README : QUESTION 3

Bridge Crossing Problem Implementation

This C code simulates a bridge crossing scenario where cars from the left and right sides need to cross a bridge with a limited capacity. The code uses pthreads for multi-threading and semaphores for synchronization.

Key Components:

- **passing Function:** Simulates a car crossing the bridge, displaying information about the car's side and its crossing status.
- **Semaphores:** The code utilizes two semaphores:
  - `mutex`: Controls access to the critical section to ensure mutual exclusion when updating the `bridgecar` count.
  - `bridge`: Limits the number of cars allowed on the bridge simultaneously, preventing congestion.

Deadlock Prevention:

To prevent deadlocks in this bridge crossing scenario, a few strategies have been implemented:
1. **Semaphore Usage:** Semaphores (`mutex` and `bridge`) are used to ensure exclusive access to critical sections and regulate the number of cars allowed on the bridge concurrently.
2. **Mutex-Controlled Increment/Decrement:** The `bridgecar` variable, representing the number of cars on the bridge, is managed within critical sections controlled by the `mutex` semaphore. This prevents multiple threads from accessing/updating `bridgecar` simultaneously.
3. **Sleep and Retry Mechanism:** If the bridge's maximum capacity is reached, a waiting car will sleep for a short period and retry until it can proceed. This avoids a deadlock situation where all cars might get stuck indefinitely.

Assumption:
Not necessary to make left and right function.