Solution Sketches

Assignment # 7

1. The number of interleavings is $^9C_3$, which is 84.
2. The number of serial schedules is, of course, 2.
3. The number of serializable schedules is slightly tricky. We need to first decide on a good consistency constraint. If all we wanted to do was preserve the database state \( i.e. \), the A and B elements as modified by some serial schedule), then `ReportSum` could be interleaved with `TransferBalance` in any order, since it doesn’t do any writing/modifications. A reasonable constraint would be to expect that, in addition, the value printed by `ReportSum` is the same (irrespective of even whether `TransferBalance` executes or not).

Let us first consider the serial schedule `TransferBalance` followed by `ReportSum`. This looks like:

- READ(A,x)
- \( x = x - 50 \)
- WRITE(A,x)
- READ(B,y)
- \( y = y + 50 \)
- WRITE(B,y)
- READ(A,x)
- READ(B,y)
- print x+y

The `READ(A,x)` of `ReportSum` could be moved as far back as just after the `WRITE(A,x)` of `TransferBalance` which is four choices (including its current position). However, the other two statements of `ReportSum` do not have any more degrees of freedom. Thus, the number of serializable schedules that are equivalent to this serial schedule is 4.

Let us now consider the second serial schedule (\( \text{ReportSum followed by TransferBalance} \)), which is:

- READ(A,x)
- READ(B,y)
- print x+y
- READ(A,x)
- x = x-50
- WRITE(A,x)
- READ(B,y)
- y = y+50
- WRITE(B,y)

Let us again try to move around the operations of ReportSum (since they are lesser in number). Here's a calculation tree-structure:

- READ(A,x) of ReportSum occurs just before READ(A,x) of TransferBalance. Combinations: 26
  * READ(B,y) and print x+y of ReportSum both appear before WRITE(B,y) of TransferBalance. Combinations: \(7C_2\) (Why 7, instead of 6?; because there are two operations to be interleaved).
  * READ(B,y) of ReportSum occurs before WRITE(B,y) of TransferBalance but print x+y occurs after. Combinations: 6 (the 6 positions in which READ(B,y) of ReportSum could be placed).
- READ(A,x) of ReportSum occurs just before x = x-50 of TransferBalance. Combinations: 20
  * READ(B,y) and print x+y of ReportSum both appear before WRITE(B,y) of TransferBalance. Combinations: \(6C_2\)
  * READ(B,y) of ReportSum occurs before WRITE(B,y) of TransferBalance but print x+y occurs after. Combinations: 5
- READ(A,x) of ReportSum occurs just before WRITE(A,x) of TransferBalance. Combinations: 14
  * READ(B,y) and print x+y of ReportSum both appear before WRITE(B,y) of TransferBalance. Combinations: \(5C_2\)
  * READ(B,y) of ReportSum occurs before WRITE(B,y) of TransferBalance but print x+y occurs after. Combinations: 4

The final answer (in Millionaire style :-)) is 65.

- The number of conflict-serializable schedules, among these is all of the 65. Why?
  Notice that any one of them, the way we have constructed them will lead to a serial schedule by a sequence of non-conflicting swapping operations.

2. It is clear that each of the serial schedules allows one swapping (and only one) to occur. Thus the number of conflict-serializable schedules is \(2^*(1+1) = 4\). If the order of increments in the second transaction were reversed, then the conflict-serializable orderings will be 3.

3. Before we proceed with the warning I locks, it is helpful to review our concepts of lock dominance and hybrid varieties. Consider the traditional matrix for S, X and I locks:
<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>X</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>I</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>

As can be seen $X$ dominates both $S$ and $I$, but neither of $S$ and $I$ dominate each other. We could thus attempt to create a hybrid lock for these two types. Now what would the row and/or column of such a lock look like? It would have to be the logical AND of the the rows and/or columns for $S$ and $I$ respectively. If you indeed do this calculation, you will notice that it turns out to be the $X$ mode! Thus, if a transaction requests both $S$ and $I$ locks on an element, granting just an $X$ lock will suffice. In other words, no additional rows and/or columns need be added.

We can now go into more ambitious territory. Consider the compatibility matrix with all our additional types of locks (the values for the entries are self-explanatory):

<table>
<thead>
<tr>
<th></th>
<th>IS</th>
<th>IX</th>
<th>S</th>
<th>X</th>
<th>I</th>
<th>II</th>
<th>SIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>IX</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>S</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>I</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>II</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>SIX</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Let us consider the issue of dominance:

- **IS and I.** Neither dominates the other. We could create another lock mode; its entries would be \{x, x, x, x, ✓, x\}.
- **IX and I.** Neither dominates the other. We could create another lock mode; its entries would be \{x, x, x, x, ✓, x\}. In other words, these are the same as the previous case. This is convenient, since we don’t have to create an additional lock mode.
- **S and I.** Neither dominates the other. The logical AND of their lock modes would just be the $X$ mode.
- **X and I.** Whenever $I$ has a x, $X$ also has a x. Thus, $X$ dominates $I$.
- **II and I.** Whenever $II$ has a x, $I$ also has a x. Thus, $I$ dominates $II$.
- **SIX and I.** Neither dominates the other. The logical AND of their lock modes would just be the $X$ mode.
- **IS and II.** Neither dominates the other. The logical AND of their lock modes would just be the IX mode. (Think about it, the logical AND of $S$ and $I$ was $X$! :))
- **IX and II.** Whenever $II$ has a x, $IX$ also has a x. Thus, $IX$ dominates $II$. 
• $S$ and $II$. Neither dominates the other. The logical AND of their lock modes would just be the $SIX$ mode (again, no surprises here).
• $X$ and $II$. Whenever $II$ has a $\times$, $X$ also has a $\times$. Thus, $X$ dominates $II$.
• $SIX$ and $II$. Whenever $II$ has a $\times$, $SIX$ also has a $\times$. Thus, $SIX$ dominates $II$.
• Phew!

Thus we have shown that we have to add just one other locking mode, which will be the group mode when either of both $IS$ and $I$ or both $IX$ and $I$ is requested. Stated in English terms, this lock mode wants to increment a whole hierarchy and (later) read/write only a part of it. Let's call it $M$ (for mystery mode). It is only compatible with the $II$ mode. The final compatibility matrix is:

<table>
<thead>
<tr>
<th></th>
<th>$IS$</th>
<th>$IX$</th>
<th>$S$</th>
<th>$X$</th>
<th>$I$</th>
<th>$II$</th>
<th>$SIX$</th>
<th>$M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$IS$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$IX$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$S$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$X$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$I$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$II$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>$SIX$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$M$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
</tbody>
</table>