**2021-2022 Fall CS 307 Programming Assignment 3**

**Elif Cemre Durgut – 26493**

I used pthread semaphores and pthread barriers and coded two functions for fan threads, but the logic is the same. My algorithm satisfies the correctness points mentioned in the assignment document because I lock/unlock when I want to increment/decrement/change a variable, fan threads wait in sem\_wait until there is a valid combination. When that combination is found, the last thread (which makes the combination valid) wakes up the necessary number of threads. Threads wait at the barrier until the others are woken up as well. After that, they wait at the second barrier until everyone finds their spot. After this second barrier, the captain drives the car. In the end, the main thread waits for fan threads to join and then terminates. No deadlocks.

Global variables:

lock

a(# of waiting threads for team A)

b(# of waiting threads for team B)

two semaphores for team A and team B

two barriers (one is to find a spot, the other is to decide on the captain)

isDriving and isCaptain booleans to check

Fan thread:

print looking for a car

lock()

increment the # of waiting threads for that team(a++ or b++)

initialize the isDriving and isCaptain as zero

if there is no one who finds a spot (isDriving = 0), and there are either 4 A fans, or 4 B fans, or 2 A and 2 B fans:

isDriving = 1

if there are 4 A/B fans wake up all of them (sem\_post)

else wake up 2 A and 2 B fans (sem\_post)

decrement the # of waiting fans

unlock()

else:

unlock()

sem\_wait()

wait until there is enough fans to drive(4) (barrier)

print found a spot

wait until the car is full (barrier2)

choose a captain and print the captain

Main thread:

get the # of fans

if the # of fans are valid:

initialize the semaphores and barriers

create the threads

join the threads

destroy the semaphores and barriers

print main terminates