1.

a)

Deadlock is a state that in which both transactions must to wait for other to unlock the requested locks, neither transaction can continue.

Rollback is an operation that returns some set of records in database to some previous state, usually to previous commit point.

Blind write is an action that a transaction writes a value to an object without reading the value of that object.

Unrecoverable schedule is a schedule(an abstract model describing execution of transactions running) that in which a transaction reads values from uncommitted transaction and commit it.

Phantom problem occurs when a transaction has two identical reads and there is an insertion between two reads. Inserted records will not be locked using 2PL protocol so that two reads may get different values.

Dirty read is an action that a transaction reads values from uncommitted transaction and commit it.

Unrepeatable read happens when a transaction has more than one read action for the same object and gets different values.

If two schedules are **View equivalent**, then they contain same transactions and those transactions read same value for each object in their schedules. Also, final write action on each data object should match in both schedules.

A serializable schedule is a non-serial schedule whose effect on a consistent database instances is guaranteed to be the same as some serial schedules.

If two schedules are **Conflict equivalent**, then one schedules can be transformed into another schedule by swapping non-conflicting operations.

2.

a) $R_1(A)\ R_1(B)\ W_1(A)\ W_2(A)\ W_1(B)\ R_2(B)\ W_2(B)$

b) $W_2(A)\ R_1(A)\ R_1(B)\ W_1(A)\ W_1(B)\ R_2(B)\ W_2(B)$

c) $W_2(B)\ R_1(A)\ W_1(A)\ R_3(A)\ W_3(A)\ R_1(B)\ W_1(B)\ R_2(D)\ W_2(D)\ W_3(D)$

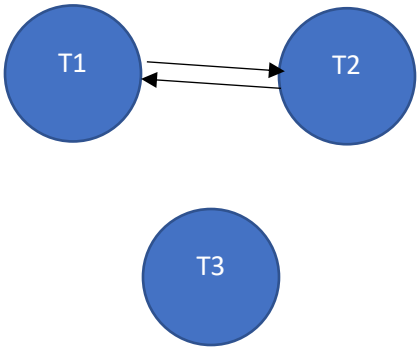
d) $R_1(A)\ R_1(B)\ W_2(A)\ W_1(A)\ W_1(B)\ W_3(A)$

e) $R_1(A)\ W_2(A)\ R_3(A)\ W_3(A)\ W_1(A)\ C_1\ C_2\ C_3$

f) $R_1(A)\ W_2(A)\ R_3(A)\ W_3(A)\ W_1(A)\ C_3\ C_1\ C_2$

3.

a)



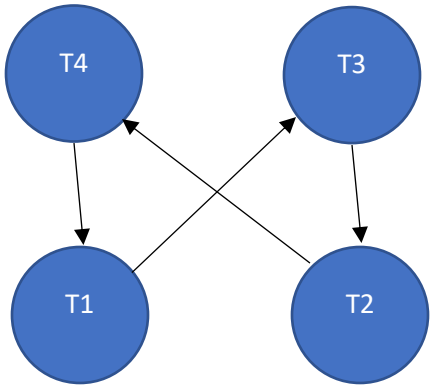
b) If T1 happens before T2, we cannot swap $R_2(C)$ with $R_1(C)$ to result in a serial schedule. If T2 happens before T1, we cannot swap $W_1(C)$ with $W_2(C)$ to result in a serial schedule.

c) $S_1(A)$ $R_1(A)$ $S_3(A)$ $R_3(A)$ $U_3(A)$ $X_2(B)$ $R_2(B)$ $W_2(B)$ X2(C) $U_2(B)$ $R_2(C)$ T_1 suspended $W_2(C)$ $U_2(C)$ $X_1(C)$ $R_1(C)$ W1(C) $X_1(B)$ $U_1(A)$ $U_1(C)$ $R_1(B)$ $W_1(B)$ $U_1(B)$

d) $X_1(A)$ $R_1(A)$ $W_1(A)$ $X_2(C)$ $R_2(C)$ $W_2(C)$ $X_1(B)$ $R_1(B)$ $W_1(B)$ C_1 $U_1(A)$ $U_1(B)$ $S_2(A)$ $R_2(A)$ C_2 $U_2(C)$ $U_2(A)$ $S_3(B)$ $R_3(B)$ $S_3(C)$ $R_3(C)$ C_3 $U_3(B)$ $U_3(C)$

e) $X_1(A)$ $R_1(A)$ $W_1(A)$ $X_2(C)$ $R_2(C)$ $W_2(C)$ $S_3(B)$ $R_3(B)$ $X_1(B)$

f)



g) T_4

h) T_3 and T_4

4.

a) WT(B)<TS(T)<RT(B), write too late happens because transaction 1 tries to write B but transaction 2 has already read B first. So, T_1 will be aborted.

b) TS(T) < WT(X), read too late happens because transaction 1 tries to read B but transaction 2 has already written B first. So, T_1 will be aborted.

c) T_1 may read dirty data. Since T_1 reads data after written by T_2, but before T_2 commits, If T_2 aborts, the read by T_1 will be incorrect. So, T_2 will be suspended.

d) $TS(T_1) < TS(T_2)$ and T_1 tries to write A after T_2 writes B. Due to Tomas Write Rule, T_2's timestamp is later than T_1's meaning T_1 would have been over-written. So, $W_1(A)$ should be ignored.