**CS307 2022 SPRING PA3 – Riding to a Soccer Game REPORT**
**UĞUR ÖZTUNÇ 28176**

At the beginning of the main function, after main inputs are stored in an integer array name ‘counts’, validity check is performed and if input numbers are not valid proper output is printed and program terminates. If inputs are valid, program continues in else part:

First, a semaphore array is created and both are initialized (one for team A and one for team B). After that, a thread array is created with the size of ‘totalCount’ (which is the sum of inputs), and an integer array is created (fans[2]) with the size of 2, and initialized with integers 0. This array will store the number of fans, who are waiting to find a car. ($0^{th}$ index is for team A fans and $1^{st}$ for team B fans)

## `struct threadArgs:`

Before creating threads with the function ‘findRide()’, the arguments that will be passed to the function must be prepared. At this point, I have created a struct named ‘threadArgs’ which will store some variables and fulfill this purpose. A ‘threadArgs’ object will store 3 variables: a semaphore pointer that will point the main semaphores array, an integer pointer that will point the fans array, and an integer that will take 0 or 1 value only, and will determine which team’s fan the current thread is running for. I used this struct, and this method, because I did not want to use global variables except a global mutex lock named.

## `Initializing threads:`

After creating ‘threadArgs’ objects for both team, a while loop will run ‘totalCount’ times for initializing threads. An if condition checks for which team’s fan the current thread will run for.
For example: If main input numbers are 2 and 4. The while loop will run for 6 times, and first 2 loop will initialize a thread for team A, and rest 4 will initialize for team B. Both team’s threads will be initialized with ‘findRide()’ function and that team’s ‘threadArgs’ object is passed.

After all threads initialized, a while loop runs for totalCount times again for waiting all threads to terminate and join. After all, “The main terminates” is printed and both semaphores in ‘sems’ array are destroyed.

## `void* findRide(void* arg):`

In findRide() function, arguments are casted to ‘struct threadArgs’ first. Then, essential variables are declared, and globalMutex is locked for preventing other threads to interfere. Current thread’s ID is printed in proper format (...looking for a car...), and current team’s fan counter looking for car is incremented by one. At this point, an if block checks for placing fans in cars with proper order:

> -If current team’s waiting fan count is at least 2 and other team’s waiting fan count is 2, in other words there are enough people to form a 2 v 2 car, 1 current team’s fan thread is woken up (and also one fan already comes from current thread), and 2 other team’s fan thread is woken up. Both team’s waiting fan counters are decremented by 2 and isFull is set to 1.
>
> -Else if current team’s waiting fan count is 4, in other words there are enough people to form 4 fan from same team car, 3 current team’s fan thread is woken up (and also one fan already comes from current thread). Current team’s waiting fan counter is decremented by 4 and isFull is set to 1.
>
> -If none of the conditions above is met, then it means that there are no possible scenario to form a car right now, thus globalMutex is unlocked for other threads to continue running, and current team’s semaphore is locked to be woken up later, when conditions met to form a proper car.

After if block, current thread’s ID is printed again with proper format (...found a car...), and if isFull is 1, which means this thread has formed a car and will drive this car, current thread’s ID is printed with proper format (...captain and driving…).