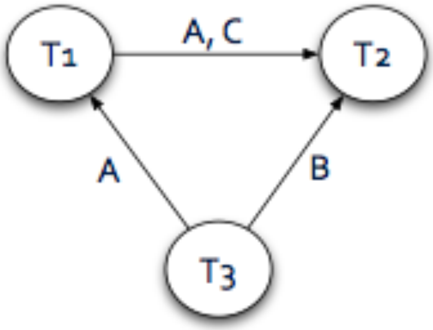
a) [6 points] Draw the dependency graph for this schedule. Be sure to list the object(s) (A, B, or C) that is (are) the cause of each dependency on each edge.



b) [5 points] Is this schedule conflict-serializable? If so, list a serial ordering of the transactions that would produce an equivalent schedule. If not, state why not.

Yes. T3 -> T1 -> T2.

c) [2 points] This schedule of read and write operations could be generated by a system following the regular **2PL** (two phase locking) protocol. (Circle one)

TRUE [✓ **FALSE**]

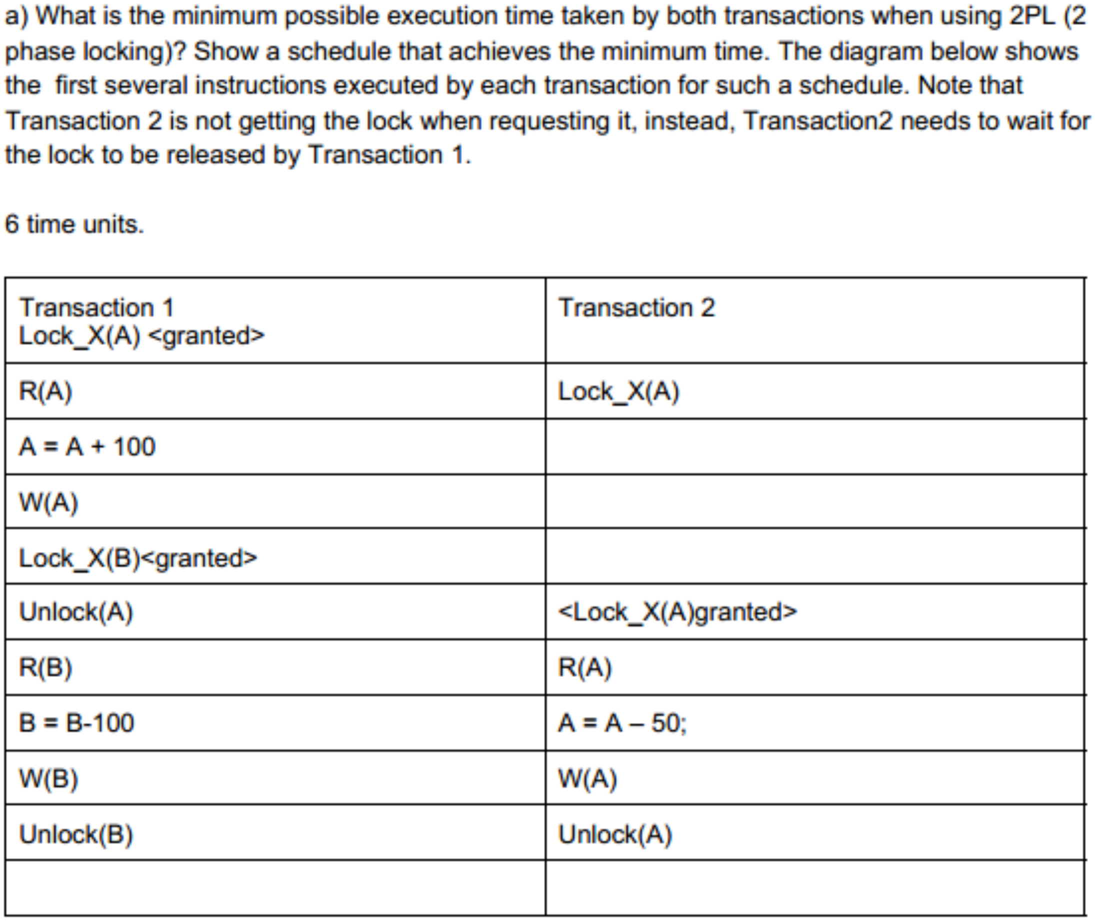
(We originally made a mistake on grading this question. We gave the points for "True", but the correct answer to this is "False".)

d) [2 points] This schedule of read and write operations could be generated by a system following the **Strict 2PL** protocol. (Circle one)

TRUE [✓ **FALSE**]

e) [4 points] In general, is Strict 2PL is more likely to encounter deadlocks than regular 2PL? State **Why** or **Why Not**.

Yes. Locks are held longer in Strict 2PL, thus increasing the likelihood of deadlocks.



a) What is the minimum possible execution time taken by both transactions when using 2PL (2 phase locking)? Show a schedule that achieves the minimum time. The diagram below shows the first several instructions executed by each transaction for such a schedule. Note that Transaction 2 is not getting the lock when requesting it, instead, Transaction2 needs to wait for the lock to be released by Transaction 1.

6 time units.

Transaction 1 Lock_X(A) <granted>	Transaction 2
R(A)	Lock_X(A)
A = A + 100	
W(A)	
Lock_X(B)<granted>	
Unlock(A)	<Lock_X(A)granted>
R(B)	R(A)
B = B-100	A = A – 50;
W(B)	W(A)
Unlock(B)	Unlock(A)

w