CS 162 Section 9

True/False:

1. Serial schedules are necessary to preserve ACID transaction semantics
   False. The schedule just needs to be semantically equivalent to a serial schedule

Short Answer:

1. What does ACID stand for? Explain each of them.

   Atomicity: all actions in the transaction happen, or none happen
   Consistency: if each transaction is consistent, and the database starts consistent, it ends up consistent, e.g.,
   – Balance cannot be negative
   – Cannot reschedule meeting on February 30
   Isolation: execution of one transaction is isolated from that of all others
   Durability: if a transaction commits, its effects persist

2. What are some elements you might want to lock in a database?

   Row, Table, Database, Page, Predicates (In theory, never done in practice), Ranges

3. What are types of possible conflicts in an execution of multiple transactions?

   Read-Write conflict (Unrepeatable reads)
   Write-read conflict (reading uncommitted data)
   Write-write conflict (overwriting uncommitted data)

4. What are the requirements for two transaction operations to conflict?

   Belong to different transactions
   Are on the same data
   At least one of them is a write

5. Two schedules are conflict equivalent iff:

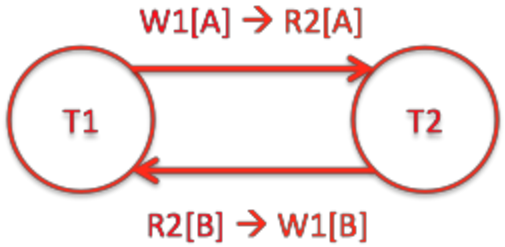
   Involve the same operations of the same transactions
   Every pair of conflicting operations is ordered the same way

Long Answer:

1. Consider the following two transactions and schedule (time goes from left to right). Is this schedule conflict-serializable? Explain why or why not.

T1:	R1[A]	W1[A]				R1[B]	W1[B]
T2:			R2[A]	R2[B]			

The schedule is not conflict serializable because the precedence graph contains a cycle. The graph has an edge T1 → T2 because the schedule contains W1[A] → R2[A]. The graph has an edge T2 → T1 because the schedule contains R2[B] → W1[B].



2. Consider a database with objects X and Y and assume that there are two transactions T1 and T2. T1 first reads X and Y and then writes X and Y. T2 reads and writes X then reads and writes Y. Give an example schedule that is not serializable. Explain why your schedule is not serializable.

T1:	R1[X]	R1[Y]				W1[X]	W1[Y]	
T2:			R2[X]	W2[X]	R2[Y]			W2[Y]

A schedule is serializable if it contains the same transactions and operations as a serial schedule and the order of all conflicting operations (read/writes to the same objects by different transactions) is also the same. In the above schedule, T1 reads X before T2 writes X. However, T1 writes X after T2 reads and writes it. The schedule is thus clearly not serializable.

Additionally, according to the above schedule, the final content of object X is written by T1 and the final content of object Y is written by T2. Such a result is not possible in any serial execution, where transactions execute one after the other in sequence.