The system should:

1. **Use Nested Structures:**
    - **University contains an array of Course structures.**
    - **Course contains an array of Student structures and a Professor structure.**
    - **Professor contains personal and subject details.**
    - **Student contains personal details and grades.**
2. **Use Dynamic Memory Allocation (DMA):**
    - **The system should allocate memory dynamically for n courses and m students per course.**
3. **Use Structure Pointers in Functions:**
    - **Implement functions to:**
        - **Add a new course (assign professor and allocate students dynamically).**
        - **Enroll a student in a course (update student list).**
        - **Display all courses and students.**

        - **Update a student's grade.**
        - **Remove a student from a course (shift students in memory).**
4. **Ensure Proper Memory Management:**
    - **Use new to allocate memory and delete to free memory when removing students or courses.**

Q3) A global Fleet Management System (FMS) is designed to track thousands of delivery vehicles across multiple regions. Your task is to implement a highly efficient system that:

1. **Manages dynamically allocated vehicle records categorized into regions.**
2. **Implements a self-balancing hierarchical data structure (nested structures with dynamic memory allocation).**
3. **Uses structure pointers and function pointers for efficient record retrieval and modification.**
4. **Efficiently removes inactive vehicles by restructuring memory without fragmentation.**
5. **Ensures concurrency safety when multiple users access vehicle data at the same time.**

🚚 **System Requirements:**

1. Nested Dynamic Structures:

- **FleetManager contains dynamically allocated regions (Region struct).**
- **Each Region manages dynamically allocated vehicles (Vehicle struct).**

2. Efficient Data Storage and Retrieval:

- **Vehicles must be stored in an array of pointers to allow easy reallocation.**
- **Vehicles must be assigned a priority score based on fuel level and last update timestamp.**

3. Functional Requirements:

- **Add a new region dynamically.**
- **Add a vehicle to a region (ensure sorted order based on priority).**
- **Find the vehicle with the lowest fuel level in a region.**

- **Remove inactive vehicles (last updated more than X hours ago)(hint: use ctime library).**
- **Print fleet status across all regions.**