Consider the following classes of schedules: serializable and 2PL. For each of the following schedules, state which of the preceding classes it belongs to. If you cannot decide whether a schedule belongs in a certain class based on the listed actions, explain briefly.

The actions are listed in the order they are scheduled and prefixed with the transaction name. If a commit or abort is not shown, the schedule is incomplete; assume that abort or commit must follow all the listed actions.

1. T1:R(X), T2:R(Y), T3:W(X), T2:R(X), T1:R(Y)
2. T1:R(X), T1:R(Y), T1:W(X), T2:R(Y), T3:W(Y), T1:W(X), T2:R(Y)
3. T1:W(X), T2:R(X), T1:W(X), T2:Commit, T1:Commit
4. T1:R(X), T2:W(X), T1:W(X), T3:R(X), T1:Commit, T2:Commit, T3:Commit

Solution:

1. The serialization graph is T1 → T3 → T2. Since the serialization graph does not have any cycle, the schedule is serializable. It is also 2PL. One possible locking sequence for this schedule that is compatible with the 2PL rule is as follows. In the following schedule, SLock, SRelease, XLock, and XRelease denote acquiring a shared lock, releasing a shared lock, acquiring an exclusive lock, and releasing an exclusive lock, respectively.

T1:SLock(X), T1:SLock(Y), T1:R(X), T1:SRelease(X),
T2:SLock(Y), T2:R(Y),
T3:XLock(X), T3:W(X), T3:XRelease(X),
T2:SLock(X), T2:R(X), T2:SRelease(X),
T2:SRelease(Y),
T1:R(Y), T1:SRelease(Y)

2. The serialization graph is:

Since the serialization graph has a cycle between T2 and T3, the schedule is not serializable. Because the schedule is not serializable, it is not a 2PL schedule.

3. The serialization graph is:

Because the serialization graph has a cycle between T1 and T2, the schedule is not serializable. Thus, it is not a 2PL schedule.
4. The serialization graph is:

![Serialization Graph](image)

Because the serialization graph has a cycle between T1 and T2, the schedule is **not**
serializable. Hence, it is not 2PL.

---

**2: Multi-granularity Locking (1 point)**

Consider a database DB with relations R1 and R2. The relation R1 contains tuples t1 and t2 and
the relation R2 contains tuples t3, t4, and t5. Assume that the database DB, relations, and tuples
form a hierarchy of lockable database elements. Explain the sequence of lock requests and the
response of the locking scheduler to the following schedule. You may assume all lock requests
occur just before they are needed, and all unlocks occur at the end of the transaction, i.e., EOT.

- T1:R(t1), T2:W(t2), T2:R(t3), T1:W(t4)

**Solution:** At the first step, T1 puts a IS lock on the DB and on R1, and an S lock on t1. At step
2, T2 puts an IX lock on the DB and on R1, both of which are compatible with the IS locks
already there. T2 also puts an X lock on t2. At step 3, T2 puts an IS lock on R2 and an S lock on
t3, then releases its locks. In this step, T2 did not need to put a IS lock on DB because it already
has an IX lock on DB (from step 2). At step 4, T1 puts an IX lock on the DB and on R2 and an
X lock on t4, then releases its locks.

---

**3: Degrees of Consistency (1 points)**

What are the maximum degrees of consistency for T1, T2, and T3 in the following schedule.
Explain your answer.

- T1:R(X), T1:R(Y), T1:W(X), T2:R(Y), T3:W(Y), T1:W(X), T2:R(Y)

**Solution:** The maximum degree of consistency of T1 is 2. T1 is degree 0 because it only writes
on X, and no other transactions dirty data on X. T1 is degree 1 because it can set a write lock on
X and release it until EOT (after commit). T1 is degree 2 because it reads X and Y before other
transactions write on them. T1 is NOT degree 3 because it reads Y, then T3 writes on Y before
T1 has completed.

The maximum degree of consistency of T2 is 2. T2 is degree 0 and 1 because it does not write.
T2 is degree 2 because it can set short shared locks on Y. Notice that T3 writes on Y and then
the transaction ends, so it commits. Then Y is not dirty once T3 reads it again. T2 is NOT
degree 3 because it may read two different values of Y, as T3 writes on Y. T2 cannot set a long
shared lock on Y because T3 needs to set an exclusive lock on Y.

The maximum degree of consistency of T3 is 3. T3 is degree 0 because it can set an exclusive lock
on Y and then write on Y. T3 is degree 1 because it can set an exclusive lock on Y, then write on
Y, then commit and release the lock on Y. T3 is degree 2 and 3 because it does not read.