CSC742 Homework 7
Solution

Ex. 1

21.28 b
21.29 b
21.30 b
21.31 a
21.32 c
21.33 b
21.34 c
21.35 c
21.36 b
21.37 b

Ex. 2

- **S1: T1:R(X), T2:W(X), T2:W(Y), T3:W(Y), T1:W(Y), T1:Commit, T2:Commit, T3:Commit**

1). Strict 2PL with timestamps used for deadlock prevention.

Wait_die:

T1	**T2**	**T3**
Read_lock (X)		
Read_item (X)		
	TS (T2) > TS (T1): die	
	Abort (T2)	
		Write_lock (Y)
		Write_item (Y)
TS (T1) < TS (T3): wait		
		Commit
		Unlock (Y)
Write_lock (Y)		
Write_item (Y)		
Commit		
Unlock (X)		
Unlock (Y)		
	Restart	
	Write_lock (X)	
	Write_item (X)	
	Write_lock (Y)	
	Write_item (Y)	
	Commit	
	Unlock (X)	
	Unlock (Y)	

Wound-wait:

T1	T2	T3
Read_lock (X)		
Read_item (X)		
	TS (T2) > TS (T1): wait	
		Write_lock (Y)
		Write_item (Y)
TS (T1) < TS (T3): wound T3		
		Abort
		Unlock (Y)
Write_lock (Y)		
Write_item (Y)		
Commit		
Unlock (X)		
Unlock (Y)		
	Write_lock (X)	
	Write_item (X)	
	Write_lock (Y)	
	Write_item (Y)	
	Commit	
	Unlock (X)	
	Unlock (Y)	
		Restart
		Write_lock (Y)
		Write_item (Y)
		Commit
		Unlock (Y)

2). Strict 2PL with deadlock detection. (Show the waits-for graph if a deadlock cycle develops).

T1	T2	T3
Read_lock (X)		
Read_item (X)		
	wait	
		Write_lock (Y)
		Write_item (Y)
wait		
		Commit
		Unlock (Y)
Write_lock (Y)		
Write_item (Y)		
Commit		
Unlock (X)		
Unlock (Y)		
	Write_lock (X)	
	Write_item (X)	
	Write_lock (Y)	
	Write_item (Y)	
	Commit	
	Unlock (X)	

Unlock (Y)

3). Conservative (and strict, i.e., with locks held until end-of-transaction) 2PL.

T1	**T2**	**T3**
Read_lock (X)		
Write_lock (Y)		
Read_item (X)		
Write_item (Y)		
Commit		
Unlock (X)		
Unlock (Y)		
	Write_lock (X)	
	Write_lock (Y)	
	Write_item (X)	
	Write_item (Y)	
	Commit	
	Unlock (X)	
	Unlock (Y)	
		Write_lock (Y)
		Write_item (Y)
		Commit
		Unlock (Y)

4). Optimistic concurrency control.

T1	**T2**	**T3**
Read Phase		
Read_item (X)		
	Read Phase	
	Write_item (X) (Local Copy)	
	Write_item (Y) (Local Copy)	
		Read Phase
		Write_item (Y) (Local Copy)
Write_item (Y) (Local Copy)		
Validation (Success)		
Write Phase		
Commit		
	Validation (success)	
	Write Phase	
	Commit	
		Validation (Success)
		Write Phase
		Commit

5). Timestamp concurrency control with buffering of reads and writes (to ensure recoverability) and the Thomas Write Rule.

T1	**T2**	**T3**
Read_item (X) (Read_TS (X)=1)		

Write_item (X) (Write_TS (X)=2)
Write_item (Y) (Write_TS (Y)=2)

Write_item(Y) (Write_TS(Y)=3)

(Ignore Write (Y))
Commit

Commit

Commit

6). Multiversion concurrency control.

T1	**T2**	**T3**
Read_item (X)		
(Read_TS (X1)=1)		
	Write_item (X)	
	(Read_TS (X2)=Write_TS (X2)=2)	
	Write_item (Y)	
	(Read_TS (Y1)=Write_TS (Y1)=2)	
		Write_item (Y)
		(Read_TS (Y2)=Write_TS (Y2)=3)
Write_item (Y)		
(Read_TS (Y3)=Write_TS (Y3)=1)		
Commit		
	Commit	
		Commit

- **S2: T1:R(X), T2:W(Y), T2:W(X), T3:W(Y), T1:W(Y), T1:Commit, T2:Commit, T3:Commit**

