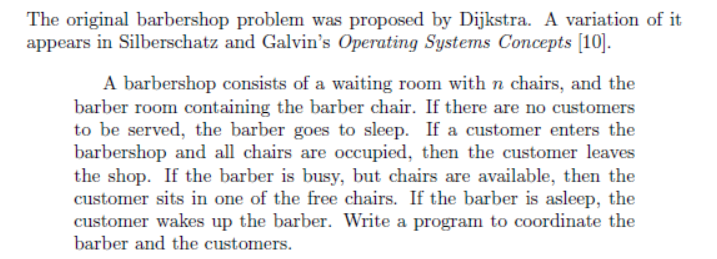
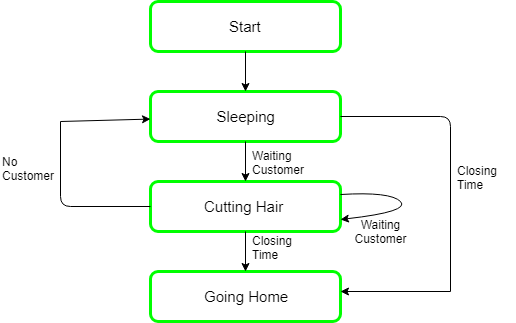
**Problem 1: The barbershop problem**

****

[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwiVhoCztYDjAhX58OAKHaFsBC4QjRx6BAgBEAU&url=%2Furl%3Fsa%3Di%26rct%3Dj%26q%3D%26esrc%3Ds%26source%3Dimages%26cd%3D%26ved%3D%26url%3Dhttps%253A%252F%252Fwww.geeksforgeeks.org%252Foperating-system-sleeping-barber-problem%252F%26psig%3DAOvVaw191adfAu4rWR5DT3TJWioA%26ust%3D1561407237398236&psig=AOvVaw191adfAu4rWR5DT3TJWioA&ust=1561407237398236)

In computer science, the sleeping barber problem is a classic inter-process communication and synchronization problem between multiple operating system processes. The problem is analogous to that of keeping a barber working when there are customers, resting when there are none, and doing so in an orderly manner. The barber has one barber chair and a waiting room with a number of chairs in it. When the barber finishes cutting a customer's hair, he dismisses the customer and then goes to the waiting room to see if there are other customers waiting. If there are, he brings one of them back to the chair and cuts his hair. If there are no other customers waiting, he returns to his chair and sleeps in it. Each customer, when he arrives, looks to see what the barber is doing. If the barber is sleeping, then the customer wakes him up and sits in the chair. If the barber is cutting hair, then the customer goes to the waiting room. If there is a free chair in the waiting room, the customer sits in it and waits his turn. If there is no free chair, then the customer leaves.

Solution:

**Semaphore Customers = 0;**

**Semaphore Barber = 0;**

**Mutex Seats = 1;**

**int FreeSeats = N;**

**Barber {**

**while(true) {**

**/\* waits for a customer (sleeps). \*/**

**Customers.Wait();**

**/\* mutex to protect the number of available seats.\*/**

**Seats.Wait();**

**/\* a chair gets free.\*/**

**FreeSeats++;**

**/\* bring customer for haircut.\*/**

**Barber.Signal(s);**

**/\* release the mutex on the chair.\*/**

**Seats.Signal();**

**/\* barber is cutting hair.\*/**

**}**

**}**

**Customer {**

**while(true) {**

**/\* protects seats so only 1 customer tries to sit**

**in a chair if that's the case.\*/**

**Seats.Wait(); //This line should not be here.**

**if(FreeSeats > 0) {**

**/\* sitting down.\*/**

**FreeSeats--;**

**/\* notify the barber. \*/**

**Customers.Signal();**

**/\* release the lock \*/**

**Seats.Signal();**

**/\* wait in the waiting room if barber is busy. \*/**

**Barber.Wait();**

**// customer is having hair cut**

**} else {**

**/\* release the lock \*/**

**Seats.Signal();**

**// customer leaves**

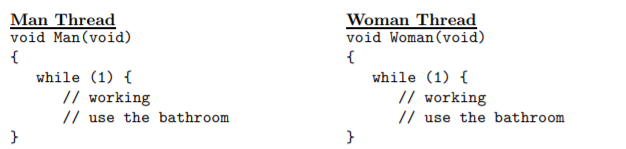
**}**

**}**

}

**Problem 2: The unisex bathroom problem**

A unisex bathroom is shared by men and women. A man or a woman may be using the room, waiting to use the room, or doing something else. They work, use the bathroom and come back to work. The rule of using the bathroom is very simple: there must never be a man and a woman in the room at the same time; however, people with the same gender can use the room at the same time.

