1) Interrupts are not appropriate to implement synchronization primitives in a multiprocessor system for a variety of reasons. Firstly if the interrupts are on a timer then nested interrupts can heavily delay tasks from running possibly ruining the synchronization. If the interrupts are not on a timer then the unpredictability on when they will occur can also ruin synchronization. Mostly they are also more complex to make and account for increasing complexity in design and in debugging.
2) A race condition might be possible if the two methods were to see the current condition of the bank account at one time and change it without regard to the other, say 1 thread adding $5 to $20 and the other removing $5 dollars from $20. These 2 threads might now disagree on the current amount in the bank account ($25 vs $15). To prevent this a mutex lock might be used so that only one of the threads does its work at a time.
3)
    a) 5 Processes are being made considering a thread as a subset of a process
    b) 1 thread is being created
4)
    a)
        i) Mutual exclusion - each car is waiting for the next lane (resource) to be free before continuing
        ii) Hold and wait - each car is holding its position in its lane while waiting for the next to be free
        iii) No preemption - the cars took their spot in a lane without knowing if the next lane was free or not
        iv) Circular wait - each car is waiting for a lane which is waiting for a lane which is waiting for a lane which is waiting for the first lane again
    b) Looking at the other lanes that you will cross before going into your lane would solve the deadlock. This is known as implementing preemption
5) There is a deadlock in the graph that can be resolved by adding one resource to R3. If that resource is added the sequence of processes would be as follows.
    a) P3 would run freeing resources in R1 and R4
    b) P1 runs
    c) P4 runs freeing resources in R2
    d) P2 runs