List of Semaphores:

- roomInLine:
  - Initial value: 4
  - Purpose: serve as the agent line limit of 4 customers
- agentAvailable:
  - Initial value: array of 0's (size of 2)
  - Purpose: serve as an indicator of availability of the agent(s)
- mutex1:
  - Initial value: 1
  - Purpose: serve as mutual exclusion for enqueue and dequeue
- mutex2:
  - Initial value: 1
  - Purpose: serve as mutual exclusion for enqueue and dequeue
- customerEntered:
  - Initial value: 0
  - Purpose: serve as an indication of customer entering the DMV
- numbered:
  - Initial value: 0
  - Purpose: serve as an indication of information desk assigning the customer a number
- waiting:
  - Initial value: 0
  - Purpose: let customer know to the announcer they're in the waiting room
- called:
  - Initial value: 0
  - Purpose: for info desk to call customer to move into the agent line
- waitLine:
  - Initial value: 0
  - Purpose: for customer to let annoncer they're in the agent line
- inLine:
  - Initial value: 0
  - Purpose: for announcer to let agent know a new customer is in the agent line
- serveCustomer:
  - Initial value: 0
  - Purpose: for agent to serve the customer
- beingServed:
  - Initial value: 0
  - Purpose: for customer to let agent know they're receiving service
- agentAsksExam
  - Initial value: 0
  - Purpose: for agent to tell customer to take photo and eye exam
- customerTakesExam
  - Initial value: 0
  - Purpose: for customer to finish taking photo and eye exam for agent

- giveLicense:
    - Initial value: 0
    - Purpose: for agent to give customer license
- getsLicense:
    - Initial value: 0
    - Purpose: for customer to receive license and depart
- joined:
    - Initial value: array of 0's (size of 20)
    - Purpose: serve as an indicator of whether or not each of 20 customers received license or not

Pseudocode:

- Semaphore roomInLine = 4
- Semaphore[2] agentAvailable = {0}
-
- Semaphore mutex1 = 1
- Semaphore mutex2 = 1
-
- Semaphore customerEntered = 0
- Semaphore numbered = 0
- Semaphore waiting = 0
- Semaphore called = 0
- Semaphore waitLine = 0
- Semaphore inLine = 0
- Semaphore serveCustomer = 0
- Semaphore beingServed = 0
- Semaphore agentAsksExam = 0
- Semaphore customerTakesExam = 0
- Semaphore giveLicense = 0
- Semaphore getsLicense = 0
-
- Queue entry = linked list
- Queue waitingRoom = linked list
- Queue line = linked list
- Queue served = linkedlist
-
- Semaphore joined[20] = {0}
-
- Void information_desk() {
    - Int num = 0
    -
    - While (true) {
        - wait(customerEntered)
        - // Critical section
        - wait(mutex1)

- giveNumber() // same as enqueue()
- ++num
- signal(mutex1)
- //
- signal(numbered)
- }
- }
-
- Void announcer() {
- While (true) {
- wait(roomInLine)
-
- wait(waiting)
- // Critical section
- wait(mutex2)
- call()
- signal(mutex2)
- //
- signal(called)
-
- wait(waitLine)
- // Critical section
- wait(mutex1)
- putInLine()
- signal(mutex1)
- //
- signal(inLine)
- }
- }
-
- Void agent() {
- Int agentID
- Int customer
-
- While (true) {
- wait(inLine)
- // Critical section
- wait(mutex1)
- serve()
- signal(mutex1)
- //
- signal(serveCustomer)
-
- wait(beingServed)

```
                - // Critical section
                - wait(mutex1)
                - askExam()
                - signal(mutex1)
                - //
                - signal(agentAsksExam)
                -
                - wait(customerTakesExam)
                - // Critical section
                - wait(mutex1)
                - giveLicense()
                - signal(mutex1)
                - //
                - signal(giveLicense)
                -
                - wait(getsLicense)
                - signal(finished[customer])
                -
                - wait(agentAvailable[agentID])
                - signal(roomInLine)
        - }
- }
-
- Void customer() {
        - Int customerID
        - Int number
        - Int agent
        -
        - wait(mutex1)
        - enter()
        - signal(mutex1)
        -
        - signal(customerEntered)
        -
        - wait(numbered)
        - // Critical section
        - signal(waiting)
        -
        - wait(called)
        - // Critical section
        - signal(waitLine)
        -
        - wait(serveCustomer)
        - // Critical section
```

- wait(mutex1)
- beingServed()
- signal(mutex1)
- //
- signal(beingServed)
-
- wait(agentAsksExam)
- // Critical section
- wait(mutex1)
- takeExam()
- signal(mutex1)
- //
- signal(customerTakesExam)
-
- wait(giveLicense)
- // Critical section
- wait(mutex1)
- signal(mutex1)
- //
- signal(getsLicense)
-
- wait(finished[customerID])
- // Critical section
- wait(mutex1)
- finish()
- signal(mutex1)
- //
- signal(agentAvailable[agent])

- }