Assignment Based Quiz
Duration: 24 Hours

There is a high voltage match between Manchester City and Tottenham Hotspurs on December 2. Fans of both teams are very excited about the match and they are keen to watch the match live in the stadium. Tickets for the match have almost been sold out. There are only 10 tickets left.

- Fans have to maintain a queue according to the first come first serve manner while purchasing tickets.
- In order to maintain the rule a serial number has been assigned to each who is in the queue.
- Serials have been assigned in 1, 2 , 3, ..., n according to the first come first serve order.
- The current scenario is, there are 10 tickets available and 12 persons in the queue. And there are 2 booths for purchasing tickets.
- Tickets gets sold in the following order, first two persons (1 and 2 in this case) gets called to two booths together but after the purchase gets completed of person 1 then person 2 starts the procedure of his purchase then after the completion of his purchase both of them left booths then next two persons from the queue gets called to booths for purchasing (3 and 4 in this case).
- If tickets get finished, booths get closed. Then nobody from the queue can purchase tickets anymore. That is how the entire ticket selling procedure works.
- According to the current scenario, there are 10 tickets left and 12 persons in the queue for purchasing tickets.

Design a proper solution to the scenario that how the entire procedure of selling tickets can be synchronized based on the procedure described above.
You have to design a solution and explain it briefly in order to provide the solution. Consider the number of remaining tickets as the shared variable and order of persons in the queue 1,2,3,..........,12. Consider 12 persons as 12 threads here.
The outcome of the solution should look like the following:

```
Person 1 is purchasing
Tickets left: 10
Person: 1, Purchase Done
Person 2 is purchasing
```

```
Tickets left: 9
Person: 2, Purchase Done
Person 3 is purchasing
Tickets left: 8
Person: 3, Purchase Done
Person 4 is purchasing
Tickets left: 7
Person: 4, Purchase Done
Person 5 is purchasing
Tickets left: 6
Person: 5, Purchase Done
Person 6 is purchasing
Tickets left: 5
Person: 6, Purchase Done
Person 7 is purchasing
Tickets left: 4
Person: 7, Purchase Done
Person 8 is purchasing
Tickets left: 3
Person: 8, Purchase Done
Person 9 is purchasing
Tickets left: 2
Person: 9, Purchase Done
Person 10 is purchasing
Tickets left: 1
Person: 10, Purchase Done
Person 11 can not purchase
Tickets left: 0
Person: 11, Purchase failed
Person 12 can not purchase
Tickets left: 0
Person: 12, Purchase failed
```

Ps: Those who will construct proper solutions in code using c language will get bonus marks.

Plagiarism, of any sort, will be strictly dealt with. Copied answers from ChatGPT, Paraphrase from Quilbot or other usage of AI will result in a 0 for the quiz. No exceptions.

Form Submission Link:
https://docs.google.com/forms/d/e/1FAIpQLSe6GtMWCXZD-bfT5BHQzuRfZ05z0Hu3U5VMuCvFjf51G1IQSQ/viewform?usp=sf_link

This Form will be closed at 29/11/23 5:00PM

Design a proper solution to the scenario that how the entire procedure of selling tickets can be synchronized based on the procedure described above.

For the scenario, both mutex and semaphore can be used while coding. But as the scenario specifically said the system works in pairs, two booths, two purchasers, done purchasing in pairs. Then next in queue, purchaser 3 and 4 goes to the two booths respectively. So, it seems to follow semaphore process synchronization. Here, the critical section, ticket purchasing, allows multiple threads (person) with only one person to access at a time. By starting with initializing the semaphore with count 1 which allows two threads (persons) to access the ticket purchase process at a time.

Then thread creation represents the person in the queue, attempting to purchase tickets. When semaphore is acquired by thread successfully it proceeds to purchase the tickets and semaphore count effectively controls the access to the critical section ensuring that only two threads proceed at a time. In short, the access to the critical section is managed by the semaphore that allows threads to purchase tickets in pairs. After one ticket purchase, semaphore releases by incrementing the count which allows the waiting threads to now access the critical section. At the end when threads are completed the purchase process, semaphore gets destroyed.

To sum up the synchronization that explained so far, semaphore initiated with a count 1 that controls access to the critical section allowing two threads (people) execute the ticket purchase process at a time. And these threads execute the purchase in pairs, in concurrent and sequential order of two people being called to the booths as the specific order. Moreover, semaphores manage access to the shares resource which is how many tickets left, and ensures the synchronized and controlled access among multiple threads which are being the people in queue who wants to purchase the tickets and orderly ticket sales, also prevents race conditions.