Assignment 2 – Solution (revised)

1. $w_0[x,y,z]$ c_0 $r_1[x]$ $r_2[y]$ $w_2[y]$ $r_3[z]$ $w_3[z]$ $r_2[z]$ $w_2[y]$ $w_1[z]$ $w_1[y]$ c_1 c_2 c_3

   a. An equivalent serial history must preserve the order of conflicting operations. So, which operations conflict? We'll use ⇒ to mean "precedes and conflicts with".

      $w_0[x,y,z]$ ⇒ all other reads and writes
      $r_2[y]$ and both $w_2[y]$'s ⇒ $w_1[y]$
      $w_3[z] \Rightarrow r_2[z]$
      $r_3[x]$ and $w_3[z]$ ⇒ $w_1[z]$

      So, the only equivalent serial history has transactions in the order 0-3-2-1

   b. Since $w_3[z] \Rightarrow r_2[z]$ and $c_2 \Rightarrow c_3$ the history is not recoverable. Hence, it doesn't avoid cascading aborts and isn't strict. There are two other violations of strictness: $w_3[z] < w_1[z] < c_3$ and $w_2[y] < w_1[y] < c_2$.

2. $w_0[x,y,z]$ c_0 $r_1[x]$ $r_2[y]$ $w_2[y]$ $r_3[z]$ $r_2[z]$ $w_2[y]$ $w_1[z]$ $w_1[y]$ c_1 c_2 c_3
   (same as (1), except delete $w_3[z]$)

   a. We no longer have $w_3[z] \Rightarrow r_2[z]$. So the order of T_3 relative to T_2 is unconstrained. Therefore, the history is now equivalent to a serial history with transactions in the order 0-3-2-1 or 0-2-3-1.

   b. The history is now recoverable and avoids cascading aborts. But it still isn't strict because $w_3[z] < w_1[z] < c_3$ and $w_2[y] \Rightarrow w_1[y] \Rightarrow c_2$

3. $w_0[x,y,z]$ c_0 $r_1[x]$ $r_2[y]$ $w_2[y]$ $r_3[z]$ $w_3[z]$ $r_2[z]$ $w_2[y]$ $w_1[z]$ $w_1[y]$ c_1 $\mathbf{c_3}$ $\mathbf{c_2}$
   (same as (1), except that c_2 is moved after c_3)

   a. This has no effect on serializability, so the answer is the same as 1a.

   b. This also makes the history recoverable, since $w_3[z] \Rightarrow r_2[z]$ and $c_3 \Rightarrow c_2$. But it still doesn't avoid cascading aborts, because of the same conflict: T_2 reads uncommitted data (z) from T_3. Obviously, it is not strict.

4. $w_0[x,y,z]$ c_0 $r_1[x]$ $r_2[y]$ $\mathbf{w_2[x]}$ $r_3[z]$ $w_3[z]$ $r_2[z]$ $w_2[y]$ $w_1[z]$ $w_1[y]$ c_1 c_2 c_3
   (same as (1), except the first $w_2[y]$ becomes $w_2[x]$)

   a. Now we have $r_1[x] \Rightarrow w_2[x]$ and $w_2[y] \Rightarrow w_1[y]$ forming a cycle, so there is no equivalent serial history.

   b. $w_3[z] \Rightarrow r_2[z]$ and $c_2 \Rightarrow c_3$ is unchanged from (1), so the history is not recoverable since T_2 reads uncommitted data.

5. $w_0[x,y,z]$ c_0 $r_1[x]$ $r_2[y]$ $w_2[y]$ $r_3[z]$ $w_3[z]$ $r_2[z]$ $w_2[y]$ $\mathbf{c_2}$ $\mathbf{c_3}$ $w_1[z]$ $w_1[y]$ c_1
   (same as (1), except $\mathbf{c_2}$ and $\mathbf{c_3}$ are moved before $w_1[z]$)

   a. This has no effect on serializability

   b. It is tempting to think that this helps strictness, since we now have $w_3[z] < \mathbf{c_3} < w_1[z]$ and $w_2[y] < \mathbf{c_2} < w_1[y]$. But strictness implies avoidance of cascading aborts, which implies recoverability. And we still have the same old violation of recoverability: T_2 still reads uncommitted data (z) from T_3. So the execution isn't strict.