21.9 What is a serial schedule? What is a serializable schedule? Why is a serial schedule considered correct? Why is a serializable schedule considered correct?

**Serial Schedule**.
A schedule "S" is referred as serial, for each transaction "T" participating in schedule, the operations of T must be executed consecutively in schedule. So from this perspective, it is clear that only one transaction at a time is active and whenever if that transaction is committed, then it initiates the execution of next transaction.

**Serializable schedule**.
The schedule is referred as "serializable schedule. When a schedule t T be a set of n transactions ($T_1$, $T_2$, …, $T_n$ ), is serializable and if it is equivalent to n transactions executed serially.
Consider that possibly there are "n" serial schedule of "n" transactions and moreover there are possibly non-serial schedules. If two disjoined groups of the nonserial schedules are formed then it is equivalent to one or more of the serial schedules. Hence, the schedule is referred as serializable.

**Reason for the correctness of serial schedule**.
A serial schedule is said to be correct on the assumption of that each transactions is independent of each other. So according to the "consistency preservation" property, when the transaction runs in isolation, it is executed from the beginning to end from the other transaction .Thus, the output is correct on the database.
Therefore a set of transaction executed one at a time is correct.

**Reason for the correctness of the serializable schedule**.
The simple method to prove the correctness of serializable schedule is that to prove the satisfactory definition. In this definition, it compares the results of the schedules on the database, if both produce same final state of database. Then, two schedules are equivalent and it is proved to be serializable. Therefore, the serializable schedule is correct when the two schedules are in the same order.

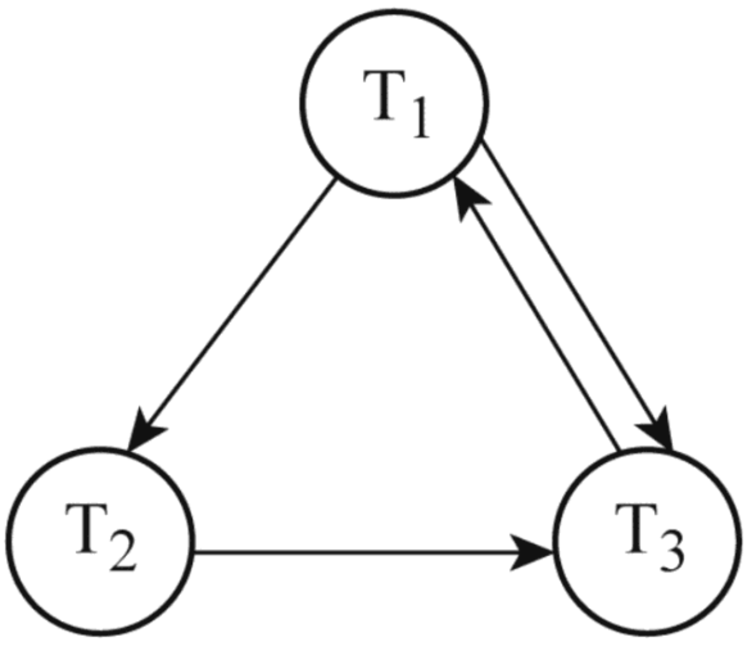
21.22 Which of the following schedules is (conflict) serializable? For each serializable schedule, determine the equivalent serial schedules.

**Serializable schedule**.
A conflict graph corresponding to a schedule decides whether given schedule is conflict serializable or not. If conflict graph contains cycle, then the schedule is not serializable. The drawing sequence of conflict graph

1) Create a node labeled $T_i$ in graph for each of the transaction $T_i$ which participates in schedule S.
2) An edge is created from $T_i$ to $T_j$ in graph, where a write_item(X) is executed by $T_i$ and then a read_item(X) is executed by $T_j$
3) Create an edge in graph from $T_i$ to $T_j$ , where a read_item(X) is executed by $T_i$ and then a write_item(X) is executed by $T_j$
4) Create an edge in graph from $T_i$ to $T_j$ , where a write_item(X) is executed by $T_i$ and then a write_item(X) is executed by $T_j$.
5) If no cycles are present in conflict graph, then it is a serializable schedule.

a. r1(X); r3(X); w1(X); r2(X); w3(X);

