SPOT QUESTION FOR MULTITHREADING

1. Multi-Threaded Cinema Ticket Booking System

A cinema ticket booking system needs to handle multiple users booking movie tickets simultaneously. The system should ensure that:

- A user can select a seat and book it if available.
- If the seat is already booked, the user should receive a notification.
- Multiple users (threads) should be able to check seat availability and book tickets concurrently while maintaining data consistency.

1. **Design a Java program** that simulates a cinema ticket booking system using **multithreading**.
2. Implement a **shared resource** (`CinemaBooking`) that maintains seat availability.
3. Use **synchronized methods** to ensure only one user can book a seat at a time.
4. If a user tries to book an already booked seat, display an appropriate message.
5. Demonstrate the functionality with at least **three user threads** booking different seats concurrently.

2. Multi-Threaded Online Food Ordering System

An online food ordering system needs to handle multiple customers placing orders at the same time. The system should ensure that:

- Users can place orders concurrently, but food preparation follows a **first-come, first-served** basis.
- If the restaurant has limited resources (e.g., limited chefs preparing food), the system should handle order processing accordingly.
- Orders should be **processed sequentially** even if multiple users place them at the same time.

1. **Design a Java program** that simulates an online food ordering system using **multithreading**.
2. Implement a **shared resource** (`FoodOrderSystem`) that maintains a queue of food orders.
3. Use **synchronized methods or locks** to ensure orders are processed one at a time.
4. If the kitchen is busy, new orders should wait until the previous order is processed.
5. Demonstrate the functionality with at least **three user threads** placing orders concurrently.