Assignment 8 (noted graded) Solutions

1. State which of the following schedules S_1, S_2, and S_3 over transactions T_1, T_2, and T_3 are conflict-serializable, and for each of the schedules that is serializable, given a serial schedule with which that schedule is conflict-equivalent.

   (a) $S_1 = R_1(x)R_2(y)R_1(z)R_2(x)R_1(y)$.

   Solution: This schedule consists of just read operations. Thus, there are no conflicting pairs and thus all T_1 operations can be swapped with all T_2 operations and as such the schedule S_1 can be transformed to the serial schedule $R_1(x)R_1(z)R_1(y)R_2(y)R_2(x)$, i.e. the schedule $T_1; T_2$.

   (b) $S_2 = R_1(x)W_2(y)R_1(z)R_3(z)W_2(x)R_1(y)$.

   Solution: The precedence graph $P(S_2) = \{(T_1, T_2), (T_2, T_1)\}$ which is cyclic. Thus, S_2 is not serializable.

   (c) $S_3 = R_1(z)W_2(x)R_2(z)R_2(y)W_1(x)W_3(z)W_1(y)R_3(x)$.

   Solution: The precedence graph $P(S_3) = \{(T_1, T_3), (T_2, T_1), (T_2, T_3)\}$ which is acyclic. Thus, S_3 is serializable.

   Using topological sort on this graph, S_3 is equivalent with the serial schedule $T_2; T_1; T_3$.

2. Given 3 transactions T_1, T_2, T_3 and a serializable schedule S on these transactions whose precedence graph (i.e. serialization graph) consists of the edges (T_1, T_2) and (T_1, T_3). Give 2 serial schedules that are conflict-equivalent with S.

   Solution: One of these serial schedules is $T_1; T_2; T_3$. Another is $T_1; T_3; T_2$.

3. Give 3 transactions T_1, T_2, T_3 and a schedule S on these transactions whose precedence graph (i.e. serialization graph) consists of the edges (T_1, T_2), (T_2, T_3), and (T_1, T_3).

   Solution:
   $$\begin{array}{rcl} T_1 & = & R_1(x)R_1(z) \\ T_2 & = & W_2(x)W_2(y) \\ T_3 & = & W_3(y)W_z(z) \end{array}$$

   $S = R_1(x)R_1(z)W_2(x)W_2(y)W_3(y)W_3(z)$.

4. Give 3 transactions T_1, T_2, T_3 and a schedule S on these transactions whose precedence graph (i.e. serialization graph) consists of the edges (T_1, T_2), (T_2, T_1), (T_1, T_3), (T_3, T_2).

   Solution:
$$\begin{array}{rcl} T_1 & = & R_1(u)R_1(x)R_1(z) \\ T_2 & = & W_2(u)W_2(x)W_2(w) \\ T_3 & = & W_3(w)W_3(z) \end{array}$$

   $S = W_2(u)R_1(u)R_1(x)R_1(z)W_2(x)W_3(w)W_3(z)W_2(w)$.

5. Give 3 transactions T_1, T_2, and T_3 that each involve read and write operations and a schedule S that is conflict-equivalent with **all** serial schedules over T_1, T_2, and T_3.

   Solution: We could pick the following transactions
$$\begin{array}{rcl} T_1 & = & R_1(x)W_1(x) \\ T_2 & = & R_2(y)W_2(y) \\ T_3 & = & R_3(z)W_3(z) \end{array}$$

   In this example there are no conflicting pairs and therefore any schedule over T_1, T_2, and T_3 can be transformed into any serial schedule using appropriate swap operations of non-conflicting operations.

6. Consider the following transactions:

```
T1:  read(A);
     read(B);
     if A = 0 then B := B+1;
     write(B).

T2:  read(B);
     read(A);
     if B = 0 then A := A+1;
     write(A).
```

Let the consistency requirement be $\mathtt{A} = 0 \vee \mathtt{B} = 0$, and let $\mathtt{A} = \mathtt{B} = 0$ be the initial values.

(a) Show that each serial schedule involving transactions T_1 and T_2 preserves the consistency requirement of the database.

Solution:

For the serial schedule $T_1; T_2$ the value for $A = 0$ and that for $B = 1$.

For the serial schedule $T_1; T_2$ the value for $A = 1$ and that for $B = 0$.

(b) Construct a schedule on T_1 and T_2 that produces a non-serializable schedule.

Solution:

T1	T2
R(A)	
	R(B)
	R(A)
	if B= 0 then A := A+1
	W(A)
R(B)	
if A = 0 then B := B+1	
W(B)	

The precedence graph consists of the edges (T_1, T_2) and (T_2, T_1). This is a cyclic graph so this schedule is not serializable.

(c) Is there a non-serial schedule on T_1 and T_2 that produces a serializable schedule. If so, give an example.

Solution:

The answer is no.

Assume that the schedule begins with R1(A). Observe that this action conflicts with W2(A).

Assume that we consider the partial schedule R1(A) R2(B). Observe that R2(B) conflicts with W1(B). So if we start the schedule with R1(A)R2(B) we get a cyclic precedence graph. Consequently, if we begin the schedule with R1(A) then the next action must be R1(B). So we must have R1(A)R1(B). We could now consider R2(B). But this will again create a cycle in the precedence graph. Thus we are required to do "R1(A)R1(B) if A = 0 then B:=B+1". Given this if we consider R2(B) we again have a problem since we will get a cyclic precedence graph. Thus we conclude that if we begin our schedule with R1(A), then we must execute the serial schedule $T_1; T_2$.

Now assume that we begin the schedule with R2(B). A similar analysis as above shows that if we want a serializable schedule, then we need to execute the serial schedule $T_2; T_1$.

(d) i. Add lock and unlock instructions to T_1 and T_2, so that they observe the two-phase locking protocol, but in such a way that interleaving between operations in T_1 and T_2 is still possible.
Solution By the previous argument, this can not be done.

ii. Can the execution of these transactions result in a deadlock? If so, give an example.
Solution.
Yes this can happen. Notice the following initial segment of a possible schedule $l_1(A)l_2(B)r_1(A)r_2(B)$ at this point (1) T_1 wants to read B but T_2 has a lock on B and (2) T_2 wants to read A but T_1 has a lock on A. So this schedule deadlocks.