Andrew Dawson
1220796

This lab focused on building a log based recovery system. The high level idea is that every action taken is recorded to a log, the log is pushed to disk and then the log can be used to recover the database state after a crash. The log is useful for both recovering from a crash and for rolling back a transaction.

If a transaction is aborted the updates made by that transaction must be undone. This is easy if we use no-steal and force but, when when using steal and no-force - a log file needs to be used to rollback the effects of a transaction. Rollback takes a transactions that should be rollbacked and iterates over the log file - for every update by that transaction the before image of the page is extracted and the page is reverted to the before image. If an abort record is encountered for the transaction being aborted the abort is finished. All other types of records are ignored. For flexibility a private helper method was created to do rollbacks that accepts a Long instead of a transactionId. Without this helper method transactionsId read from the log file would have to deserialized before they could be used.

If the whole database system crashes, the entire in memory representation of the database is lost. When this happens the log file must be used to restore the database state. When the database is recovered both redos and undo happen. Any transactions that committed before the crash should have their updates reflected on disk, while transactions that aborted or did not commit should have their updates undone. In order to do this the log file is iterated over - every update is redone and a set of loser transactions is kept maintained. A loser transaction is a transaction that did not commit. So the logic is basically if a transaction commits remove it from the loser set, when a transaction starts add it to the loser set, if a transaction aborts undo its updates and remove it from the loser set. There are two options of where to start the recovery - the start of the file of a checkpoint. Starting at the start of the file is easy because at that point there are no loser transactions present. If starting from a checkpoint all active transactions at the checkpoint are initially loser transactions until they commit.

Outside of LogFile I had to change BufferPool.flushPage() to not set the page to clean. In LogTest insert is called, then flushAllPages is called then commit is called. Insert marks a page as dirty, flushAllPages before the change would mark the page as not dirty then when commit was called the page was not dirty so the beforeImage was not getting set. I documented this much more clearly in an email to Brandon.

The hardest part of this assignment was debugging. There were several small errors associated with moving through the log correctly. It was also unclear how the map from tid to log records was suppose to be used at first. Overall, this lab was not that much work and very interesting to learn about how rollbacks worked.