EE 360P, Sample TEST 1 Problems, Vijay K. Garg

NAME:

UT EID:

Q. **(10 points)** Consider the following algorithm proposed by a programmer for mutual exclusion for two processes. The algorithm uses two shared boolean bits called $flag[0]$ and $flag[1]$.

```
shared boolean flag[0] = false;  shared boolean flag[1] = false;

Program for Process P0

1.  while (flag[1] ) no-op();
2. flag[0] := true;
3.  Critical Section
4. flag[0] := false;

Program for Process P1

1. flag[1] := true;
2.  while (flag[0] ) no-op();
3.  Critical Section
4. flag[1] := false;
```

(a) Does this algorithm satisfy mutual exclusion? Why or why not? (You will not get credit without proper justification).

(b) Is the algorithm correct if the lines 1 and 2 are swapped for Process P0? Why or why not? (You will not get credit without proper justification).

Q. **(22 points)** Write a Java class `FifoLock` that provides the following methods `getTicket(), requestCS(int ticketnumber)` and `releaseCS()`. When any thread invokes `getTicket()`, it gets a ticket number. You should ensure that no two ticket numbers are identical. After getting a ticket number, the thread can invoke `requestCS` anytime. It must provide the ticket number that it received using `getTicket`. Assume that threads do not cheat and provide the genuine ticket number. You should guarantee that threads enter the critical section in the order of their ticket numbers and two threads are never in the critical section simultaneously.

Q. **(23 points)** An old bridge has only one lane and can only hold at most 4 cars at a time without risking collapse. Create a Java monitor with methods `arriveBridge(int direction)` and `exitBridge()` that controls traffic so that at any given time, there are at most 4 cars on the bridge, and all of them are going the same direction. A car calls `arriveBridge` when it arrives at the bridge and wants to go in the specified direction (0 or 1); `arriveBridge` should not return until the car is allowed to get on the bridge. A car calls `exitBridge` when it gets off the bridge, potentially allowing other cars to get on. Don't worry about starving cars trying to go in one direction; just make sure cars are always on the bridge when they can be. You may use Java wait, notify, notifyall, or ReentrantLock and Condition variables.

Q. **(20 points)** Implement the class `multiLock` to coordinate entry into a set of rooms by various threads. There are k different rooms numbered 1 through k. A thread requests entry into a room by explicitly specifying the room number via the method `request(int roomNumber)`. It exits the room by calling `release(int roomNumber)`. The constraints are as follows:
1) There can be at most one thread in room 1; other rooms can have any number of threads.
2) At most one room is occupied at any given time.

Implement the class `multiLock` with methods `void request(int roomNumber)`, and `void release(int roomNumber)`. You do not have to worry about starvation.