**Slide1:** Assalamu alaikum. My name is Abdullah al Tamim, and my other groupmates are Fatema Akter and Fahad Ahammed. The topic of our mini-project was DENTAL CLINIC PROBLEM.

**Slide2:** In detail, suppose we have a dental clinic where there is only one dentist and one dental chair. And there are n waiting chairs. When there are no patients, the dentist sleeps in his chair. When a patient arrives, he has to wake up the dentist. Only one patient can take treatment from the dentist at a time. Other patients will wait if there are empty chairs. Otherwise, they will leave. Now we have to synchronize the activities of the dentist and the patients.

**Slide3:** To do so, we have used POSIX threads, Mutex lock, and Semaphore.

**Slide4:** We have represented the dentist as a thread, named dentist, and patients as an array of threads, named array\_patient. To create and initialize the threads we have used the pthread\_create function. Finally, the pthread\_join function will make the parent thread wait, until the child thread is terminated.

Now My groupmates Fahad Ahmed and Fatema Akter will continue the rest of the topics. Thank you.

**Slide5:** Thank you Abdullah al Tamim. Now I will discuss the use of Mutex lock in our project. We have used the mutex lock to handle the critical section problem. Here the pthread\_mutex\_lock function locks the specified critical section. If the critical section is already locked, the calling thread blocks until the critical section becomes available. When the thread completes its function, it unlocks the critical section again. In this way, we handled the race condition or the critical section problem.

**Slide 6:** We have used semaphores to synchronize the activities of the threads. We declared total 3 semaphore-type objects and initialized them using the sem\_init function. The sem\_wait function decreases the value of the semaphore, and if the value is less than 0, then it puts the threads in the block. Finally, the sem\_post function increases the value of the semaphore and unblocks the blocked threads.

**Slide 7:** This is the total flowchart of our project. Here first of all we take input the number of patients and number of chairs from the user. Then if there is no patient taking treatment from the doctor then a patient goes to take treatment, if the doctor is already busy then the patient will wait in the waiting chair if there are enough chairs available, otherwise, they will take leave. After all waiting patients take their treatment the program will end.

Now my groupmate Fatema Akter will explain rest of the topic.

**Slide 8:** Thank you Fahad ahammed, Now I will discuss some of the inputs and outputs of our project. Suppose when the number of patients and chairs both are 5. One person goes to take treatment from the doctor and the other 4 person waits in the chair and 1 chair remains empty. Then all the patients will take treatment from the doctor one by one and will take leave.

In the second input, the number of patients is 5 and the number of chairs is 3. So, if one patient goes to the doctor, 4 patients are left but there are only 3 chairs. So, one patient will leave without treatment and the rest of them will take treatment in the respective order.

**(go to next slide)**

**Slide 9:** In the third input when there are no patients and 5 chairs, the doctor sleeps and the program terminates.

For the last input, when the number of patients is 3 and chairs is 0, only 1 patient directly goes to the doctor and takes treatment, others take leave.

So this was all about our presentation, Thank you, everyone.