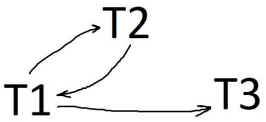
SEAN OVERTON SN:6421490
CSCI235-Assignment 2

TASK 1:(assuming initial values are x=1, y=2, z=3)



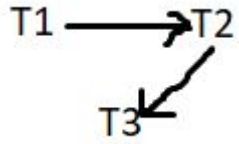
(1) Sample Concurrent execution of deferred-update transactions that is non-conflict serializable because no possible serial execution exists with same order of conflicting operations (ie. operations including a write on the same data type). A conflict serialization graph is cyclical in nature and so the schedule is non conflict serializable. The schedule is view serializable though because a possible serial execution exists (T1 → T2 → T3) where same values are read initially and the final state is the same ie.(final writes are both completed by T3). Commit location is irrelevant.

	x	y	z
Initial read	t1	t2	t3
Final update (write)	t3	t3	-

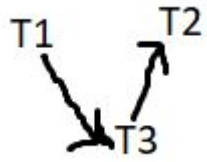
T1	**T2**	**T3**
read(x)		
	read(y)	
	write(x, y+1)	
write(y, x+ 1)		
	commit	
commit		
		read(z)
		write(x, z+1)
		write(y, z+2)
		commit

(2) Sample Concurrent execution that is conflict serializable because the serialization graph exists with no cycles formed. The schedule is not-order-preserving serializable because serialization graph order differs from start timestamps of schedule as seen below in visual depictions of schedules.

Conflict serialization graph:



Start timestamp schedule:



T1	T2	T3
read(x)		
		read(z)
write(y, x+ 1)		
	read(y)	
	write(x, y+1)	
commit		
	commit	
		write(x, z+1)
		write(y, z+2)
		commit

(3) Sample concurrent execution that is recoverable because for any transaction Ti that reads data written by another transaction Tj, transaction Ti commits after Tj commits. The schedule is not strict though because the data can still be read/written by Ti before Tj is committed. As observed below similar data (x, y) are committed in the same order as they are read/written by transactions but also each consecutive execution (T2, T3) reads/writes before the previous commit has occurred which means it is recoverable but not strict.

T1	T2	T3
read(x)		
write(y, x+ 1)		
		read(z)
	read(y)	

	write(x, y+1)	
commit		
		write(x, z+1)
		write(y, z+2)
	commit	
		commit