1. a) The 4 conditions for deadlock are present in the figure:

- Mutual exclusion: Only one car can occupy the space on the road.
- Hold-and-wait: A car holds onto its place on the road while waiting to move forward.
- No preemption: A car cannot be removed from its space on the road.
- Circular wait: A circular pattern can be observed from the figure as the cars form a circle around the middle block.

1. b) A simple rule to avoid this situation is that a car is not allowed to move into the intersection if it cannot immediately continue past the intersection.

2. a) From section *8.7.2 Several Instances of a Resource Type* of the book Operating System Concepts (10th edition) by Silberschatz et al., we have:

a. $\sum_{i=1}^{n} Max_i \; < m + n$
b. $Max_i \; \geq \; i$ for all i

Proof: $Need_i \; = \; Max_i \; - \; Allocation_i$

If there is a deadlock state, then:

c. $\sum_{i=1}^{n} Allocation_i \; = \; m$

From a. and b., we get: $\sum Need_i \; + \; \sum Allocation_i \; = \sum_{i=1}^{n} Max_i \; < m + n$

Use c., we get: $\sum Need_i \; + m < m + n$

Therefore: $\sum_{i=1}^{n} Need_i \; < n$

This means that there exists a process P_i such that $Need_i \; = \; 0$. Since $Max_i \; \geq \; 1$, it follows that P_i has at least one resource it can release. Thus, the system cannot be in a deadlock state.

3. a) Not safe. Processes P_2 , P_1, and P_3 can finish, but the remaining processes cannot finish.

3. b) Safe. Processes P_1, P_2, P_3, P_4 and P_0 can finish.

4. a) P_2, P_3, P_0, P_1, P_4.

4. b) No, the request cannot be granted immediately and may create a deadlock.

4. c) Yes, the request can be granted immediately.

4. c) Yes, the request can be granted immediately.