1. Strict 2PL with timestamps used for deadlock prevention.

Wait-die

T1	T2	T3
Read_lock (X)		
Read_item (X)		
	Write_lock (Y)	
	Write_item (Y)	
	TS (T2) > TS (T1): die	
	Abort (T2)	
	Unlock (Y)	
		Write_lock (Y)
		Write_item (Y)
TS (T1) < TS (T2): wait		
		Commit
		Unlock (Y)
Write_lock (Y)		

Write_item (Y)
Commit
Unlock (X)
Unlock (Y)

Restart
Write_lock (Y)
Write_item (Y)
Write_lock (X)
Write_item (X)
Commit
Unlock (Y)
Unlock (X)

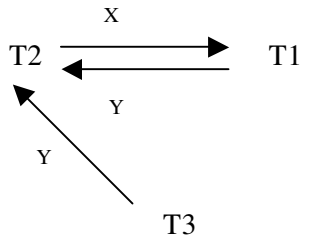
Wound-wait

T1	**T2**	**T3**
Read_lock (X)		
Read_item (X)		
	Write_lock (Y)	
	Write_item (Y)	
	TS (T2) > TS (T1): wait	
		TS (T3) > TS (T2): wait
TS (T1) < TS (T2): wound T2		
	Abort (T2)	
	Unlock (Y)	
Write_lock (Y)		
Write_item (Y)		
Commit		
Unlock (X)		
Unlock (Y)		
		Write_lock (Y)
		Write_item (Y)
		Commit
		Unlock (Y)
	Restart	
	Write_lock (Y)	
	Write_item (Y)	
	Write_lock (X)	
	Write_item (X)	
	Commit	
	Unlock (Y)	
	Unlock (X)	

2. Strict 2PL with deadlock detection. (Show the waits-for graph if a deadlock cycle develops).

T1	**T2**	**T3**
Read_lock (X)		
Read_item (X)		
	Write_lock (Y)	
	Write_item (Y)	
	Wait	

		wait
wait		
(Deadlock detected)		
	Abort (T2)	
	Unlock (Y)	
		Write_lock (Y)
		Write_item (Y)
		Commit
		Unlock (Y)
Write_lock (Y)		
Write_item (Y)		
Commit (T1)		
Unlock (X)		
Unlock (Y)		
	Restart	
	Write_lock (Y)	
	Write_item (Y)	
	Write_lock (X)	
	Write_item (X)	
	Commit	
	Unlock (Y)	
	Unlock (X)	



(Deadlock occurred.)

3. Conservative (and strict, i.e., with locks held until end-of-transaction) 2PL.

T1	**T2**	**T3**
Read_lock (X)		
Write_lock (Y)		
Read_item (X)		
Write_item (Y)		
Commit		
Unlock (X)		
Unlock (Y)		
	Write_lock (Y)	
	Write_lock (X)	
	Write_item (Y)	
	Write_item (X)	
	Commit	
	Unlock (Y)	

Unlock (X)

Write_lock (Y)
Write_item (Y)
Commit
Unlock (Y)

4. Optimistic concurrency control.

T1	**T2**	**T3**
Read Phase		
Read_item(X)		
	Read Phase	
	Write_item (Y) (Local Copy)	
	Write_item (X) (Local Copy)	
		Read Phase
		Write_item (Y) (Local Copy)
Write_item (Y) (Local Copy)		
Validation (Success)		
Write Phase		
Commit		
	Validation (success)	
	Write Phase	
	Commit	
		Validation (Success)
		Write Phase
		Commit

5. Timestamp concurrency control with buffering of reads and writes(to ensure recoverability) and the Thomas Write Rule.

T1	**T2**	**T3**
Read_item (X) (Read_TS (X)=1)		
	Write_item (Y) (Write_TS (Y)=2)	
	Write_item (X) (Write_TS (X)=2)	
		Write_item (Y) (Write_TS(Y)=3)
(Ignore Write (Y))		
Commit		
	Commit	
		Commit

6. Multiversion concurrency control.

T1	**T2**	**T3**
Read_item (X)		

(Read_TS (X1)=1)

Write_item (Y)
(Read_TS (Y1)=Write_TS (Y1)=2)
Write_item (X)
(Read_TS (X2)=Write_TS (X2)=2)

Write_item(Y)
(Read_TS (Y2)=Write_TS (Y2)=3)

Write_item (Y)
(Read_TS (Y3)=Write_TS (Y3)=1)
Commit

Commit

Commit

Ex. 3

1. Read record P1200: 5.
   D: IS
   F2: IS
   P1200: IS
   P1200: 5: S.

2. Read records P1200: 98 through P1250: 2.
   D: IS
   F2:IS
   P1200:IS
   P1250: IS
   P1201 through P1249: S
   P1200: 98: S; P1200:99 : S; P1200:100: S; P1250:1: S; P1250:2: S.

