Part 3: Two-Phase Locking (20 points)

A) Now modify the above schedule by adding locks, which may block some transactions from doing their operations until the lock is released. You'll need to **rewrite** the above schedule in a table form. (The lecture slides show how to represent blocking in your schedules.)

Use two-phase locking (doesn't need to be "strict") in your modified schedule to ensure a conflict-serializable schedule for the transactions above.

Use the notation L(A) to indicate that the transaction acquires the lock on element A and U(A) to indicate that the transaction releases its lock on.

T_1	T_2	T_3
L(A); L(B)		
R(A)		
W(A), U(A)		
		L(A)
	L(A) blocked...	
		L(B) blocked...
R(B)		
W(B)		
U(B)		
		...granted L(B)
		R(A)
		W(A)
		U(A)
	...granted L(A)	
	L(B) blocked...	
		R(B)
		W(B)
		U(B)
	...granted L(B)	
	R(A)	
	R(B)	
	U(B) U(A)	

B) If 2PL ensures conflict-serializability, why do we need strict 2PL? Explain briefly.

The strict 2PL releases the lock immediately after the commit or rollback executes. The strict 2PL has the advantage mechanism that guarantees recoverable transactions. For instance, if we have a transaction that relies on previous results, if the transaction fails to update, then the following transaction would abort. So, we need strict 2PL to make sure that all transactions successfully commit.