1. Consider the expression $r_1 \bowtie r_2$ and the following numerical information. r_1 with schema R_1 is stored at S_1 and r_2 with schema R_2 is stored at S_2 and the result is needed at S_1.

- Size of r_1: 300,000 bytes
- Size of r_2: 180,000 bytes
- Number of tuples in r_1: 20,000
- Number of tuples in r_2: 15,000
- Size of $R_1 \cap R_2$: 9 bytes
- $R_1 \cap R_2$ in R_1 is a foreign key referencing R_2

Determine whether ***semi-join*** strategy should be used in the following two cases.

a) 2/3 tuples of r_2 join with some tuples of r_1.

b) Only 1/3 tuples of r_2 join with some tuples of r_1.

a)

Strategy 1: Ship r_2 to S_1

Transmission cost = 180,000 bytes

Strategy 2: Semijoin

1. Compute $temp_1 \leftarrow \prod_{R1 \cap R2} (r_1)$ at S_1
2. Ship $temp_1$ from S_1 to S_2; transmission cost = 90,000 bytes
3. Compute $temp_2 \leftarrow r_1 \bowtie temp_1$ at S_2
4. Ship $temp_2$ from S_2 to S_1; transmission cost = 120,000 bytes
5. Compute $r_1 \bowtie temp_2$ at S_1.

Total transmission cost = 90,000 + 120,000 = 210,000 bytes

Strategy 1 has a lower transmission cost.

b)

Strategy 1: Ship r_2 to S_1

Transmission cost = 180,000 bytes

Strategy 2: Semijoin

1. Compute $temp_1 \leftarrow \prod_{R1 \cap R2} (r_1)$ at S_1
2. Ship $temp_1$ from S_1 to S_2; transmission cost = 45,000 bytes
3. Compute $temp_2 \leftarrow r_2 \bowtie temp_1$ at S_2.
4. Ship $temp_2$ from S_2 to S_1; transmission cost = 60,000 bytes
5. Compute $r_1 \bowtie temp_2$ at S_1.

Total transmission cost = 45,000 + 60,000 = 105,000 bytes

Strategy 2 has a lower transmission cost.

2. Consider the following schedule.

$r_1(X)$; $r_2(X)$; $w_1(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$;

(a) Should the schedule be allowed? Why?
(b) Draw a precedence graph for the schedule.
(c) Is this schedule conflict-serializable?

(a) The schedule should not be allowed. There is a ***lost update problem***: The final value of X is incorrect because T_2 reads the value of X *before* T_1 changes it in the database, and hence the updated value resulting from T_1 is lost.
(b) $T_2 \rightarrow T_1 \rightarrow T_2$
(c) The schedule is not conflict-serializable.

3. Consider the following schedule.

$r_1(X)$; $w_1(X)$; $r_2(X)$; $w_2(X)$; $r_1(Y)$;

(a) Is there any problem if T_1 fails at the end of the schedule?
(b) Is there any problem if T_2 commits followed by T_1 fails at the end of the schedule?
(c) What can be done to fix the problem in (b)?

(a) ***Dirty read problem***: T_1 updates X and then fails, so the system must change X back to its original value. Before it can do so, however, T_2 reads the temporary value of X, which will not be recorded permanently in the database because of the failure of T_1. The value of X that is read by T_2 is called dirty data because it has been created by a transaction that has not completed and committed yet.
(b) The schedule becomes ***unrecoverable*** because T_2 reads X from T_1, but T_1 aborts after T_2 commits. Then, the value of X that T_2 read is no longer valid and T_2 must be aborted *after* it is committed, leading to a schedule that is not recoverable.
(c) The commit instruction of T_2 must be postponed until after T_1 commits. So, if T_1 aborts, then T_2 can also abort.

4. Consider the following schedule in which T_3 is calculating a total of data items A to Y.

$r_3(A)$; … $r_1(X)$; $w_1(X)$; $r_3(X)$; $r_3(Y)$; $r_1(Y)$; $w_1(Y)$;

(a) Is the total calculated by T_3 correct? Why?
(b) Draw a precedence graph for the schedule.
(c) Is this schedule conflict-serializable?

(a) The total calculated by T_3 is incorrect. There is an ***incorrect summary problem***: T_3 is calculating an aggregate summary function (such as total) while T_1 is updating X and Y. The aggregate function calculates Y before it is updated and X after it is updated.
(b) $T_1 \rightarrow T_3 \rightarrow T_1$
(c) The schedule is not conflict-serializable.

5. Consider the following schedule that involves three transactions, T_1, T_2, and T_3.

$r_2(A)$; $r_1(B)$; $w_2(A)$; $r_3(A)$; $w_1(B)$; $w_3(A)$; $r_2(B)$; $w_2(B)$;

(a) Draw a precedence graph for the schedule.
(b) Is this schedule conflict-serializable?
(c) Construct a conflict-equivalent serial schedule by swapping instructions.

(a) $T_1 \rightarrow T_2 \rightarrow T_3$
(b) This schedule is conflict-serializable.
(c) $r_1(B)$; $w_1(B)$; $r_2(A)$; $w_2(A)$; $r_2(B)$; $w_2(B)$; $r_3(A)$; $w_3(A)$;