LogFile Implementation:

- rollback():
    - My implementation of rollback was, for a particular tid, seek() to the start for that transaction (BEGIN_RECORD) and read the log file until the end of file.
    - Have a set data structure to keep record of already processed pages. Since we are doing a rollback, the page data should be the beforeImage for the first time the passed in tid made any changes to that page.
    - Reading the log file from the BEGIN_RECORD for tid, revert all changes that tid made to pages.
- recover():
    - Step 1: Find and seek() to the last successful checkpoint, if none start reading the log file from 8 bytes.
    - Step 2: If we have a checkpoint, get a list of all active transactions at the time of the checkpoint (before).
    - Step 3: Continue reading from the end of the checkpoint and get a list of all new transactions that may have started after our checkpoint. Also, during the forward read, keep record of all the transactions that commit after our checkpoint.
    - Step 4: Start again (seek()) from the earliest active transaction. Read the log file to the end of file, redoing any changes a transaction made if in our committed set, or reverting changes if not.

BufferPool Modifications:

I made the modifications to BufferPool.java to allow dirty uncommitted pages to be flushed to disk (STEAL policy). Adding a write-record to the database log ensures that we can revert changes that may have been flushed to disk in the event that a transaction aborts or the database crashes before it can commit. The database eviction policy now allows for any page to be flushed to disk, dirty or not. I also made changes to BufferPool to not require pages to be flushed to disk when a transaction commits (NO_FORCE). When a transaction commits, we just need to write a write-record to the database log, saving the new page data in case the database needs to recover from a crash before the page could be written to disk.

Unit Test:

We can add a more extensive unit test, similar to that of
TestOpenCommitCheckpointOpenCrash()
but instead add multiple checkpoints. This will help make sure our implementation does not affect previous checkpoints (commits from past checkpoints). On that note, we can add a unit test that makes sure we are not redoing any changes from past checkpoints. Although it shouldn't affect the flow of things, the point of checkpoints is to avoid reading from the start of our log file every time on recovery.