PA3 – Report

Enter Method

The pseudocode can be found below:

1. Printing that the thread is arrived in the court.
2. Incoming thread is decrementing the *match_played* semaphore by one. If the semaphore value is negative, thread will wait.
3. Then, the count of people in the court will be incremented by one.
4. If referee does not exist
    a. If the number of people in the court is the same as the total people, algorithm set *match_exist* to true. And prints the statement: “Thread ID: 25, There are enough players, starting a match.”
    b. If the number of people in the court is not the same as the total people, algorithm prints “Thread ID: 25, There are only %d players, passing some time.”
5. If referee exist
    a. If the number of people in the court is the same as the total people, algorithm set *match_exist* to true, and prints "Thread ID: 25, There are enough players, starting a match.”. Then pick this thread as a referee.
    b. If the number of people in the court is not the same as the total people, algorithm prints “Thread ID: 25, There are only %d players, passing some time.”

In this function, I used mutex and semaphore. The purpose of the mutex is to make sure that the number of people in the court is counted atomically. When it comes to the semaphore, I initialized it like this *sem_init*(&match_played, 0, total_people) because when every people that is needed to the game will decrement it by 1 and no other player can enter to the match. Waiting threads must wait for last thread to leave.

Leave Method

The pseudocode can be found below:

1. If there is no match, prints "Thread ID: 25, I was not able to find a match and I have to leave." The number of people in the court will be decremented by 1 and return.
2. If there is a match
    a. The algorithm checks if this thread is the referee that I appointed in the enter method. If yes, print "Thread ID: 25, I am the referee and now, match is over. I am leaving." Then decrement the number of people in the court by one and wait in the barrier. Then increment match_played semaphore by 1.
    b. If the thread is not a referee, it will wait in the barrier again. When all the threads get out from the barrier, they will print "Thread ID: 25, I am a player and now, I am leaving." And decrement the number of people in the court by 1.
        i. If people in the court is 0
            1. We set match_exist to false and print "Thread ID: 25, everybody left, letting any waiting people know." And increase the semaphore match_played by 1.

In this method, I couldn't handle some cases like *court_test 12 4 1* because I could not arrange the semaphore well.

I used mutex, semaphore and barrier to implement this method. Firstly, I used mutex to protect critical sections. Secondly, I used semaphores to keep track of the threads' entrance and their exit. It works correctly when there is at most 1 match but if there are more matches to be played, the algorithm fails. Thirdly, I used barriers in both cases which are referee exist or not. It works fine for because if the referee exists, all the other threads will wait for referee to prints its output. If there is no referee, it prints the last thread that enters the barrier, then the other threads waiting in the barrier will be printed.