Problem 1  (Textbook Problem 12.3)

Let relations \( r_1(A,B,C) \) and \( r_2(C,D,E) \) have the following properties: \( r_1 \) has 20,000 tuples, \( r_2 \) has 45,000 tuples, 25 tuples of \( r_1 \) fit on one block, and 30 tuples of \( r_2 \) fit on one block. Estimate the number of block transfers required, using each of the following join strategies for \( r_1 \bowtie r_2 \):

1. Nested-loop join
2. Block nested-loop join
3. Merge join
4. Hash join

Problem 2  (Textbook Problem 13.4)

Consider the relations \( r_1(A,B,C) \), \( r_2(C,D,E) \), and \( r_3(E,F) \), with primary keys \( A \), \( C \), and \( E \), respectively. Assume that \( r_1 \) has 1000 tuples, \( r_2 \) has 1500 tuples, and \( r_3 \) has 750 tuples. Estimate the size of \( r_1 \bowtie r_2 \bowtie r_3 \) and give an efficient strategy for computing the join.

Problem 3

Consider the following query on the \( account(aID, name) \) and \( deposit(aID, date, amount) \) relations:

\[
\text{SELECT a.name, d.date, d.amount} \\
\text{FROM deposit AS d} \\
\text{INNER JOIN account AS a} \\
\text{ON a.aID = d.aID} \\
\text{WHERE amount \geq 400}
\]

Assume that \( account \) contains 10,000 accounts, and every account has made 50 deposits on average (the \( deposit \) table contains 500,000 deposits total). Both relations are not sorted in any particular order, and there are no indices on the relations. The in-memory buffer can hold up to 12 blocks and there is 100 tuples on average in every block (for both tables). Deposits range from $100 (inclusive) to $500 (exclusive), and you may assume an even distribution.

1. Using a Block Nested-Loop Join, compute the number of block accesses that will be required to perform the operation.
2. Compute the block accesses again using a Merge-Join instead. Both the \( deposit \) and \( account \) relations remain unordered.
3. Now let’s add some indices to these tables. First let’s add a primary index on \( deposit.amount \). The \( account \) relation remains unordered. Again, using a Block Nested-Loop Join, compute the number of block accesses that will be required to perform the operation. Assume that the \( WHERE \) clause is evaluated before the \( JOIN \), with an index fan out of 100.
4. Assume that the primary index is now changed to \( deposit.aID \), rather than \( deposit.amount \). The \( account \) relation remains unordered. Using a Indexed Nested-Loop Join, compute the number of block accesses that will be required to perform the operation.
5. A primary index is added on \( account.aID \). Now that both relations are sorted, recompute the number of block accesses using a Merge-Join.
Problem 4

Consider a database with the following initial values, and the attached command log:
\[ A = 50, B = 48, C = 0, D = 47 \]

LOG:
\[
< T_0, \text{start} > \\
< T_0, A, 50, 75 > \\
< T_1, \text{start} > \\
< T_1, B, 48, 92 > \\
< T_2, \text{start} > \\
< T_2, C, 0, 33 > \\
< T_1, B, 92, 108 > \\
< \text{checkpoint} : T_0, T_1, T_2 > \\
< T_3, \text{start} > \\
< T_0, A, 75, 100 > \\
< T_2, \text{commit} > \\
< T_3, D, 47, 52 > \\
< T_3, \text{commit} >
\]

Assume that the system crashes before the remaining transactions can commit. Use the recovery protocol for concurrent transactions (which persists all in-memory dirty pages and transaction log entries at each checkpoint) to answer the following questions.

1. List any transactions that will need to be undone or redone in the recovery process.
2. List, in order, the set of logged operations to be performed to undo or redo the transactions. (i.e. "Set A to 7", "Set B to 39", etc.)
3. Give the final values for A, B, C, and D.

Problem 5

Assume the following transactions: T1, T2 and T3:

<table>
<thead>
<tr>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

Answer the following questions for the schedule:

1. Is the schedule serializable? Justify your answer by drawing the precedence graph.
2. Assume that all three transactions begin and end at the same time. Could the schedule be produced by the two-phase locking protocol? Insert lock and unlock operations into the schedule to justify your answer.
3. Making the same assumptions, could the schedule be produced by the strict two-phase locking protocol? Insert lock and unlock operations into the schedule to justify your answer.