Part 1: Schedules and Anomalies (10 points)

Consider a database with objects X, Y, and Z and assume that there are two transactions T1 and T2 that attempt the following operations.

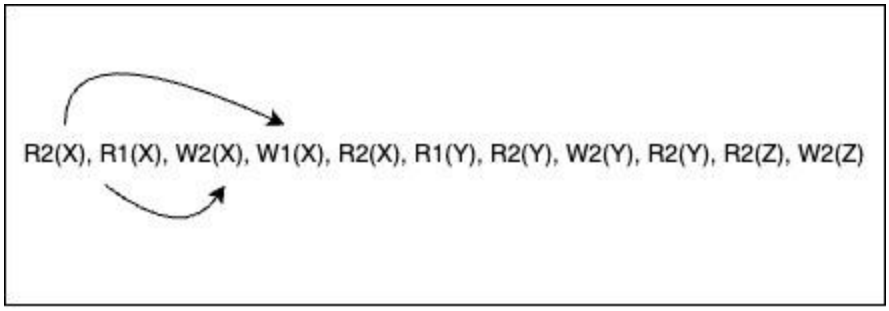
T1: R(X), R(Y), W(X)

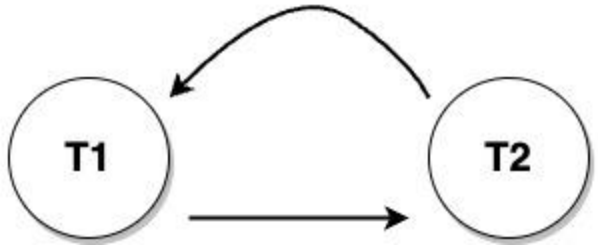
T2: R(X), R(Y), W(Y), R(X), R(Y), W(X), R(Z), W(Z)

a.) Write an example schedule that interleaves operations between T1 and T2, that is NOT conflict serializable.

T1	T2
	R(X)
R(X)	
	W(X)
W(X)	
	R(X)
R(Y)	
	R(Y)
	W(Y)
	R(Y)
	R(Z)
	W(Z)

As shown below, we write this schedule in the “in-line” format as well as show the precedence graph for this schedule. It is important to note that I showed the first occurrence of each conflict (i.e. the conflict from node 1 to node 2 and node 2 to node 1), however, I didn’t show all possible conflicts as that would be redundant. The first conflict shown below is a read-write conflict on the X component between transaction 2 and transaction 1. The second conflict shown below is a read-write conflict on the X component between transaction 1 and transaction 2.





As seen above, since there is a cycle in the precedence graph, this schedule is not conflict serializable. Hence we've gotten the result we wanted.

b.) If T1 is instead just "R(X)", this corresponds to T1 just being a single query like

```
SELECT * FROM Flights WHERE id=1024;
```

Should the database treat a single SQL statement like this as a transaction? Why or why not?

Yes, the database should treat a single SQL statement like R(X) as a transaction. The reasoning behind treating such a simple query as a transaction is that, even with a single read operation, a dirty read/write-read conflict is still possible. Thus, it is crucial that even a single read is placed appropriately in the transaction schedule, along with strict two-phase locking, to avoid such conflicts. It is important to note that a dirty read can still happen in the case where strict two-phase locking isn't implemented. In conclusion, the only way to ensure that the single read query follows the ACID properties is to make it a transaction. Furthermore, regardless of if we decide to treat simple read queries like the one above as transactions or not, the DBMS will automatically treat them as such. That is, DBMS automatically treats each SQL statement as its own transaction, regardless of whether the user explicitly wrote them out as a transaction or not.