Summary of Lab 3: This lab mainly focused on the implementation of Transactions using both NO STEAL and the FORCE buffer management policy at the page level. FORCE was used to flush dirty pages after a transaction commits while NO STEAL evicts dirty pages after a transaction is completed. From there we created a LockManager that keeps track of which transactions are holding locks on each specific page. This allows the Buffer Pool to check whether a transaction is able to get a page when they call getPage(). The Lock Manager also makes sure to check the dependencies of the given transactions to guarantee that they are acyclical. The Lock Manager more specifically has to store and link PageId to PageLock instances, acquire read or write lock permissions, detect deadlocks, and also must release all locks on a transaction. Optimally, we should try to prevent a deadlock from occurring before it develops. Even if the transactions system encounters a deadlock, aborting allows transactions to be terminated before they develop a dependency cycle. Once no deadlocks occur, we can commit and transfer the data that was written to the disk.

Example of a unit test that could be added to improve the set of unit tests: One unit test that could be used to improve the set of unit tests would be to check whether a lock was created by the same given transactions on acquire and also eviction. This would help a lot with confirming if the same lock worked as intended. It would also help to add more deadlocking tests. This would allow me to understand the situations where the code develops a deadlock instead of debugging a series of concurrent threads.

Design Decisions: When I implemented LockManager, I locked at the page level. Acquiring the locks was done by repeating requests at the level of permission. This was because as long as no deadlocks occurred, the program is fine. The deadlock detection was performed with graph dependency. As for the BufferPool, I added Dirty page detection and flushed pages as well as released them once a transaction was completed. Then I also incorporated lock acquisition. I wasn't able to pass the testAverageNoGroup because the concurrency map can't return null on keys. I also failed tesetAllDirtyFails and AbortEvictionTest which I'm not entirely sure why that is the case.

Changes to the API: I made no changes.

Additional Feedback: None.