Name: _____

Midterm 2: CS186, Spring 2016

Class Login: cs186- _____ _____ _____

You should receive 1 double-sided answer sheet and an 11-page exam. Mark your name and login on ***both sides of the answer sheet****,* ***and in the blanks above****. For each question, place* ***only your final answer*** *on the answer sheet—do not show work or formulas there. You may use the blank spaces for scratch paper, but* ***do not tear off any pages****. You will* ***turn in both question and answer sheets****.*

I. Locking and Serializability [35 points]

1. [8 points] For the following statements, list the true statement(s) in alphabetical order.

   A. Schedules that are conflict serializable are also view serializable.
   B. Under two-phase locking, once a transaction releases a lock, it can no longer acquire any new locks.
   C. Schedules that are conflict serializable have to be produced by two-phase locking.
   D. Schedules produced by two-phase locking are guaranteed to prevent cascading aborts.
   E. Strict two-phase locking is both necessary and sufficient to guarantee conflict serializability.
   F. Wound-wait and wait-die algorithms are pessimistic deadlock avoidance algorithms and can cause more transaction aborts than needed.
   G. Under multi-granularity locking, locks should be acquired and released from the top level to the bottom level.
   H. Under multi-granularity locking, when a transaction T1 is holding an ‘IX’ lock on page A, it is possible for transaction T2 to hold a ‘S’ lock on record B of page A.

Consider the following schedule consisting purely of reads and writes on the tuple level ({A, B, C} are tuples). The meanings of operations are shown below:

W(A): transaction writes tuple A.
R(A): transaction reads tuple A.
COM: transaction commits.

	1	2	3	4	5	6	7	8	9	10	11	12	13
T1		W(A)						W(A)		COM			
T2	R(A)		R(B)		W(C)						COM		
T3				R(B)		W(C)						COM	
T4							R(C)		W(B)				COM

2. [4 points] Draw the dependency graph for this schedule.

3. [4 points] Of the following, list all conflict equivalent serial orderings of this schedule in alphabetical order:
    A. T2 T1 T4 T3
    B. T2 T3 T4 T1
    C. T2 T1 T3 T4
    D. T2 T4 T1 T3

4. [2 point] List one read that could occur between timestep 9 and 10 that would make this schedule not conflict serializable. For example, if T1 reading A is an answer, write “T1: R(A)”

Consider the following schedule. The meanings of operations are the same as in the previous problem. You may assume that locks could have been acquired or released at any time in between timesteps, and the scheduler may know all operations of transactions ahead of time.

	1	2	3	4	5	6	7	8	9	10
T1	W(A)	R(B)				W(C)	COM			
T2				W(A)	R(B)				COM	
T3			R(B)					W(B)		COM

5. [3 points] For the following statements, list the true statement(s) in alphabetical order.

   A. It is possible for this schedule to be produced by two-phase locking.
   B. It is possible for this schedule to be produced by strict two-phase locking.
   C. This schedule is guaranteed not to have cascading aborts.

Suppose that we have a table Y, with pages {A, B, C}, and tuples {A_1, A_2, A_3, …} in page A, and tuples {B_1, B_2, B_3, …} in page B, and tuples {C_1, C_2, C_3, …} in page C. Below is a schedule that correctly follows strict two-phase locking and multi-granularity locking protocols. For clarity, operations that are blocked (because of failing to acquire the lock) are italicized and underlined.

	1	2	3	4	5	6	7	8	9
T1	W(B_1)	R(A)							*W(C)*
T2					R(B_2)	*W(B)*			
T3								W(C_2)	
T4			R(C_1)	R(A_1)			*W(A_1)*		

For the following problems, assume that all transactions are still active (i.e. they have not yet committed/aborted) at timestep 9.

6. [4 points] What locks have been acquired on table Y at timestep 9? For each lock, list the transaction it belongs to, and its type. Do not list locks that are on the wait queue, and use locks with minimal privilege. For example, if T1 has an S lock, and T2 has an X lock, write "T1:S, T2:X".

