Hunter Antal
1181729

Task 1 (Concept Questions, Total Marks: 30)

a) Interrupts can lead to race conditions because they don't ensure atomicity. Disabling interrupts only affects the current CPU and not others in a multiprocessor system, leading to potential data inconsistency.
b) A race condition occurs if the husband and wife access the shared account simultaneously. For example, if the husband withdraws while the wife deposits, inconsistent balance updates may result. Use locks (e.g., mutex) to prevent this by ensuring only one method (withdraw or deposit) can execute at a time.
c) 1. 3 processes are created
   2. 1 thread is created by thread_create();

Task 2 (Concept Questions, Total Marks: 10)

a) i)
   1. Mutual exclusion: Each car holds a portion of the road, preventing others from accessing it.
   2. Hold and wait: Every car holds its current position and waits to move forward, but it can't because of the vehicle ahead.
   3. No preemption: The cars cannot be forced to back up or leave their current positions.
   4. Circular wait: There is a cycle of cars, each waiting for the car in front, creating a circular dependency.
   ii) Mutual Exclusion: Use one-way streets to break the circular wait condition, ensuring traffic flows in only one direction at intersections.
b) i) Yes, there is a deadlock.
   - Processes involved: P1, P2, P3, P4
   - Resources involved: R1, R2, R3, R4

Yes, we can resolve the deadlock by adding more resources. If each process can get the resource it’s waiting for, the cycle will break, allowing processes to complete. If we added another R1, P1 would run, freeing R3 for P3. P3 would run, freeing R1 and R4. P4 would then run, freeing R2 for P2 to run then.