CSCI 3357: Database System Implementation
Homework Assignment 5
Due Wednesday, October 16

SimpleDB currently uses timeout to detect deadlock. Change it so that it uses the wait-die deadlock detection strategy, as described in Figure 5.22 of the text. Your code should modify the class `LockTable` as follows:

- The methods `sLock`, `xLock`, and `unLock` will need to take the transaction's id as an argument.
- The variable `locks` must be changed so that a block maps to a list of the transaction ids that hold a lock on the block (instead of just an integer). Use a negative transaction id to denote an exclusive lock.
- Each time through the while loop in `sLock` and `xLock`, check to see if the thread needs to be aborted (that is, if there is a transaction on the list that is older than the current transaction). If so, then the code should throw a `LockAbortException`.
- You will also need to make trivial modifications to the classes `Transaction` and `ConcurrencyMgr` so that the transaction id gets passed to the lock manager methods. I'm sure you can figure out what those changes must be.

My solution required replacing a lot of code, but the resulting amount of code did not change much.

I have included the test program `HW5Test` for you to download. This program is similar to the class `ConcurrencyTest` of Figure 5.19 in the text. It creates three transactions, each in their own thread. Transactions A and C both need a lock held by transaction B. Under the wait-die algorithm, transaction A should wait, whereas transaction C should throw an exception. When I run the program on my solution, I get the following output:

```
new transaction: 1
Transaction A starts
Tx A: request slock block 1
Tx A: receive slock block 1
new transaction: 2
Transaction B starts
Tx B: request xlock block 2
Tx B: receive xlock block 2
new transaction: 3
Transaction C starts
Tx C: request xlock block 1
Transaction C aborts
transaction 3 rolled back
Tx A: request slock block 2
Tx B: request slock block 1
Tx B: receive slock block 1
transaction 2 committed
Transaction B commits
Tx A: receive slock block 2
transaction 1 committed
Transaction A commits
```