Hands-on 2

Before crash:

```
> show_state

-----------------------
On-disk DB contents:
Account: studentA Value: 100
-----------------------

-----------------------
LOG contents:
type: START action_id: 1
type: UPDATE action_id: 1 variable: studentA redo: "100" undo: NULL
type: OUTCOME action_id: 1 status: COMMITTED
type: END action_id: 1
type: START action_id: 2
type: UPDATE action_id: 2 variable: studentB redo: "2000" undo: NULL
type: UPDATE action_id: 2 variable: studentA redo: "200" undo: "100"
type: START action_id: 3
type: UPDATE action_id: 3 variable: studentC redo: "3000" undo: NULL
type: CHECKPOINT PENDING: id: 3 id: 2 COMMITTED:  DONE: id: 1
type: OUTCOME action_id: 2 status: COMMITTED
type: UPDATE action_id: 3 variable: studentC redo: "2900" undo: "3000"
-----------------------
```

Crash and recover for the first time:

```
> crash
crashing ...
linshuhuai@sjtu-linshuhuai:~/CSE$ ./wal-sys.py
Recovering the database ..
Starting rollback ...
  The log was rolled back 8 lines
Rollback done
Winners: id: 2  Losers: id: 3  Done: id: 1
Starting forward scan ...
  REDOING: type: UPDATE action_id: 2 variable: studentB redo: "2000" undo: NULL
  REDOING: type: UPDATE action_id: 2 variable: studentA redo: "200" undo: "100"
  Logging END records for winners
Forward scan done
Recovery done
>
```

Crash and recover for the second time:

```
> crash
crashing ...
linshuhuai@sjtu-linshuhuai:~/CSE$ ./wal-sys.py
Recovering the database ..
Starting rollback ...
  The log was rolled back 4 lines
Rollback done
Winners:   Losers: id: 3  Done: id: 1 id: 2
Starting forward scan ...
  Logging END records for winners
Forward scan done
Recovery done
```

Crash and recover for more times:

```
> crash
crashing ...
linshuhuai@sjtu-linshuhuai:~/CSE$ ./wal-sys.py
Recovering the database ..
Starting rollback ...
  The log was rolled back 4 lines
Rollback done
Winners:   Losers: id: 3  Done: id: 1 id: 2
Starting forward scan ...
  Logging END records for winners
Forward scan done
Recovery done
```

Question 1: During checkpoint, wal-sys divides actions into three types: "PENDING", "COMMITTED" and "DONE", what is the meaning of these types?

Answer 1: "PENDING" means such actions haven't been committed or ended before the checkpoint. "COMMITTED" means such actions have been committed but hasn't been written to database before checkpoint. "DONE" means such actions have been committed and written to database before checkpoint.

Question 2: What is the relationship between the action categories during checkpoint ("PENDING", "COMMITTED" and "DONE") and action categories during recovery ("Winners", "Losers", and "Done")?

Answer 2: The "PENDING" actions in a checkpoint could be a "Winner", "Loser" or "Done" in the later recovery, depending on whether it is committed and whether it's end after the checkpoint. The "COMMITTED" action in a checkpoint could be a "Winner" or "Done" in the later recovery, depending on whether it's end after that. The "DONE" action must be done in the later recovery.

Question 3: How many lines were rolled back? What is the advantage of using checkpoints?

Answer 3: Only 8 lines were rolled back. With checkpoints, we can reduce the lines needed to be rolled back during a recovery, because it updates the actions ended before checkpoints into database. In that case, the efficiency of recovery could improve.

Question 4: Does the second run of the recovery procedure restore "DB" to the same state as the first run? What is this property called?

Answer 4: Yes, there is nothing to redo in the second recovery. It's called idempotence(幂等性).

Question 5: Compare the `action_ids` of "Winners", "Losers", and "Done" from the second recovery with those from the first. The lists are different. How does the recovery procedure guarantee the property from Question 4 even though the recovery procedure can change? (Hint: Examine the "LOG" file).

Answer 5: During the first recovery, action 2 was found as a "Winner". To guarantee the idempotence, the recovery procedure would redo action 2 and flush the new status into database. Hence it's alright to mark it "Done" and don't need to redo this action in the next recovery.

Question 6 (Optional): Wal-sys has a hitherto unmentioned option: if you type wal-sys -undo it will perform undo logging and undo recovery. Try the above sequences again with undo logging to see what changes.

Answer 6:

./wal-sys.py -reset -undo <cmd.in

```
> show_state

-----------------------
On-disk DB contents:
Account: studentC Value: 2900
Account: studentB Value: 2000
Account: studentA Value: 1100
-----------------------

-----------------------
LOG contents:
type: START action_id: 1
type: UPDATE action_id: 1 variable: studentA redo: "1000" undo: NULL
type: OUTCOME action_id: 1 status: COMMITTED
type: END action_id: 1
type: START action_id: 2
type: UPDATE action_id: 2 variable: studentB redo: "2000" undo: NULL
type: UPDATE action_id: 2 variable: studentA redo: "1100" undo: "1000"
type: START action_id: 3
type: UPDATE action_id: 3 variable: studentC redo: "3000" undo: NULL
type: CHECKPOINT PENDING: id: 3 id: 2 COMMITTED:  DONE: id: 1
type: OUTCOME action_id: 2 status: COMMITTED
type: UPDATE action_id: 3 variable: studentC redo: "2900" undo: "3000"
-----------------------
```

recovery 1:

```
> crash
crashing ...
linshuhuai@sjtu-linshuhuai:~/CSE$ ./wal-sys.py -undo
Recovering the database ..
Starting rollback ...
  UNDOING: type: UPDATE action_id: 3 variable: studentC redo: "2900" undo: "3000
"
  UNDOING: type: UPDATE action_id: 3 variable: studentC redo: "3000" undo: NULL
  The log was rolled back 5 lines
Rollback done
Winners: id: 2  Losers: id: 3  Done: id: 1
 Logging STATUS records for losers
Recovery done
```

recovery 2:

```
> crash
crashing ...
linshuhuai@sjtu-linshuhuai:~/CSE$ ./wal-sys.py -undo
Recovering the database ..
Starting rollback ...
  The log was rolled back 4 lines
Rollback done
Winners: id: 3 id: 2  Losers:   Done: id: 1
 Logging STATUS records for losers
Recovery done
```

"LOG" file shows "OUTCOME action_id : 3 status ABORTED".