6.033
COMPUTER SYSTEMS ENGINEERING
HANDS ON 7: LOGGING

JOHN WANG

1. Using Walsys

(1) The database doesn't show *studentB* because the action that created *studentB* has not yet been committed.
(2) The *studentA* and *studentC* accounts should be active because their creations were committed. However, *studentA* should have only 900 in his account because his debit transaction was committed and *studentC* should have 3100 in his account because his credit transaction was committed.
(3) The database does not contain a record for *studentC* and there is a pre-debit balance for *studentA* because the second transaction has not yet been committed. Therefore, the database has not yet been written to yet.

2. Recovering the Database

(4) I expect the database to be in the state where everything in transactions 1 and 3 have been completed, but not the things that were performed only in transaction 2. Therefore, I don't expect *studentB* to exist, but I do expect *studentC* to exist with 3100 in his account. I expect this because the recovery mechanism will make sure that everything that was committed is displayed in the recovered database.
(5) Yes, the state matches my expectations because the recovery did indeed bring back everything that was committed, and threw away the data that had not yet been committed.
(6) The winners are all all-or-nothing actions that logged an outcome status before the crash, the losers are the set of all-or-nothing actions that were still in progress at the time of crash, and done category gives all actions that were finished and committed before the crash.

3. Checkpoints

(7) That particular action had not yet been committed, therefore, it did not get put into the checkpoint. Since the action was in the middle of being committed (but had not yet been committed), the checkpoint did not include the action.
(8) The log was rolled back 6 lines. The advantage of checkpoints is that the recovery mechanism does not have to roll back all 12 lines, since it can stop once it gets to a checkpoint. This reduces the time it takes for recovery.
(9) Yes, the second run of the recovery procedure correctly restores the database state to the same state as was recovered in the first run. This property is called *idempotence*.
(10) The first recovery writes an END to all actions which are marked with loser, and transforms the loser into a completed action. This guarantees that future recoveries will ignore it and not perform the recovery again. This prevents future updates from incorrectly undoing updates to the actions. The process of finishing all loser actions in volatile storage makes sure that recovery process is idempotent, even when the log changes.