7. [2 points] What locks have been acquired on page B at timestep 9? For each lock, list the transaction it belongs to, and its type. Do not list locks that are on the wait queue, and use locks with minimal privilege.

8. [6 points] There is a deadlock in this schedule (i.e. it is waiting indefinitely for locks).
   8.1. Draw the waits-for graph for this schedule.

   8.2. Which transactions are involved in the deadlock?

   8.3. Circle all true statements (on the answer sheet):
   A. We can resolve this deadlock by aborting any transaction involved in the deadlock.
   B. No transaction can proceed until this deadlock is resolved.

9. [2 points] Suppose T1 wishes to scan page B between timestep 8 and 9. Which locks are upgraded at this time? For example, if an S lock on A1 is upgraded to an X lock, write "S(A1) → X(A1)". If there are no upgrades, or if all upgrades are placed on the wait queue, write "None".

This space left intentionally blank for scratch work.

II. Query Optimization [26 points]

1. [6 points] Write down letters for all the correct statements (in alphabetical order)

   A. When evaluating potential query plans, the set of left deep join plans are always guaranteed to contain the best plan.
   B. As a heuristic, the System R optimizer avoids cross-products if possible.
   C. Considering all join orders and join methods, there are n! ways to join n tables.
   D. A plan can result in an interesting order if it involves a sort-merge join.
   E. The System R algorithm is greedy because for each pass, it only keeps the lowest cost plan for each combination of tables.
   F. If the statistics needed to compute the result size of a table are missing, the System R optimizer aborts.

For the following question, assume the following:

- the System R assumptions about uniformity and independence from lecture hold
- **primary key IDs are sequential, starting from 1**
- all of our tables are stored as heap files

We have the following schema:

`CREATE TABLE Flight (`
`fid INTEGER` **`PRIMARY KEY,`**
`from_id INTEGER` **`REFERENCES City,`**
`to_id INTEGER` **`REFERENCES City,`**
`aid INTEGER` **`REFERENCES Airline`**`)`
	NTuples: 100K, **NPages:** 50
	Index:
	(I) unclustered B+-tree on aid. 20 pages.
	(II) clustered B+-tree on (from_cid, fid). 10 pages.

`CREATE TABLE City (`
`cid INTEGER` **`PRIMARY KEY,`**
`name VARCHAR(16),`
`state VARCHAR(16),`
`population INTEGER)`
	NTuples: 50K, **NPages:** 20
	Index:
	(III) clustered B+-tree on population. 10 pages.
	(IV) unclustered index on cid. 5 pages.
	Statistics:
	state in [1, 50],
	population in [10^6, $8*10^6$]

`CREATE TABLE Airline (`
`aid INTEGER` **`PRIMARY KEY,`**
`hq_cid INTEGER` **`REFERENCES City,`**
`name VARCHAR(16))`
	NTuples: 5K, **NPages:** 2

2. Consider the following query:

```
SELECT *
FROM Flight F, City C, Airline A
WHERE F.to_cid = C.cid AND F.aid = A.aid
AND F.aid >= 2500
AND C.population > 5e6
AND C.state = 'California';
```

2.1. [4 points] Considering each predicate in the WHERE clause separately, what is the reduction factor for each?

i. R1 = C.state='California' _____/_____
ii. R2 = F.to_cid = C.cid _____/_____
iii. R3 = F.aid >= 2500 _____/_____
iv. R4 = C.population > $5 * 10^6$ _____/_____

2.2. [5 points] For each blank in the our System R DP table for Pass 1, choose the letter corresponding to the correct solution (Assume this is before the optimizer discards any rows it isn't interested in keeping).

