🧩 Problem Statement: **Smart Transportation Billing System**

Scenario:

A city is implementing a **Smart Transportation Billing System** for different types of vehicles — such as **Cars**, **Bikes**, and **Electric Vehicles** — that use toll roads. Each vehicle type has different toll calculation rules:

- **Cars** pay a flat rate.
- **Bikes** pay half the rate of cars.
- **Electric Vehicles (EVs)** get a discount due to government policy and pay only 30% of the car rate.

The system should be flexible to allow new vehicle types to be added in the future with their own billing logic.

Requirements:

1. Create a base class `Vehicle` with a virtual method `calculateToll()`.
2. Derive classes `Car`, `Bike`, and `ElectricVehicle` from `Vehicle`.
3. Override the `calculateToll()` method in each derived class to apply the respective toll logic.
4. Use **runtime polymorphism** to process an array of different vehicles and print their toll charges.

Input:

- A list of vehicle types and a base toll amount for cars (e.g., $5.00).

Output:

- Print the vehicle type and the toll amount to be paid.

🧩 Problem 2: **Hospital Management System – Dynamic Consultation Billing**

Scenario:

A hospital wants to build a system to dynamically calculate the **consultation charges** for different types of doctors based on their **specialization** and **experience**.

- **General Physicians** charge a flat consultation fee.
- **Surgeons** charge more, and their fee increases based on years of experience.
- **Specialists** (e.g., cardiologists, neurologists) may have a **dynamic rate** based on the patient's condition severity (scale of 1 to 5).

Requirements:

1. Create an abstract base class `Doctor` with a pure virtual method `calculateConsultationFee()`.
2. Implement derived classes: `GeneralPhysician`, `Surgeon`, and `Specialist`.
3. Override the fee calculation method in each subclass:
    - `GeneralPhysician`: fixed rate (e.g., $50).
    - `Surgeon`: $100 + $10 * years of experience.
    - `Specialist`: $70 + $20 * severityLevel.
4. Use **runtime polymorphism** to loop through a list of doctors and output the name, specialization, and consultation fee.

Input:

- List of doctors with their types, experience, or severity level as applicable.

Output:

- Doctor name, specialization, and calculated consultation fee.

🧩 Problem 3: **Online Learning Platform – Instructor Payment System**

Scenario:

An online learning platform pays its instructors differently based on the **type of course** they deliver:

- **Recorded Course Instructors** get paid a fixed amount per course.
- **Live Class Instructors** are paid per hour.
- **Project Mentors** get a base pay plus a bonus per completed student project.

The company wants a **scalable payment system** using OOP principles.

Requirements:

1. Create a base class `Instructor` with a virtual method `calculateEarnings()`.
2. Implement subclasses:
    - `RecordedCourseInstructor`
    - `LiveClassInstructor`
    - `ProjectMentor`
3. Override the `calculateEarnings()` method:
    - Recorded: $300 per course.
    - Live: $25/hour.
    - Mentor: $200 base + $50 per completed project.
4. Create an array of instructors and display their names and total earnings using polymorphism.

Input:

- Instructor name, type, and relevant work data (hours taught, courses created, projects reviewed).

Output:

- Instructor name and payment for the month.
