Multiple Sleeping Barber problem

Pseudocode :

```
Semaphore Customers = 0;

Semaphore Barber = 0;

Mutex Seats = 1;

int FreeSeats = N;


Barber {
    while(true) {
            /* waits for a customer (sleeps). */
            down(Customers);

            /* mutex to protect the number of available seats.*/
            down(Seats);

            /* a chair gets free.*/
            FreeSeats++;

            /* bring customer for haircut.*/
            up(Barber);

            /* release the mutex on the chair.*/
            up(Seats);
```

```
                /* barber is cutting hair.*/
        }
}

Customer {
        while(true) {
                /* protects seats so only 1 customer tries to sit
                in a chair if that's the case.*/
                down(Seats); //This line should not be here.
                if(FreeSeats > 0) {

                        /* sitting down.*/
                        FreeSeats--;

                        /* notify the barber. */
                        up(Customers);

                        /* release the lock */
                        up(Seats);

                        /* wait in the waiting room if barber is busy. */
                        down(Barber);
                        // customer is having hair cut
                } else {
                    /* release the lock */
                    up(Seats);
                    // customer leaves
                }
        }
}
```

Examples of Deadlock :

A deadlock occurs if each process or thread in a group is holding a resource that is needed by another member of the group, in a circular-waiting pattern. Multiple Sleeping Barber Problem if all customers want to getHairCut in the same time and each starts to invoke getHairCut this will, eventually leads to a deadlock because every one will be waiting for his neighbor forever.

How did solve deadlock:

```
int counts = 0 //number of customers;
mutex = Semaphore(1);
customer = Semaphore(0);
barber = Semaphore(0);

void customer (void){
wait(mutex);
if (counts==n+1) {
signal(mutex);
leave();
}
counts +=1;
signal(mutex);
signal(customer);
wait(barber);
getHairCut();
wait(mutex);
counts -=1;
signal(mutex);
}

void barber (void){
wait(customer);
signal(barber);
cutHair();
}
```

Example of starvation problems 1: when a customer waits too long time waiting when the customer don't follow order

Example of starvation problem 2: when the barber calls out customer randomly

Example of starvation problem 3: customer 3 and customer 2 on the waiting seats

And customer 3 enter first to the barber

Solve the problem of starvation:

Add customer once they arrive and barber take from it   each customer in order, we solve the problem number 3 using a queue.

Explain for real world application:

In this section we will apply sleeping barber problem on a bank system image we have a bank and two accountants that see the people who wants services and we have many chairs for customers in the bank.

if we have n chairs and those chairs are now n-2 and two customers went to the bank, then we don't have any available chairs for newcomers, so if we have new customers they will exit the bank as there is no chairs for them, now the accountant has finished the service of a customer and other customer must go to the window of the accountant, now we have again n-1 chairs so if a newcomer came from outside the bank he will stay in the queue of services

Represent the starvation :

we will represent the starvation as when a customer number 10 go to the accountant before customer number 7, 8 and 10.