Table(s)	Plans	Interesting Orders from Plan (N/A if none)	Cost (I/Os)
Flight	Index (I)	_____(i)_____	_____(ii)_____
City	Filescan	_____(iii)_____	_____(iv)_____
City	Index (III)	population	_____(v)_____

i.	A. N/A	B. aid	C. from_cid	D. to_cid
ii.	A. 100020*R2	B. 100020*(R2*R3)	C. 70*(R2*R3)	D. 70*R3
iii.	A. N/A	B. state	C. cid	D. population
iv.	A. 50020*(R1*R4)	B. 30*(R1*R4)	C. 30*R4	D. 20
v.	A. 50020*(R1*R4)	B. 30*(R1*R4)	C. 30*R4	D. 20

2.3. [3 points] For simplicity, let's assume each individual reduction factor (Ri) is 1/2. Then, what is the **READ** I/O cost of (City [Index (III)] JOIN Flights [File Scan]), using Page Nested Loops Join?

2.4. [3 points] After Pass 2, which of the following plans could be in the DP table?

A. City [Index(III)] JOIN Airline [File scan]
B. City [Index (III)] JOIN Flight [Index (I)]
C. Flight [Index (II)] JOIN City [Index (III)]

2.5. [5 points] Suppose we want to optimize for queries similar to the query in #2, which of the following suggestions could reduce I/O cost?

A. Maintain a more detailed histogram of Cities.population
B. Change Index (III) to be unclustered
C. Reduce the size of Airline.name field using string compression
D. Add a hash index on Flights.aid
E. Store City as a sorted file on population

This space left intentionally blank for scratch work.

III. Recovery [24 points]

1. [6 points] List all the true statements in alphabetical order:
   A. When a transaction commits, any modified buffer pages must be written to durable storage.
   B. When aborting a transaction, it may be necessary to modify pages on disk.
   C. During recovery, the ARIES protocol may redo aborted transactions.
   D. The pageLSN contains the LSN of the last operation to modify the page.
   E. The tail of the log is always flushed after every update operation.
   F. A system that uses a FORCE, STEAL policy does not need to undo any operations after a crash.

Questions 2 through 7 use the log below. Our database system is using ARIES for recovery. Some of the information for update and CLR log records is omitted for brevity. Adjacent log records are spaced exactly 10 LSN's apart. The system crashes after the last log record is written.

LSN	Record	prevLSN
0	update: T1 writes P1	null
10	update: T2 writes P2	null
20	update: T3 writes P1	null
30	update: T2 writes P3	10
40	begin_checkpoint	-
50	update: T1 writes P4	0
60	update: T2 writes P5	30
70	end_checkpoint	-
80	commit: T2	60
90	update: T3 writes P5	20
100	T2 end	80
110	T1 abort	50
120	CLR: T1 LSN 50	110

Transaction Table and Dirty Page Table recorded at end_checkpoint (LSN 70).

Transaction Table			Dirty Page Table	
Xact	**lastLSN**	**Status**	**PageID**	**recLSN**
T1	0	Running	P1	20
T2	30	Running	P3	30
T3	20	Running		

2. [1 point] The page updated by LSN 0 has been written to disk when the system starts recovering from the crash.

   A. True
   B. False
   C. Not enough information

3. [1 point] The page updated by LSN 60 has been written to disk when the system starts recovering from the crash.

   A. True
   B. False
   C. Not enough information

4. [1 point] What is the undoNextLSN of the CLR at LSN 120?

5. [6 points] Fill in the transaction table and dirty page table after Analysis. Do not add any transactions to the transaction table (it should only contain T1 and T3). You may not need to use all rows provided in the dirty page table.

6. [3 points] List the LSN's of the actions that are redone during REDO. Assume that nothing in the buffer pool was flushed to disk between begin checkpoint and the time of crash.

7. [6 points] What log records are written during UNDO? Fill in the remaining columns for the records below. You may not need to use all rows in the table.
   Hint: T1 and T3 are the only "loser" transactions.

This space left intentionally blank for scratch work.