National University of Computer and Emerging Sciences, Lahore
Spring Semester 2014

Course: ADVANCED DATABASES Time Allowed: *90 min.*
Max Points: *50*

# Midterm 1

Section: ____ Name: _____ Roll No: _____

**Question 1** *(10 points)*

**a)** What are transaction commit points, and why are they important?

**b)** What is conflict equivalence and view equivalence?

**c)** Define the violations caused by each of the following: dirty read, non-repeatable read, and phantoms.

**d)** What do the terms steal/no-steal and force/no-force man with regard to buffer management for transactions processing?

**e)** Describe the wait-die and wound-wait protocols for deadlock prevention.

**Question 2** *(6 points)*

Consider the following sequence of actions, listed in the order it is submitted to the DBMS (S is a shared lock, X is an exclusive lock):

S1: $X_1(A)$, $S_2(C)$, $S_1(B)$, $S_2(B)$, $X_3(B)$, $X_2(A)$
S2: $S_1(A)$, $X_2(A)$, $X_3(B)$, $X_1(B)$, $S_3(A)$

For both the sequences S1 and S2 given above: Show the waits-for graph and indicate whether there will be a deadlock or not at the end of each sequence.

**Question 3** *(5 points)*

Consider the following classes of schedules: non-recoverable, *recoverable, cascadeless and strict.* For each of the following schedules, state which of the preceding classes it belongs to. Justify your answer. The actions are listed in the order they are scheduled.

**a)** S2: r1(X), w3(X), c3, w1(Y), c1, r2(Y), w2(Z), c2
**b)** S4: r1(X), w2(X), w1(X), a2, c1
**c)** S5: r1(X), w2(X), w1(X), c2, c1
**d)** S3: r1(X), w2(X), w1(X), r3(X), c1, c2, c3
**e)** S1: w1(X), r2(X), w1(X), c2, c1

**Question 4** *(10 points)*

Assume that the initial values of items are A=10, B=20, C=30. Given the following log of a recovery manager performing **deferred update (no undo/redo)**,

**a)** Identify which (if any) transactions need undo/ redo operation(s)?
**b)** Specify which operations in the log are redone (in correct order) and which (if any) are undone.
**c)** Write down the values of items A, B, and C after system recovery.

[start_transaction, T1]
[read_item, T1, B, 20]
[write_item, T1, B, 20, 15]
[start_transaction, T2]
[read_item, T2, A, 10]
[write_item, T2, A, 10, 5]
[read_item, T2, B, 20]
[write_item, T2, B, 20, 12]
[commit, T2]
[start_transaction, T3]
[read_item, T1, A, 5]
[write_item, T1, A, 5, 2]
[read_item, T3, C, 30]
[write_item, T3, C, 30, 25]
[checkpoint]
[commit, T3]
[start_transaction, T4]
[read_item, T4, C, 25]
[write_item, T4, C, 25, 22]
[start_transaction, T5]
[read_item, T5, C, 25]
[write_item, T5, C, 25, 21]
[commit, T5]
**System failure**

**Question 5** *(4 points)*

Consider the following schedule:

**S:** $R_1(X)$, $R_2(Y)$, $W_1(Z)$, $C_1$, $R_3(Y)$, $R_3(Z)$, $W_2(Y)$, $W_3(X)$, $C_2$, $C_3$

State which of the following concurrency control protocols allows it, that is,allows the actions to occur in exactly the order shown: Basic 2PL, Strict 2PL, Rigorous 2PL. Please provide a brief explanation for your answer...If YES, show where the lock requests could have happened; If NO, explain briefly

**Question 6** *(15 points)*

Consider the following schedule of actions, listed in the order they are submitted to the DBMS:

**S:** $R_1(X)$, $R_2(Y)$, $W_1(Z)$, $C_1$, $R_3(Y)$, $R_3(Z)$, $W_2(Y)$, $W_3(X)$, $C_2$, $C_3$

For each of the following concurrency control mechanisms, describe how the concurrency control mechanism handles the schedule.

Assume that the timestamp of transaction *Ti* is *i*. For lock-based concurrency control mechanisms, add lock and unlock requests to the above schedule of actions as per the locking protocol. The DBMS processes actions in the order shown. If a transaction is blocked, assume that all its actions are queued until it is resumed; the DBMS continues with the next action (according to the listed schedule) of an unblocked transaction.

**a)** Strict 2PL with timestamps used for deadlock avoidance (use wait-die policy)
**b)** Strict Timestamp Ordering
**c)** Optimistic Concurrency Control Technique (use Defer the validation option when required)