Winter 2016 #4
(a) Consider a concurrency control manager that uses strict two phase locking that schedules three transactions:

- *T1 : R1(A), R1(B), W1(A), W1(B), Co1*
- *T2 : R2(B), W2(B), R2(C), W2(C), Co2*
- *T3 : R3(C), W3(C), R3(A), W3(A), Co3*

Each transaction begins with its first read operation, and commits with the *Co* statement. Answer the following questions for each of the schedules below:

- Is the schedule conflict-serializable? If yes, indicate a serialization order.
- Is this schedule possible under a strict 2PL protocol?
- If strict 2PL does not allow this schedule because it denies a read or write request, is the system in a deadlock at the time when the request is denied?

i. Schedule 1:

R2(B), W2(B), R3(C), W3(C), R3(A), W3(A), Co3, R2(C), W2(C), Co2, R1(A), R1(B), W1(A), W1(B), Co1

α) Is this schedule conflict-serializable? If yes, indicate a serialization order.

β) Is it possible under strict 2PL?

γ) Does strict 2PL lead to a deadlock?

ii. Schedule 2:

R2(B), W2(B), R3(C), W3(C), R1(A), R1(B), W1(A), W1(B), Co1, R2(C), W2(C), Co2, R3(A), W3(A), Co3

α) Is this schedule conflict-serializable? If yes, indicate a serialization order.

β) Is it possible under strict 2PL?

γ) Does strict 2PL lead to a deadlock?

(b) Consider the following three transactions:

- *T1 : R1(A), W1(B), Co1*
- *T2 : R2(B), W2(C), Co2*
- *T3 : R3(C), W3(D), Co3*

Given an example of a conflict-serializable schedule that has the following properties: transaction T1 commits before transaction T3 starts, and the equivalent serial order is T3, T2, T1.

(c) A read-only transaction is a transaction that only reads from the database, without writing/inserting/deleting. Answer the questions below by circling the correct answer.

i. If all transactions are read-only, then every schedule is serializable.
TRUE or FALSE

ii. Only one transaction can hold a shared lock on the same item at any time.
TRUE or FALSE

iii. Only one transaction can hold an exclusive lock on the same item at any time.
TRUE or FALSE

Autumn 2016 #4
Given the following three transactions:

```
T1: R(A), W(B), I(D), R(C)
T2: R(B), R(D), W(C)
T3: R(D), R(C), R(D), W(A)
```

Assume that `R(X)` reads all tuples in table `X`, `W(X)` updates all tuples in `X`, and `I(X)` inserts one new tuple in `X`. `Co` and `Ab` mean commit and abort, respectively.

a-c) SKIPPED

d) Does there exist a schedule of the above transactions that would result in a deadlock if executed under strict 2PL with **both shared and exclusive table locks**? If so write such a schedule with lock / unlock ops, and explain why the transactions are deadlocked. Otherwise write "No". Use `L1(A)` to refer to `T1` locking table `A`, and `U1(A)` for unlocking.

e) Does there exist a schedule of the above transactions that would result in a deadlock if executed under strict 2PL with **only exclusive table locks**? If so write such a schedule with lock and unlock operations and indicate why the transactions are deadlocked. Otherwise write "No". Use `L1(A)` to refer to `T1` locking table `A`, and `U1(A)` for unlocking.

f-g) SKIPPED