A) Now modify the above schedule by adding locks, which may block some transactions from doing their operations until the lock is released. You will 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 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.

T1	T2	T3

L(A)
L(B)
	L(A) blocked…
	L(B) blocked…
		L(A) blocked…
		L(B) blocked…

R(A) W(A)		
R(B) W(B)		
COMMIT U(A) U(B)	L(A) blocked… L(B) blocked…	…granted L(A) …granted L(B)
		R(A) W(A)
		R(B) W(B)
	…granted L(A) …granted L(B)	COMMIT U(A) U(B)
	R(A)	
	R(B)	
	COMMIT U(A) U(B)	

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

If we do not use strict 2PL – where it follows standard 2PL and that all unlocks are done in conjunction with the commit or rollback – then it is not guaranteed to be a recoverable schedule. With strict 2PL since it is a cascadeless schedule, recovery (using rollback) is very trivial/easy.