1. A history is *serializable* if its effects on the database are the same as that of some serial history, regardless of what the initial state of the database is. Consider each of the following histories:

(a) $r_1[A], r_2[A], r_3[B], c_3, w_1[A], r_2[B], w_2[B], c_2, w_1[C], c_1$
(b) $r_1[A], r_2[A], w_1[B], w_2[B], r_1[B], r_2[B], w_2[C], c_2, w_1[D], c_1$

Answer the following questions about each of the above histories.

(a) Draw the serialization graph for the history.
(b) Is the history conflict-serializable? If so, what are all the equivalent serial histories?
(c) Is the history view-serializable? Explain.
(d) Is the history serializable according to the definition given above?

2. A transaction $T_i$, executed by an airline reservation system, performs the following steps:

(a) The customer is queried for a desired flight time and cities. Information about the desired flights are located in database elements A and B, which the system retrieves from disk.
(b) The customer is told about the options, and selects a flight whose data, including the number of reservations for that flight is in B. A reservation on that flight is made for the customer.
(c) The customer selects a seat for the flight; seat data for the flight is in database element C.
(d) The system gets the customer’s credit-card number and appends the bill for the flight to a list of bills in database element D.
(e) The customer’s phone and flight data is added to another list on database element E for a fax to be sent confirming the flight.

Answer the following questions:

(a) Express transaction $T_i$ as a series of $r$ and $w$ operations.

(b) If the transaction manager had available to it read, write, update, and increment locks, what lock would you recommend for each of the steps of the transaction?

3. The action of multiplication by a constant factor can be modeled by an action of its own. Suppose $MC(X, c)$ stands for an atomic execution of the steps $READ(X, t); t := c \times t; WRITE(X, t)$; where $X$ is a data item and $t$ is a temporary variable used by the transaction. We can also introduce a lock mode that allows only multiplication by a constant factor.

(a) Show the compatibility matrix for read, write, and multiplication-by-a-constant lock.

(b) Show the compatibility matrix for read, write, incrementation, and multiplication-by-a-constant locks.

4. Here is a history with one action missing:

\[ r_1[A], r_2[B], r_3[C], w_1[A], w_2[B] \]

Your problem is to figure out what actions of certain types could replace the and make the history not be conflict serializable. Tell all possible non-serializable replacements for each of the following types of operation:

(a) Read operation
(b) Write operation
(c) Update operation
(d) Increment operation

5. Consider the following history:

\[ r_1[A], r_2[B], r_3[C], r_1[B], r_2[C], r_3[D], w_1[A], w_2[B], w_3[C] \]

Do each of the following:
(a) Insert read and write locks, and insert unlock actions. Place a read lock immediately in front of each read operation that is not followed by a write operation of the same data item by the same transaction. Place a write lock in front of every other read or write operation. Place the necessary unlocks at the end of every transaction.

(b) Tell what happens when each history is run by a scheduler that supports read and write locks.

(c) Insert read and write locks in a way that allows upgrading. Place a read lock in front of every write, and place the necessary unlocks at the ends of the transactions.

(d) Tell what happens when the history from 5c is run by a scheduler that supports read locks, write locks, and upgrading.

(e) Insert read, write, and update locks, along with unlock operations. Place a read lock in front of every read operation that is not going to be upgraded, place an update lock in front of every read operation that will be upgraded, and place an exclusive lock in front of every write operation. Place unlocks at the ends of transactions, as usual.

(f) Tell what happens when the history from 5e is run by a scheduler that supports read, write, and update locks.

6. For the following history, insert appropriate locks (read, write, or increment) before each operation, and unlocks at the ends of transactions. Then tell what happens when the history is run by a scheduler that supports these three types of locks.

\[ r_1[A], r_2[B], inc_1[B], inc_2[A], w_1[C], w_2[D] \]

7. Look at the architecture of the locking scheduler given in Lecture 3. What are suitable group modes for a lock table if the lock modes used are:

(a) read and write locks
(b) read, write and increment locks

8. Consider an object-oriented database. The objects of class C are stored on two blocks, B1 and B2. Block B1 contains objects O1 and O2, while block B2 contains objects O3, O4, and O5. Classes, blocks, and objects form a hierarchy of lockable data items. Tell the sequence of lock requests and the response of an intention-based scheduler to the following sequences of requests. You may assume all requests occur just before they are needed, and all unlocks occur at the end of the transaction.

\[ r_1[O_1], w_2[O_2], r_2[O_2], w_1[O_4] \]
9. Show how to add increment locks to an intention-based scheduler.

10. Below is a sequence of events, including start events, where \( st_i \) means that transaction \( T_i \) starts. These sequences represent real time, and the timestamp-based scheduler will allocate timestamps to transactions in the order of their starts. Tell what happens as the sequence executes.

(a) \( st_1, st_2, r_1[A], r_2[B], w_2[A], w_1[B] \)

11. Tell what happens during the following sequence of events if a multiversion, timestamp-based scheduler is used. What happens instead, if the scheduler does not maintain multiple versions?

\( st_1, st_2, st_3, st_4, w_1[A], w_2[A], w_3[A], r_2[A], r_4[a] \)

12. Can a timestamp-based scheduler having a commit bit \( C(X) \) have a deadlock? Explain.


14. Two transactions are not interleaved in a history if every operation of one transaction precedes every operation of the other. Give an example of a serializable history \( H \) that has all of the following properties:

(a) transactions \( T_1 \) and \( T_2 \) are not interleaved in \( H \),

(b) \( T_1 \) precedes \( T_2 \) in \( H \), and

(c) in any serial history equivalent to \( H \), \( T_2 \) precedes \( T_1 \)

15. Consider the following history \( H \)

\[ r_2[y], r_1[x], r_3[y], r_2[x], w_2[y], c_2, w_1[x], c_1, r_3[x], c_3 \]

(a) Assuming that each transaction is consistent, does the final database state satisfy all the integrity constraints? Explain.