**F Table Of Contents**

1. **Multiple Sleeping Barber Problem**
   1. Pseudocode……….……………………………………………..2
   2. Deadlock Examples……………………………………………..2
   3. Deadlock Solution………………………………………………..3
   4. Starvation Problem…..…………………………………………..3
   5. Starvation Solution...…………………………………………….3
   6. Real-world Problem………………………………………….….4

**Multiple Sleeping Barber Problem**

# Pseudocode:

BEGIN

Customer=0 then the barbers sleep

Customer>0 then customer wakes up barber

Customer>n (number of barbers) then

if empty chairs>0 then customer sits in waiting chair (x)

Else empty chairs (x)=0 then customer leaves

END

There are three Semaphores>>

First is for the customer which counts the number of customers present in the waiting room.

Second, for the barber to know when barber is giving a haircut and when barber is sleeping.

Third semaphore for the barber chairs to ensure only 1 customer gets a haircut at a time.

In the solution, the customer has the record of the number of customers waiting in the waiting room if the number of customers is equal to the number of chairs in the waiting room then the upcoming customer leaves the barbershop.

When the barber shows up in the morning, he executes the procedure barber, causing him to block on the semaphore customers because it is initially 0. Then the barber goes to sleep until the first customer comes up.

When a customer arrives, he executes customer procedure the customer acquires the mutex for entering the critical region, if another customer enters thereafter, the second one will not be able to anything until the first one has released the mutex. The customer then checks the chairs in the waiting room if waiting customers are less than the number of chairs then he sits otherwise he leaves and releases the mutex.

If the chair is available, then customer sits in the waiting room and increments the variable waiting value and also increases the customer’s semaphore this wakes up the barber if he is sleeping.

At this point, customer and barber are both awake and the barber is ready to give that person a haircut. When the haircut is over, the customer exits the procedure and if there are no customers in waiting room barber sleeps.

# Dead Lock Problem:

## EXAMPLE OF DEADLOCK

As the barber finishes cutting the current customer’s hair, a new customer enter the barber shop and head to the waiting room. As the customer was going to the waiting room the barber finishes the haircut and checks the waiting room before the new customer arrives and finds it empty thus gets back to sleep. Now the new customer will wait for the barber to finish, whereas the barber is sleeping as he didn’t realize that a new customer arrived.

# 1.3)Solution of Deadlock:

The first customer that arrives at the barber’s chair will get his haircut done and the customer that arrives later will have to wait for his turn. Thus we should implement a data structure, known as Queue, which follows the First-in First-out principle. Instead of using a single Semaphore for customers, a list of Semaphore will be used. From now onward, we will refer a customer as a thread. So for threads, a list of Semaphore will be used, named as a queue.

# 1.4)Starvation Problem:

## EAMPLE OF STARVATION

When the customer has to wait for a long time because he does not know that the previous customer has already got his hair cut and a new customer came directly taking his turn. Thus the customer will continue waiting as new customers arrive at the same time the current customer is getting his hair done.

# 1.5)HOW TO SOLVE IT:

The problem of starvation can be solved with a first-in first-out (FIFO) queue. The semaphore would provide two functions: wait() and signal(). Where wait() is used to let the customers know that the barber is busy cutting the current’s customer hair, and signal() is used to notify the first customer that arrived as the barber was busy cutting the current customer’s hair, that the barber is now ready to cut his hair.

# 1.6) Explanation for real world application and how did apply the problem :

* There is a barber shop which has k barbers, n barbers chair, and number of chairs for waiting customers entered by user.
* If there is no customer, then the barber sleeps in their chair.
* When a customer arrives, he has to wake up the barber.
* If there are many customers and the barber is cutting a customer’s hair, then the remaining customers either wait if there are empty chairs in the waiting room or they leave if no chairs are empty.
* There are three semaphores in this application,
  + The First one is for the customer which counts the number of customers present in the waiting room (customer in the barber chair is not included because he is not waiting).
  + Second, the barber 0 or 1 is used to tell whether the barber is idle or is working.
  + Third semaphore is used to ensure that only one customer gets a haircut at a time.
  + In the solution, the customer has the record of the number of customers waiting in the waiting room if the number of customers is equal to the number of chairs in the waiting room then the upcoming customer leaves the barbershop.
* There are two problems will appear deadlock and starvation problem. The deadlock appears when the barber finishes cutting the current customer’s hair, a new customer enter the barber shop and head to the waiting room. As the customer was going to the waiting room the barber finishes the haircut and checks the waiting room before the new customer arrives and finds it empty thus gets back to sleep. Now the new customer will wait for the barber to finish, whereas the barber is sleeping as he didn’t realize that a new customer arrived while the starvation the customer has to wait for a long time because he does not know that the previous customer has already got his hair cut and a new customer came directly taking his turn. Thus the customer will continue waiting as new customers arrive at the same time the current customer is getting his hair done.
* Then we solve both problems with First-in-First-out queue so we solve the deadlock problem as the first customer that arrives at the barber’s chair will get his haircut done and the customer that arrives later will have to wait for his turn. In addition to solve the starvation problem we will add wait() and signal() operations , Where wait() is used to let the customers know that the barber is busy cutting the current’s customer hair, and signal() is used to notify the first customer that arrived as the barber was busy cutting the current customer’s hair, that the barber is now ready to cut his hair.