Exercises:

1. In general, is it possible to have a deadlock when the regular two-phase-locking (i.e., non-strict) protocol is obeyed? If yes, give an example; if not, explain briefly. What happens with strict 2PL and conservative 2PL?

2. For each of the following schedules:

   a) $S_a = r_1(A); w_1(B); r_2(B); w_2(C); r_3(C); w_3(A);$
   b) $S_b = r_1(A); r_2(A); r_1(B); r_2(B); r_3(A); r_4(B); w_1(A); w_2(B);$

   Answer the following questions:

   i. What is the precedence graph for the schedule?
   ii. Is the schedule conflict-serializable? If so, what are all the equivalent serial schedules?

3. Consider the following two transactions:
   T1 = w1(C) r1(A) w1(A) r1(B) w1(B);
   T2 = r2(B) w2(B) r2(A) w2(A)
   Say our scheduler performs exclusive locking only (i.e., no shared locks). For each of the following three instances of transactions T1 and T2 annotated with lock and unlock actions, say whether the annotated transactions:

   1. obey two-phase locking,
   2. will necessarily result in a conflict serializable schedule (if no deadlock occurs),
   3. will necessarily result in a recoverable schedule (if no deadlock occurs),
   4. will necessarily result in a schedule that avoids cascading rollback (if no deadlock occurs),
   5. will necessarily result in a strict schedule (if no deadlock occurs),
   6. will necessarily result in a serial schedule (if no deadlock occurs), and
   7. may result in a deadlock.

   a) T1 = **L1**(C) w1(C) **L1**(A) r1(A) w1(A) **L1**(B) r1(B) w1(B) Commit **U1**(A) **U1**(C) **U1**(B)
   T2 = **L2**(B) r2(B) w2(B) **L2**(A) r2(A) w2(A) Commit **U2**(A) **U2**(B)

   b) T1 = **L1**(B) **L1**(C) w1(C) **L1**(A) r1(A) w1(A) r1(B) w1(B) Commit **U1**(A) **U1**(C) **U1**(B)
   T2 = **L2**(B) r2(B) w2(B) **L2**(A) r2(A) w2(A) Commit **U2**(A) **U2**(B)

   c) T1 = **L1**(C) **L1**(A) w1(C) r1(A) w1(A) **L1**(B) r1(B) w1(B) **U1**(A) **U1**(C) **U1**(B) Commit
   T2 = **L2**(B) r2(B) w2(B) **L2**(A) r2(A) w2(A) Commit **U2**(A) **U2**(B)

   Format your answer in a table with Yes/No entries.

4. Examine the schedule given below. There are four transactions, T1, T2, T3, and T4.

T1	T2	T3	T4
			R(tax)
R(salary)			
			W(tax)
	R(tax)		
	W(tax)		
R(tax)			
W(salary)			
		R(salary)	
W(tax)			
		W(salary)	
			R(salary)
			W(salary)

a) Draw the precedence graph for this schedule.

b) What is the equivalent serialization order for this schedule? If no order is possible, then state 'none'.

c) Assume that transaction T4 did not run at all. What is the precedence graph in this case?

d) What is the equivalent serialization order for this second schedule? If no order is possible, then state 'none'.