Indian Institute of Technology Jodhpur
Operating Systems Lab (CS330)
Assignment 5

Dated 7th April, 2021 **Total marks: 30**

The Sleeping-Barbershop problem is a classical synchronization problem, demonstrated in the Tanenbaum book as follows:
"The barber shop has one barber, one barber chair, and n chairs for waiting customers, if any, to sit on. If there are no customers present, the barber sits down in the barber chair and falls asleep, as illustrated in. When a customer arrives, he has to wake up the sleeping barber. If additional customers arrive while the barber is cutting a customer's hair, they either sit down (if there are empty chairs) or leave the shop (if all chairs are full). The problem is to program the barber and the customers without getting into race conditions."

In this assignment you are going to implement a variant of this problem, as proposed by Ralph Hilzer. The problem statement and solution hint can be found in page no. 133 of "The Little Book of Semaphores". Read the problem statement, synchronization constraints mentioned and understand the solution strategy provided.

This version of the problem has three chairs, three barbers, and a waiting area that can accommodate four customers on a sofa and has a standing room for additional customers. Current pandemic situation limits the total number of customers in the shop to 20. A customer will not enter the shop if it is filled to capacity with other customers. Once inside, the customer takes a seat on the sofa or stands if the sofa is filled. When a barber is free, the customer that has been on the sofa the longest is served and, if there are any standing customers, the one that has been in the shop the longest takes a seat on the sofa. When a customer's haircut is finished, any barber can accept payment, but because there is only one cash register, payment is accepted for one customer at a time. The barbers divide their time among cutting hair, accepting payment, and sleeping in their chair waiting for a customer.

As a further twist to this problem, you are going to implement one additional role and one new activity, called 'the gatekeeper' and 'the cleaning' to the existing problem.

The gatekeeper: The duty of the gatekeeper is to keep track of the customers who are waiting to enter the shop. Every time a customer enters the shop, collects a token from the gatekeeper and submits it while leaving the shop. So, if the shop is full in capacity, a customer does not leave but waits outside in a FIFO queue that is maintained by the gatekeeper. Once a customer exists the shop and there is more room to accommodate the new customer, a customer who is waiting for the longest period of time can enter the shop. The total number of tokens available with a gatekeeper is equal to the total capacity in the shop (the number of chairs in the waiting room + the number of barber chairs + the customers waiting in the standing area inside the shop.

The cleaning: After a customer is given a haircut, the barber has to clean the chair and sanitize the equipment in order to make them usable for the next customer. Now, as there is only one cleaning kit (a shared resource), the barbers have to synchronize among themselves for accessing it. If one barber is currently using the cleaning kit following a haircut, and another barber also need to access it, he has to wait. So, in this modified version of the problem, a barber spends his time by cutting hair, accepting payment, cleaning the chair and sleeping. Note that there is a sequential dependency between the activities in this order: cutting hair → cleaning → accepting payment→ sleeping (if permits). Once the customer is done, it cannot leave the shop until the payment is made.

1. The number of barbers, the number of chairs in the waiting room, and the waiting room capacity are given as part of the command line options -b, -c, and -w. The number of barber chairs is always the same as the number of barbers.
2. Customers in the waiting room shall be selected by the barbers in a FIFO order.
3. The program shall print a line of message for each of the following events (the **three colors below** indicate the corresponding thread that shall produce the output). Replace each blank with the proper customer/barber ID.

    a. Customer ___ arrives at the barbershop
    b. Customer _____ collects a token from the gatekeeper
    c. Customer ___ enters the barbershop
    d. Customer ___ sits in the waiting room
    e. Customer ___ sits on a barber chair ___ to get a haircut
    f. Customer ___ leaves barber chair ___ to pay
    g. Customer ___ exits the barbershop after paying the service
    h. Barber ___ is sleeping, waiting for customer
    i. Barber ___ starts haircut of customer ___
    j. Barber ___ finishes haircut of customer ___
    k. Barber ___ is cleaning.
    l. Barber receives the payment from customer _____
    m. The cashier receives payment from customer ___
    n. The customer_____ submits the token to the gatekeeper
    o. The customer _____ leaves the shop.

4. Your program shall exit after the last customer exit the barbershop. Since the gatekeeper and the barber threads will be running in a loop, your program shall be designed to have these threads **exit the loop gracefully**.
5. Customers invoke the following functions in order: enterShop, sitOnSofa, sitInBarberChair, waitforPayment, pay, exitShop, leaveShop.
6. Barbers invoke cutHair, cleanChair, and acceptPayment.
7. Gatekeeper decrements a token counter (initialized to maximum capacity) each time a customer enters the shop and increments it whenever a customer leaves the shop.
8. Customers cannot invoke enterShop if the shop is at capacity.
9. Customers wait in a queue outside the shop, if the shop is at capacity.

10. If the sofa is full, an arriving customer cannot invoke sitOnSofa until one of the customers on the sofa invokes sitInBarberChair.
11. If all three barber chairs are busy, an arriving customer cannot invoke sitInBarberChair until one of the customers in a chair invokes pay.
12. A customer can not pay, until the barber invokes cleanChair
13. A barber cannot invoke cleanChair if the cleaning kit is not available
14. The customer has to pay before the barber can acceptPayment
15. The barber must acceptPayment before the customer can exitShop
16. The customer must return the token to the gatekeeper before he invokes leaveShop

Note that you have to implement all the functions associated with these new role and activity stated, on top of the existing solution to the problem.

Extra credit: Note that with concurrent threads running from within your program, the console screen (stdout) is now a shared resource. You will get an extra credit if you can implement a synchronized printf() to print the messages in the stdout by the concurrent threads.

Prepare a readme file to explain whether your solution is free from deadlock and starvation.

Submit your program in a zip file containing Ass5_Roll.c, a log file (containing the output of your program as a series of messages and the readme file (the analysis)

Submission deadline: 17th April, 2021 11:59 hrs.

Evaluation Policy:

- Correct implementation of the Hilzer's Sleeping Barbarshop problem. (10 pts)
- Correct implementation of the Gatekeeper. (6pts)
- Correct implementation of the cleaning activity. (6pts)
- Printing of the messages for logging the events. (4pts)
- The correct explanation of the starvation and deadlock free solution. (4pts)
- Synchronized printing of the messages. (Extra 5pts)