CISS 370

Project #1

Due 11:59pm, March 6, 2023

Write the following program using **any programming language** that supports threads and semaphores, e.g. ***C, C++, Java, python***:

Name:

Parking – Parking lot simulation.

Syntax:

Parking [*C A S P*]

Where

C - Number of Car threads
A- Random time between Car Arrivals
S- Number of parking Spaces.
P- Random time to Park.

If there are no arguments the program uses the following values:

C =2, A=5, S=1, P=20

Description:

The program creates ***C*** car threads, one every ***a*** seconds (***a*** is a random number between 1-***A***).
Each car thread parks ***p*** seconds (***p*** is a random number between 1-***P***) in a parking lot of ***S*** spaces and then exits.
If there is no empty space to park, the thread waits ***1*** second and then try again.

✓ **Each car thread displays:**

1. The **arrival time** and the number of **free parking spaces** upon its arrival.
2. Parking **space number** it obtained and the **time waited** to get it.
3. The **exit time** from the parking lot.

✓ **When all car threads exits the main thread displays:**

- **Parking Spaces history:**
  For each Parking Space **S** it displays the following information for each car C that parked in the space:
  1. **Car thread number** and
  2. **Parking time**.
- **Car Threads history:**
  For each Car Thread **C** it displays the following information:
  1. **Parking Space ,**
  2. **Arrival Time**,
  3. **Waiting Time** and
  4. **Parking Time**.

Example:

```
% Parking
Using default values:  3 Cars,   Rand Arrival  Time 3 Seconds
                       2 Space,  Rand Parking  Time 20 Seconds
... PARKING LOT OPEN ...
... Expecting (3) Cars with Random Arrival Time (3 seconds)
... Parking Spaces (2) with Random Parking Time (20 seconds)

>>> C1 ARRIVED after 1 seconds... Available Spaces: 2

>>> C1 Parked in S1 after waiting 0 seconds

>>> C2 ARRIVED after 2 seconds... Available Spaces: 1

>>> C2 Parked in S2 after waiting 0 seconds

>>> C3 ARRIVED after 3 seconds... Parking is FULL

<<< C1 left S1 after parking 3 seconds

>>> C3 Parked in S1 after waiting 1 seconds

<<< C2 left S2 after parking 11 seconds

<<< C3 left S1 after parking 10 seconds

... PARKING LOT CLOSED ...  Number of Parked Cars 3

================== FINAL STATE ============================
```

```
Total Time from Open to Close: 14 seconds

Parking Spaces:
---------------
         Car      ParkingTime
S1:
         C1        3
         C3        10
S2:
         C2        11


Cars:
-----
         Space    ArrivalTime      WaitingTime      ParkingTime
C1:      S1       1                0                3
C2:      S2       2                0                11
C3:      S1       3                1                10
%
```

If your implementation includes condition variables, you will receive an additional 10% grade. In this case, instead of

> "If there is no empty space to park, the thread waits 1 second and then tries again,"

the wait() function should be called on the condition variable of the car thread. When one of the other car threads exits, it should signal the waiting car thread using the condition variable. It is important to note that the results should still show the arrival time, waiting time, and parking time for each car thread, even if a condition variable is used for the thread waiting to park.