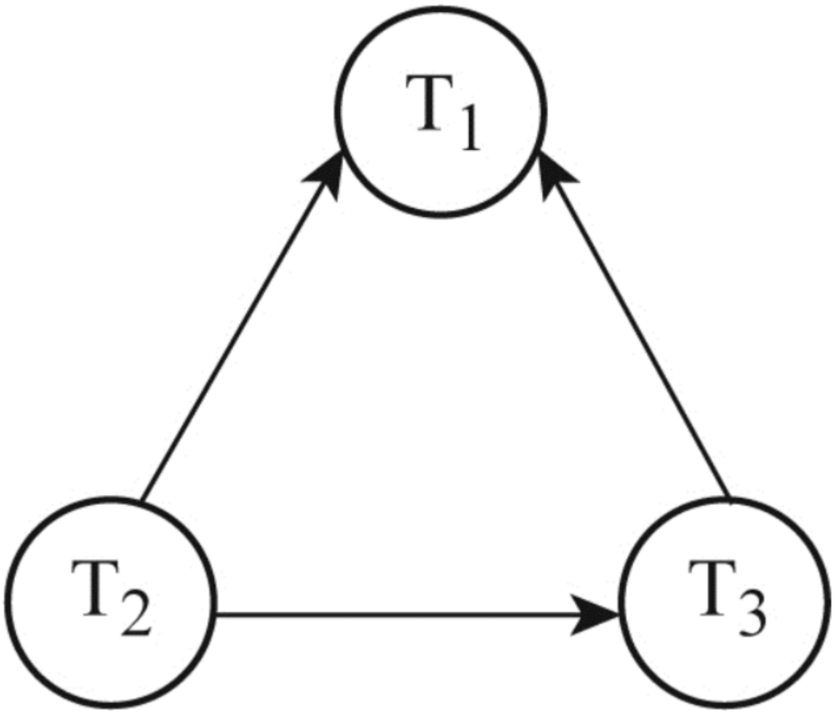
Conflict graph:



The conflict graph has cycle, in $T_1$-$T_3$. Hence, given schedule is $\boxed{\text{not serializable}}$ .

c. r3(X); r2(X); w3(X); r1(X); w1(X);

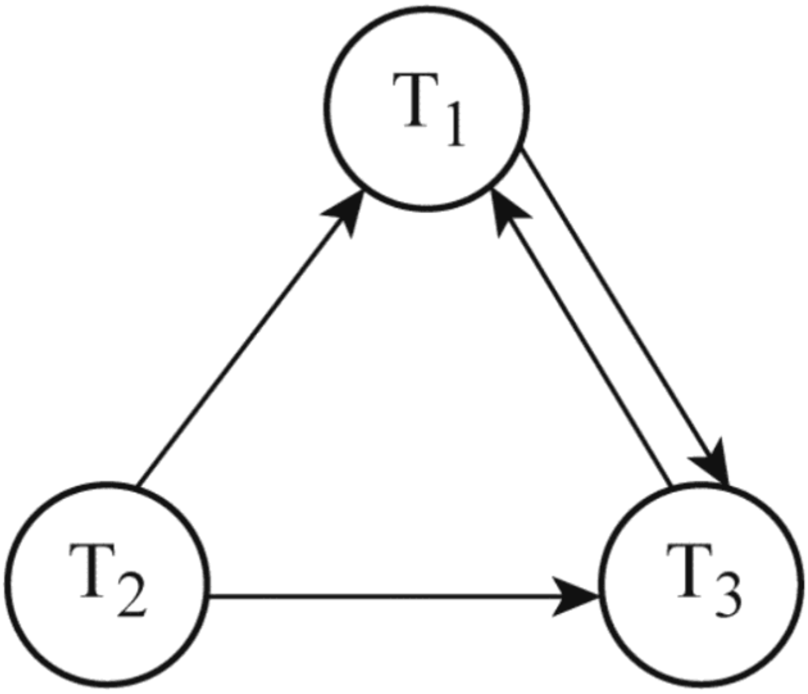
**Conflict graph:**



The graph contains no cycles. Hence, the schedule S is $\boxed{\text{serializable}}$ .

d. r3(X); r2(X); r1(X); w3(X); w1(X);

**Conflict graph:**



The conflict graph has cycle, in $T_1$-$T_3$. Hence, the schedule S is $\boxed{\text{not serializable}}$.

