Lab 3 Report

Group Members:

Yee Yi Xian	1004601
Rebecca Ling	1004413

Design Decisions and Code Explanation

Below we explain our design decisions and the ~~important~~ non-trivial part of our code, as split into the classes we were required to implement for Lab 3. We implemented a lock manager to manage the locks on a page, which allowed for acquire/upgrade/release of locks on a page when there are no deadlocks.

Write the methods that acquire and release locks in BufferPool

- Modify getPage() to block and acquire the desired lock before returning a page.
- Implement unsafeReleasePage(). This method is primarily used for testing, and at the end of transactions.
- Implement holdsLock() so that logic in Exercise 2 can determine whether a page is already locked by a transaction.
- evictPage() method
- transactionComplete() method
- Implementing deadlock detection

BufferPool

Below are the methods we implemented or edited in buffer pool in order to add support needed for the lock system

- getPage() was edited to add acquire lock before getting file from the disk
- transcationComplete() implements the NO STEAL and FORCE part of the policy, which will flush and release all locks when the transaction has been completed regardless if it has been committed or aborted
- evictPage() was edited to implement the NO STEAL part of the policy, which is to detect whether the page has been dirtied before evicting a page

LockManager

We implemented an extra class LocksOnPage for each page. This class is used to keep track of the exclusive lock and all the shared locks on it.
Additionally, below are the methods that we implemented in LockManager

- acquire() to acquire an exclusive lock or shared lock for a page, depending on the permission, READ_ONLY or READ_WRITE, and whether the page lock is already acquired. It uses lockStatus() to check the availability of the page's locks and adds an exclusive lock to a page using upgradeLock(). It also uses detectDeadlock() to identify deadlocks and abort the current transaction if a deadlock is found.

- lockStatus() returns true or false depending on whether the page is locked.
- upgradeLock() adds exclusive lock to a page with only shared lock
- releaseLock() releases a lock depending on the transaction id and page id.
- releaseAll() releases all the locks.
- detectDeadlock() detects whether there’s any deadlocks between the transactions.

Missing or incomplete elements

We managed to successfully implement everything as required in Lab 3. Yay!