I. Find an **interleaving** of the steps for the two transactions below that will produce an incorrect result.

<table>
<thead>
<tr>
<th><strong>T1:</strong></th>
<th><strong>T2:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>READ(A,s)</td>
</tr>
<tr>
<td>1.2</td>
<td>s = s+1</td>
</tr>
<tr>
<td>1.3</td>
<td>WRITE(A,s)</td>
</tr>
<tr>
<td>1.4</td>
<td>READ(B,s)</td>
</tr>
<tr>
<td>1.5</td>
<td>s = s-1</td>
</tr>
<tr>
<td>1.6</td>
<td>WRITE(B,s)</td>
</tr>
</tbody>
</table>

II. Draw precedence graphs for the schedules below. Then decide which are conflict-serializable. (Note: operations are read, write, and commite; subscripts identify the transaction.)


A. \( S_1 = r_3(A), r_2(B), w_2(B), r_1(B), w_3(C), c_3, r_2(D), c_2, r_1(D), c_1 \)

B. \( S_2 = w_3(D), r_1(A), w_2(D), w_2(A), w_3(A), c_3, w_3(C), r_2(D), c_2, c_1 \)

C. \( S_3 = r_1(X), w_2(X), r_2(x), W_3(x) \)

III. Locking. Is there an interleaving of these transactions that respects locking but isn’t serializable?

<table>
<thead>
<tr>
<th><strong>T1:</strong></th>
<th><strong>T2:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>LOCK(A)</td>
</tr>
<tr>
<td>1.2</td>
<td>READ(A,s)</td>
</tr>
<tr>
<td>1.3</td>
<td>s = s+1</td>
</tr>
<tr>
<td>1.4</td>
<td>WRITE(A,s)</td>
</tr>
<tr>
<td>1.5</td>
<td>UNLOCK(A)</td>
</tr>
<tr>
<td>1.6</td>
<td>LOCK(B)</td>
</tr>
<tr>
<td>1.7</td>
<td>READ(B,s)</td>
</tr>
<tr>
<td>1.8</td>
<td>s = s-1</td>
</tr>
<tr>
<td>1.9</td>
<td>WRITE(B,s)</td>
</tr>
<tr>
<td>1.10</td>
<td>UNLOCK(B)</td>
</tr>
</tbody>
</table>

IV. Timestamps. What is the outcome of the following schedule? Keep track of the metadata.

<table>
<thead>
<tr>
<th>TS(T1)</th>
<th>TS(T2)</th>
<th>TS(T3)</th>
<th>RT(A)</th>
<th>WT(A)</th>
<th>C(A)</th>
<th>RT(B)</th>
<th>WT(B)</th>
<th>C(B)</th>
<th>RT(C)</th>
<th>WT(C)</th>
<th>C(C)</th>
</tr>
</thead>
</table>

\[ S = r_3(A), r_3(B), w_3(B), r_2(B), r_3(B), r_2(C), w_2(C), w_1(A), w_2(B), w_3(C), c_1, c_2, c_3 \]