1. Design a bounded-buffer solution using semaphores where there are multiple producers and consumers. explain how your solution ensures no deadlock, starvation or race condition.
2. The following pseudocode is a poorly implemented version of the Peterson's algorithm for solving the critical section but something is wrong ! explain the issue in detail.

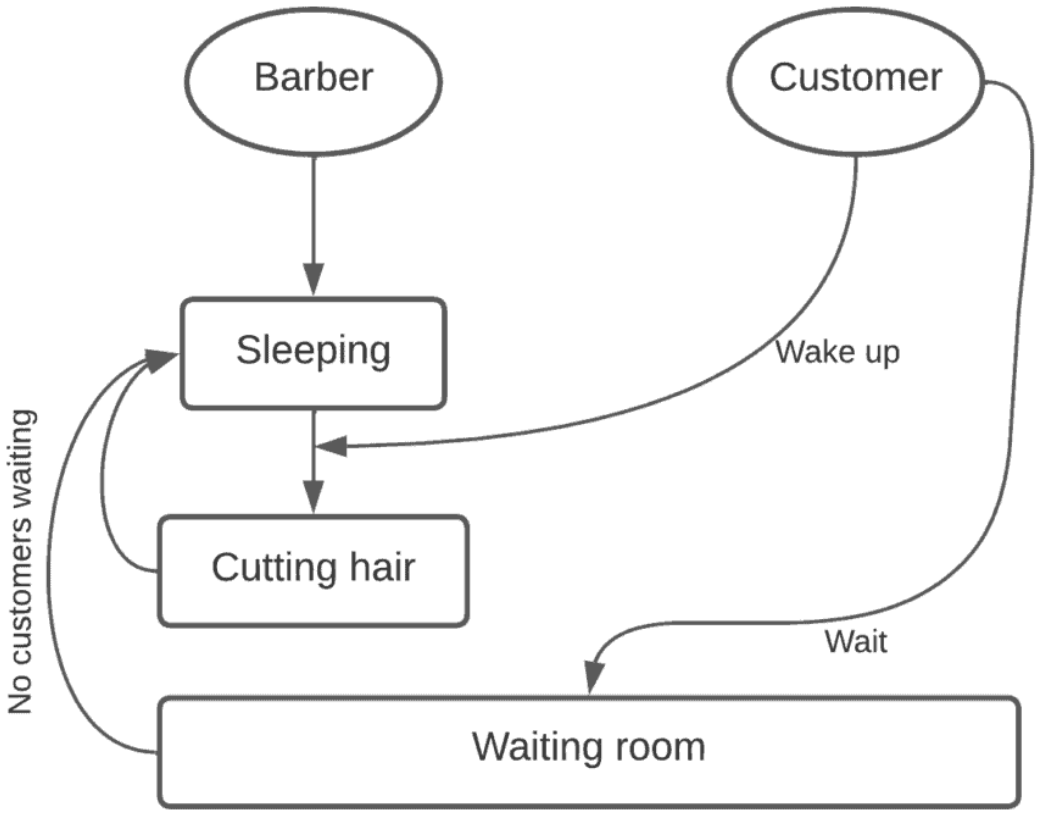
```
boolean flag[2] = {false, false};
int turn = 0;

while (true) {
    flag[i] = true;
    turn = 1 - i;
    while (flag[1 - i] && (turn == 1 - i || i == 0)) {
        // Busy wait
   }

    // Critical Section
    // Code to access the shared resource

    flag[i] = false;

    // Remainder Section
    // Code outside the critical section
}
```

3. There is a classic problem IPC Synchronization problem proposed by Dijkstra named " The sleeping barber". The problem explanation is as follows:

the barbershop has one barber, one barber chair, and a waiting room with N chairs for the customers.

characteristics of the problem:

- The barber sleeps when there is no customer in the waiting room. Therefore, when a customer arrives, he should wake up the barber
- Customers can enter the waiting room and should wait for whether the barber is available or not
- If other customers keep coming while the barber is cutting a customer's hair, they sit down in the waiting room
- Customers leave the barbershop if there is no empty chair in the waiting room



You need to synchronize the barber and the customers so there wouldn't be any race conditions.
What is your solution?

Good luck and Have fun synching (in) the problems(pun intended).