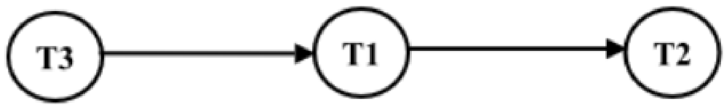
21.23 Consider the three transactions T1, T2, and T3, and the schedules S1 and S2 given below. Draw the serializability (precedence) graphs for S1 and S2 and state whether each schedule is serializable or not. If a schedule is serializable, write down the equivalent serial schedule(s).
T1: r1(X); r1(Z); w(X);
T2: r2(Z); r2(Y); w12(Z); w(Y);
T3: r3(X); r3(Y); w(Y);
S1: r1(X); r2(Z); r1(Z); r3(X); r3(Y); w1(X); w3(Y); r2(Y); w2(Z); w2(Y);
S2: r1(X); r2(Z); r3(X); r1(Z); r2(Y); r3(Y); w1(X); w2(Z); w3(Y); w2(Y);

## **<u>The schedule S1 is as follows:</u>**

S1: r1(X); r2(Z); r1(Z); r3(X); r3(Y); w1(X); w3(Y); r2(Y); w2(Z); w2(Y);

| T1 | T2 | T3 |
|---|---|---|
| r1(X) | | |
| | r2(Z) | |
| r1(Z) | | |
| | | r3(X) |
| | | r3(Y) |
| w1(X) | | |
| | | w3(Y) |
| | r2(Y) | |
| | w2(Z) | |
| | w2(Y) | |

The precedence graph for S1 is as follows:

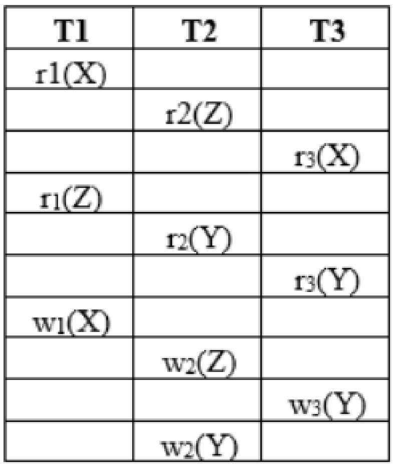

$$\begin{tikzcd} T3 \arrow[r] & T1 \arrow[r] & T2 \end{tikzcd}$$

The schedule S1 is a serializable schedule as there is no cycle in the precedence graph.
• T3 reads X before X is modified by T1.
• T1 reads Z before Z is modified by T2.
• T2 reads Y and writes it only after T3 has written to it.
The equivalent serializable schedule is as follows:

| T1 | T2 | T3 |
|---|---|---|
| r1(X) | | |
| | | r3(X) |
| | r2(Z) | |
| r1(Z) | | |
| | | r3(Y) |
| | | w3(Y) |
| | r2(Y) | |
| w1(X) | | |
| | w2(Y) | |
| | w2(Z) | |

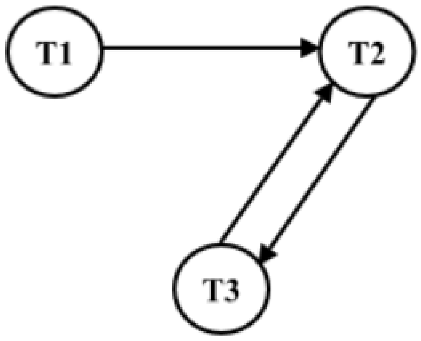
## The schedule S2 is as follows:

S2: r1(X); r2(Z); r3(X); r1(Z); r2(Y); r3(Y); w1(X); w2(Z); w3(Y); w2(Y);



| T1 | T2 | T3 |
|---|---|---|
| r1(X) | | |
| | r2(Z) | |
| | | $r_3(X)$ |
| $r_1(Z)$ | | |
| | $r_2(Y)$ | |
| | | $r_3(Y)$ |
| $w_1(X)$ | | |
| | $w_2(Z)$ | |
| | | $w_3(Y)$ |
| | $w_2(Y)$ | |

The precedence graph for S2 is as follows:



$$\begin{tikzcd}
T1 \arrow[rr] & & T2 \arrow[dl, bend left] \\
& T3 \arrow[ur, bend left] &
\end{tikzcd}$$

The schedule S2 is not a serializable schedule as there is cycle in the precedence graph.
• T2 reads Y before T3 reads it and modifies Y.
• T3 reads Y which is later modified by T2.