3. Read all (records on all) pages in file F1.
   D: IS
   F1: S.

4. Read pages P500 through P520
   D: IS
   F1: IS
   P500 through P520: S.

5. Read pages P10 through P980
   D: IS
   F1: S.

6. Read all pages in F1 and modify about 10 pages, which can be identified only after reading F1.
   D: IS and IX
   F1: SIX.

7. Delete record P1200: 98. (This is a blind write.)
   D: IX
   F2: IX

P1200: IX
P1200: 98: X.

8. Delete the first record from each page. (Again, these are blind writes)
   D: IX
   F1: X
   F2: X.
9. Delete all records.
   D: IX
   F1: X
   F2: X

Ex. 4 (20.23)

In wait-die, transactions only wait on younger transactions. So no cycle is created in the wait-for graph. Thus, there is no deadlock. Since a younger transaction requesting an item held by an older transaction is aborted and restart with the same timestamp, starvation is avoided.

In wound-wait, transactions only wait on older transactions so no cycle is created. Thus, there is no deadlock. An older transaction requesting an item held by an younger transaction preempts the younger transaction by aborting it. The younger transaction restart with the same timestamp, and starvation is avoided.

Ex. 5 (20.25)

For 19.8 (b)

Initial:	Read_TS (X) =0	Read_TS (Y) =0	Read_TS (Z) =0
	Write_TS (X) =0	Write_TS (Y) =0	Write_TS (Z) =0
	TS (T1) = 6	TS (T2) = 1	TS (T3) = 4

1. Read_TS (Z) = 1
2. Read_TS (Y) = 1
3. Write_TS (Y) = 1
4. Read_TS (Y) = 4
5. Read_TS (Z) = 4
6. Read_TS (X) = 6
7. Write_TS (X) = 6
8. Write_TS (Y) = 4
9. Write_TS (Z) = 4
10. Abort T2

The schedule is NOT allowed.

For 19.8 (c)

Initial:	Read_TS (X) =0	Read_TS (Y) =0	Read_TS (Z) =0

Write_TS (X) =0 Write_TS (Y) =0 Write_TS (Z) =0

TS (T1) = 3 TS (T2) = 7 TS (T3) = 1

1. Read_TS (Y) = 1
2. Read_TS (Z) = 1
3. Read_TS (X) = 3
4. Write_TS (X) = 3
5. Write_TS (Y) = 1
6. Write_TS (Z) = 1
7. Read_TS (Z) = 7
8. Read_TS (Y) = 3
9. Write_TS (Y) = 3
10. Read_TS (Y) = 7
11. Write_TS (Y) = 7
12. Read_TS (X) = 7
13. Write_TS (X) = 7

The schedule is allowed.

Ex. 6 (20.26)

For 19.8 (b)

1. Read_TS (Z1) = 1
2. Read_TS (Y1) = 1
3. Write_TS (Y2) = 1
   Read_TS (Y2) = 1
4. Read_TS (Y2) = 4
5. Read_TS (Z1) = 4
6. Read_TS (X1) = 6
7. Write_TS (X2) = 6
   Read_TS (X2) = 6
8. Read_TS (Y3) = 4
   Write_TS (Y3)= 4
9. Read_TS (Z2) = 4
   Write_TS (Z2) = 4
10. Read_TS (X1) = 6
11. Read_TS (Y3) = 6
12. Write_TS (Y4) = 6
    Read_TS (Y4) = 6
13. Abort (T2)

The schedule is NOT allowed.

For 19.8 (c)

1. Read_TS (Y1) = 1
2. Read_TS (Z1) = 1
3. Read_TS (X1) = 3

4. Write_TS (X2) = 3
   Read_TS (X2) =3
5. Write_TS (Y2) = 1
   Read_TS (Y2) = 1
6. Write_TS (Z2) = 1
   Read_TS (Z2) = 1
7. Read_TS (Z2) = 7
8. Read_TS (Y2) = 3
9. Write_TS (Y3) = 3
   Read_TS (Y3) = 3
10. Read_TS (Y3) = 7
11. Write_TS (Y4) = 7
    Read_TS (Y4) = 7
12. Read_TS (X2) = 7
13. Write_TS (X3) = 7
    Read_TS (X3) = 7

The schedule is allowed.