CSE 444 – Homework 3
Transactions Concurrency Control

Name: _____

Question	Points	Score
1	20	
2	20	
Total:	40	

1 Concurrency Control with Locking

1. (20 points)

   (a) (5 points) Consider a database with objects X, Y, and Z and assume that there are two transactions T1 and T2. Transaction T1 reads objects X and Y, writes X, and commits. Transaction T2 reads objects X and Y, writes object Y. It then reads objects X and Y again, writes X. Finally, it reads object Z, writes it, and commits. Give three examples of schedules for the transactions T1 and T2 to illustrate each of the points below:

      1. Your schedule should contain a write-read conflict that causes one of the transactions to perform a dirty read.
      2. Your schedule should contain a read-write conflict that causes one of the transactions to encounter an unrepeatable read.
      3. Your schedule should contain a write-write conflict that causes a lost update.

   In each case, your schedule may contain additional conflicts, but should contain at least one conflict of the type indicated. (In particular you may give a single schedule, which illustrates all three conflicts!) In each case, indicate the conflict of the type you are illustrating.

(b) (5 points) Consider the following three transactions and schedule (time goes from top to bottom). Is this schedule conflict-serializable? Explain why or why not.

Transaction T_0	Transaction T_1	Transaction T_2
$r_0[A]$		
$w_0[A]$		
		$r_2[A]$
		$w_2[A]$
	$r_1[A]$	
$r_0[B]$		
		$r_2[B]$
$w_0[B]$		
		$w_2[B]$
	$r_1[B]$	
	c_1	
c_0		
		c_2

(c) (5 points) Show how 2PL can ensure a conflict-serializable schedule for the same transactions above. Use the notation $L_i[A]$ to indicate that transaction i acquires the lock on element A and $U_i[A]$ to indicate that transaction i releases its lock on A.

(d) (5 points) If 2PL ensures conflict-serializability, why do we need *strict* 2PL?

2 Optimistic Concurrency Control

2. (20 points)

   (a) (10 points) Consider the following schedule. Explain what happens when transactions *try* to execute as per this schedule and the DBMS uses timestamp-based concurrency control. We use ST to denote the start of a transaction, C for commit, and A for abort.

   $ST_1 \rightarrow ST_2 \rightarrow ST_3 \rightarrow ST_4 \rightarrow R_2(X) \rightarrow R_1(X) \rightarrow W_2(X) \rightarrow W_4(X) \rightarrow W_1(X) \rightarrow C_1 \rightarrow W_3(X) \rightarrow A_4 \rightarrow R_2(Y) \rightarrow W_2(Y) \rightarrow R_3(Y) \rightarrow C_2 \rightarrow W_3(Y) \rightarrow C_3$

   Answer (Fill in the table below showing what happens as the transactions execute):

T_1	T_2	T_3	T_4	X	Y
1	2	3	4	RT=0	RT=0
				WT=0	WT=0
				C=true	C=true
	$R_2(X)$				

(b) (10 points) Consider the following schedule. Explain what happens when transactions try to execute as per this schedule and the DBMS uses **multiversion** concurrency control:

$ST_1 \rightarrow ST_2 \rightarrow ST_3 \rightarrow ST_4 \rightarrow R_1(X) \rightarrow R_3(X) \rightarrow W_3(X) \rightarrow R_2(X) \rightarrow R_4(X) \rightarrow W_2(X) \rightarrow W_4(X)$

Answer

(Fill in the table below showing what happens as the transactions execute):

T_1	T_2	T_3	T_4	X_0 ...
1	2	3	4	
$R_1(X)$				RT=1
...				