1. (16 points) Identify all of the following statements that are true about transactions and processes of various kinds

A. If we consider an enhanced Saga in which two consecutive subtransactions \( A \) and \( B \) may occur in either order, then those subtransactions may be compensated for in either order

**Solution:** \( A \) is false: must be compensated in reverse of order of occurrence

B. A Saga of three or more subtransactions may be defined where each of these subtransactions is vital

**Solution:** \( B \) is true: why not?

C. BPEL doesn’t support the common programming constructs of sequencing, branching, and iteration

**Solution:** \( C \) is false: it supports sequencing, branching, and iteration

D. In typical settings where Sagas and DOM extended transaction models can be applied, compensations are defined for at least some of the subtransactions

**Solution:** \( D \) is true:

E. No compensation is necessary for a read only subtransaction

**Solution:** \( E \) is true: no change in state of the information

F. In cases of failure of a subtransaction within a larger transaction, we may have to choose between retrying the one that failed or undoing some of those that succeeded

**Solution:** \( F \) is true: we may have to choose to retry the failed subtransaction, perhaps via an alternative or contingency approach, or undo those that succeeded (not necessarily all: just those that had side effects)

G. In the two-phase commit protocol, the second phase is needed only if at least one of the participants decides not to commit its transaction

**Solution:** \( G \) is false: the second phase is needed all the time to distribute a decision to the mutually committing parties

H. For \( n \geq 2 \), the \( n \)-phase commit protocol determines whether \( (n + 1) \) business partners unanimously agree to commit their respective transactions

**Solution:** \( H \) is false: \( n \)-phase commit as indicated here is bogus; two-phase commit can handle arbitrarily many parties

2. (10 points) Identify true statements from among the following about events, computations, and guards

A. “Neither \( e \) nor \( \pi \) may occur” may be expressed as \( 0 \)

**Solution:** \( A \) is true: \( e \) or \( \pi \) must occur on any computation

B. “If \( e \) precedes \( f \), then \( f \) cannot occur” may be expressed as \( \pi \lor \overline{f} \lor f \cdot e \)
<table>
<thead>
<tr>
<th>Solution</th>
<th>B is true: $f$ not occurring is one of the ways in which $e$ may fail to precede $f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.</td>
<td>When $e$ is immediate, $e \cdot f \lor \overline{f} \cdot e$ reduces to $e \cdot f \lor \overline{f} \cdot e$ (i.e., has no change)</td>
</tr>
<tr>
<td>Solution</td>
<td>C is true: $e$ is allowed in every possible state generated from the original specification</td>
</tr>
<tr>
<td>D.</td>
<td>Residuating $(\overline{e} \lor f \lor e \cdot f \lor f \cdot e)$ by $e$ yields $\top$</td>
</tr>
<tr>
<td>Solution</td>
<td>D is true:</td>
</tr>
<tr>
<td>E.</td>
<td>The guard of $f$ produced from $(\overline{e} \lor f \lor e \cdot f \lor f \cdot e)$ is $\lozenge \overline{e} \lor \neg e \lor \Box e$</td>
</tr>
<tr>
<td>Solution</td>
<td>E is false: Should be $\lozenge \overline{e} \lor \neg e \lor \Box e$, which simplifies to $\top$</td>
</tr>
</tbody>
</table>