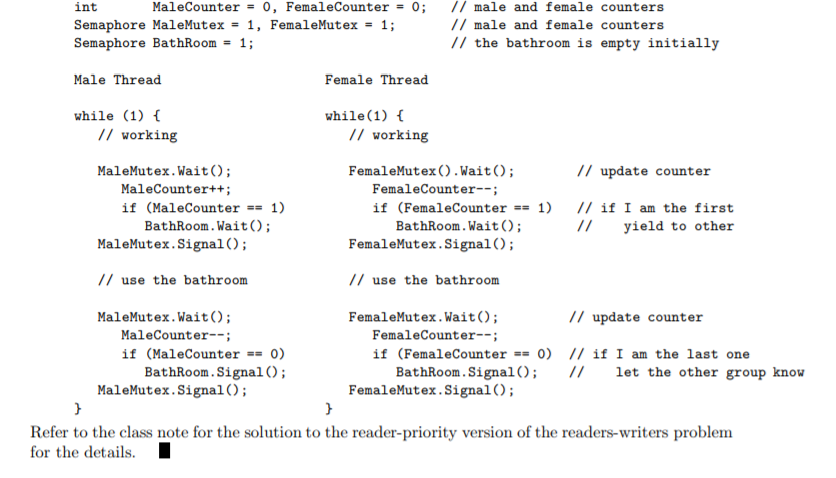


Declare semaphores and other variables with initial values, and add Wait() and Signal() calls to the threads so that the man threads and woman threads will run properly and meet the requirement. Your implementation should not have any busy waiting, race condition, and deadlock, and should aim for maximum parallelism.

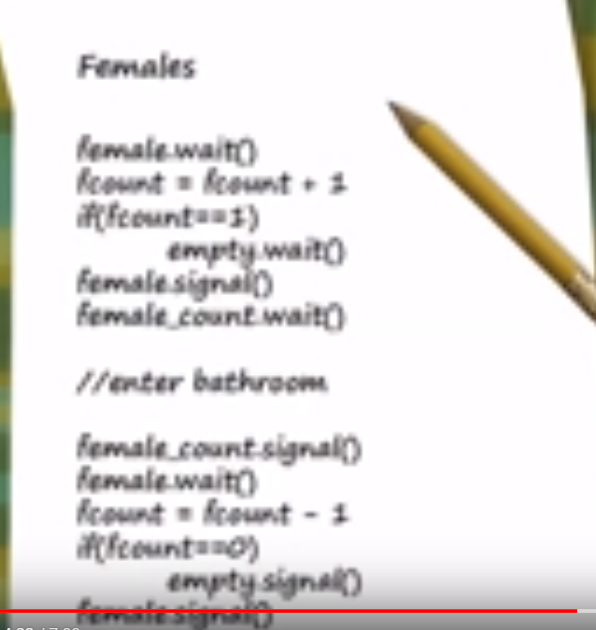
**A convincing correctness argument is needed. Otherwise, you will receive no credit for this problem**.

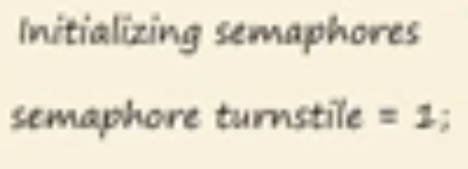
**Answer:**

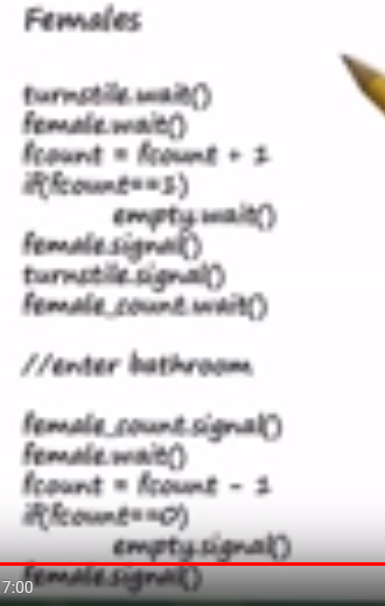
This is a simple variation of the reader-priority readers-writers problem. More precisely, we allow the “writers” to write simultaneously. Therefore, the writers have the same structures as the readers. We need to maintain two counters, one for the males MaleCounter and the other for the females FemaleCounter. Of course, we need two Mutexes MaleMutex and FemaleMutex for mutual exclusion. In addition, there is a semaphore BathRoom to block the males (resp., females) if the room is being used by the females (resp., males). Note that the male thread and female thread are symmetric.

****

**To avoid starvation:**

****

****

****