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) R1(A) R1(B) W1­(A) W2­(A) W1(B) R2(B) W2­(B)

b) W2­(A) R1(A) R1(B) W1­(A) W1(B) R2(B) W2­(B)

c) W2­(B) R1(A) W1­(A) R3(A) W3­(A) R1(B) W1(B) R2(D) W2­(D) W3(D)

d) R1(A) R1(B) W2­(A) W1­(A) W1(B) W3(A)

e) R1(A) W2­(A) R3(A) W3(A) W1­(A) C1 C2 C3

f) R1(A) W2­(A) R3(A) W3(A) W­1(A) C3 C1 C2

3.

a)

b) If T1 happens before T2, we cannot swap R2(C) with R1(C) to result in a serial schedule. If T2 happens before T1, we cannot swap W1(C) with W2(C) to result in a serial schedule.

c) S1(A) R1(A) S3(A) R3(A) U3(A) X2(B) R2(B) W2­(B) X2(C) U2(B) R2­(C) T1 suspended W2­(C) U2(C) X1(C) R1(C) W1­(C) X1(B) U1(A) U1(C) R­1(B) W­1(B) U1(B)

d) X1(A) R1(A) W1(A) X2(C) R2(C) W2­(C) X1(B) R1(B) W1(B) C1 U1(A) U1(B) S2(A) R2(A) C2 U2­(C) U2­(A) S3(B) R3(B) S3(C) R3(C) C3 U3(B) U3(C)

e) X1(A) R1(A) W1(A) X2(C) R2(C) W2­(C) S3(B) R3(B) X1(B)

f)

g) T4

h) T3 and T4

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, T1 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, T1 will be aborted.

c) T1 may read dirty data. Since T1 reads data after written by T2, but before T2 commits, If T2 aborts, the read by T1 will be incorrect. So, T2 will be suspended.

d) TS(T1) < TS(T2) and T1 tries to write A after T2 writes B. Due to Tomas Write Rule, T­2’s timestamp is later than T1’s meaning T1 would have been over-written. So, W1(A) should be ignored.