Puzzle Questions

1. Dining Philosophers Problem:
Imagine five philosophers sitting around a circular table. There is one fork between each pair of philosophers. A philosopher needs both the forks adjacent to them to eat, but can only pick up one fork at a time. How would you design a solution that ensures no philosopher will starve (i.e., each philosopher will eventually get to eat) and prevents deadlock?

2. Readers-Writers Problem:
You have a shared data structure that can be accessed by multiple threads. There are two types of threads: readers, which only read the data, and writers, which can both read and write. You need to design a system that ensures:
- Writers get exclusive access to the data.
- Multiple readers can read simultaneously.
- No reader should be kept waiting if the data is available to read (reader priority), and no writer should be kept waiting if it has the chance to write (writer priority).
How would you solve this problem, considering both reader priority and writer priority approaches?

3. Banker's Algorithm:
You are a banker managing a number of clients (processes) that request resources (like memory, CPU time, etc.) from you. You want to make sure that you never allocate resources in a way that could lead to a deadlock. Describe how you would implement an algorithm that determines whether a resource allocation request can be safely granted without leading to deadlock.

4. Sleeping Barber Problem:
Consider a barber shop with one barber, one barber chair, and several waiting chairs. When there are no customers, the barber sleeps. When a customer arrives, they either wake up the barber or, if the barber is cutting hair, they wait in one of the chairs. If all chairs are occupied, the customer leaves. Design a synchronization mechanism that prevents race conditions and ensures that the barber's shop functions smoothly.

5. Page Replacement Puzzle:
You have a memory system with a limited number of physical pages. When a new page is requested and the memory is full, one of the existing pages has to be replaced. Explain how different page replacement algorithms like FIFO, LRU, and Optimal Replacement would work in this scenario. For instance, given a sequence of page requests, how would these algorithms determine which page to replace?