Exercise 1: Assume a *deferred* database modification scheme. Consider the following example of redo log for two transactions.

1. (Start, T1);
2. (Write, T1, Q, 100);
3. (Commit, T1);
4. (Start, T2);
5. (Write, T2, P, 55);
6. (Commit, T2);

Consider the following two cases:

- Case 1: The schedule crashes after Step 5 and before Step 6, and the recovery completes successfully.
- Case 2: The schedule crashes after Step 5 and before Step 6, and the recovery process starts. During recovery process, the schedule crashes after Step 2 and before Step 3.

What actions (Redo/No action) must be done on the transactions for the two cases?

Answer 1. Case 1: T1 - Redo, T2 - No action; Case 2: T1 - Redo, T2 - No action.
In case of deferred database modification, writes are reflected on the database only after commit log has been written. Hence, for consistent state in both cases, T1 must do the redo operation. On the other hand, no action needs to be taken for T2, because the write operation of T2 has not been reflected on the database. During the second recovery, since it is not guaranteed that the first write has gone through or not, T1 needs to write once more.

Exercise 2: Assume an immediate database modification scheme. Consider the following log consisting of transactions T1, T2, and T3.

1. (Start, T1);
2. (Write, T1, P, 500, 600);
3. (Write, T1, Q, 400, 500);
4. (Commit, T1);
5. (Start, T2);
6. (Write, T2, P, 600, 550);
7. (Write, T2, Q, 500, 450);
8. (Commit, T2);
9. (Start, T3);
10. (Write, T3, P, 550, 600);
11. (Write, T3, Q, 450, 500);
12. (Commit, T3);

(a) If the schedule crashes just after the Step 3, what recovery operations will be performed?

(b) If the schedule crashes just after the Step 7, what recovery operations will be performed?
(c) If the schedule crashes just after Step 11, then after the complete recovery process the value of P and Q will be?

Answer 2:
(a) Redo (T1), Undo (T1)
(b) Redo (T1), Redo (T2), Undo (T2) as Redo operations are done first followed by Undo operations.
(c) P = 550, Q = 450

Exercise 3: Consider the following log file, created in a basic checkpointing recovery protocol environment:

(start T1); (W1. A, 2, 3); (start T2); (W2, B, 4, 5); (W1, B, 5, 6); (start T3); (commit T1); (W3, A, 3, 6); (Checkpoint, T3, T2); (start T4); (W4, A, 6, 7); (W3, A, 7, 9); (W4, B, 6, 7); (commit T4); (start T5); (W5, A, 9, 4);

If the system crashes now, what is the correct order of recovery operations using undo-list and redo-list?

Answer 3. Redo first, then undo
Redo: (T4, A=7) (T3, A=9) (T4 B=7)(T5 A=4)
Undo: {(T5 A=9) (T3, A=7) (T3 A=3) (T2 B=4)}
First all redo, then undo.

Exercise 4: Consider a database that has six elements - A, B, C, D, E and F. The initial values of the elements are:

A = 10, B = 20, C = 30, D = 40, E = 50, F = 60

Let there be three transactions U, V and W that modify these elements concurrently.

- U: A := 5, B:= 15, D := 30
- V: C := 25
- W: E := 35, F:=45

While the elements are being modified by the transactions, the database system crashes. The recovery mechanism depends on the logging scheme we use. In the questions below, we present the contents of the log at the time of crash when the logging scheme used is **steal/no force.** (For each log entry we give the relevant data)

< START U > ;< U, A, 10, 5 >; < START V >; < U, B, 20, 15 >; < V, C, 30, 25 > ;< COMMIT V >; < START CKPT >; < START W >; < U, D, 40, 30 > ;< W, E, 50, 35 >; < END CKPT >; < W, F, 60, 45 >

a) Is the following state of database elements (on disk) possible at the time of crash? Justify your answer.

A = 5, B = 15, C = 25, D = 40, E = 35, F = 45

b) What are the values of the database elements (A=?, B=?, C=?, D=?, E=?, F=?) after a successful recovery?

What different compensation log records will be written in the log during recovery?

Answer 4:

The steal policy allows the system to write modified blocks to the disk even if the transactions that made those modifications have not all committed. The No-force policy allows a transaction to commit even if it has modified some blocks that have not yet been written back to disk.

(a) Yes, before checkpointing, all updates must be on the disk. Remaining may or may not be there.
(b) A = 10, B = 20, C = 25, D = 40, E = 50, F = 60. Transactions U, W are undone. V has already committed and was not active at the start of the checkpoint. Hence, its effect has been reflected on the disk already.

Compensation log records will be as follows:

<W, F, 60>
<W, E, 50>
<U, D, 40>
<W Abort>
<U, B, 20>
<U, A, 10>
<U Abort>