IMPORTANT:

- **Upload this PDF** with your answers to **Gradescope by 11:59pm on Monday Nov 13, 2017**.
- **Plagiarism**: Homework may be discussed with other students, but all homework is to be completed **individually**.
- **You have to use this PDF for all of your answers**.

For your information:

- Graded out of **100 points**; **4 questions total**

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serializability and 2PL</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Deadlock Detection and Prevention</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Hierarchical Locking - A Blogging Website</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>B+ tree Locking</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>100</strong></td>
<td></td>
</tr>
</tbody>
</table>
Question 1: Serializability and 2PL.................................[20 points]

(a) Yes/No questions:
   i. [2 points] Schedules under Strict 2PL could have cascading aborts.
      □ Yes  □ No
   ii. [2 points] A conflict serializable schedule need not always be view serializable.
       □ Yes  □ No
   iii. [2 points] Schedules under Strict 2PL could have deadlocks.
        □ Yes  □ No
   iv. [2 points] There could be schedules under 2PL that are not serializable.
       □ Yes  □ No
   v. [2 points] If a precedence graph associated with a schedule has cycles, then it is not
      conflict serializable.
       □ Yes  □ No

(b) Serializability:
   Consider the schedule given below in Table 1. R(·) and W(·) stand for ‘Read’ and ‘Write’,
   respectively.

<table>
<thead>
<tr>
<th>time</th>
<th>t_1</th>
<th>t_2</th>
<th>t_3</th>
<th>t_4</th>
<th>t_5</th>
<th>t_6</th>
<th>t_7</th>
<th>t_8</th>
<th>t_9</th>
<th>t_10</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_1</td>
<td>R(A)</td>
<td>W(A)</td>
<td>W(E)</td>
<td>R(B)</td>
<td>W(B)</td>
<td>R(C)</td>
<td>W(C)</td>
<td>R(D)</td>
<td>W(D)</td>
<td></td>
</tr>
<tr>
<td>T_2</td>
<td></td>
<td>R(A)</td>
<td></td>
<td>W(A)</td>
<td>R(D)</td>
<td>W(D)</td>
<td>W(E)</td>
<td>R(A)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_3</td>
<td></td>
<td></td>
<td></td>
<td>R(C)</td>
<td>W(C)</td>
<td></td>
<td>R(F)</td>
<td>W(F)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: A schedule with 3 transactions

i. [1 point] Is this schedule serial?
      □ Yes  □ No

ii. [3 points] Give the dependency graph of this schedule. List each edge in the de-
        pendency graph like this: T_x → T_y because of Z (i.e., Transaction T_y reads/writes Z
        which was last written by T_x.). Order the edges in ascending order with respect to x.

iii. [1 point] Is this schedule conflict serializable?
       □ Yes  □ No

iv. [3 points] If you answer “yes” to (iii), provide the equivalent serial schedule. If you
       answer “no”, briefly explain why.

v. [2 points] Is this schedule possible under 2PL?
      □ Yes  □ No

Homework 5 continues...
Question 2: Deadlock Detection and Prevention…………………[30 points]

(a) Deadlock Detection:
Consider the following lock requests in Table 2. And note that
• S(·) and X(·) stand for ‘shared lock’ and ‘exclusive lock’, respectively.
• T₁, T₂, and T₃ represent three transactions.
• LM stands for ‘lock manager’.
• Transactions will never release a granted lock.

<table>
<thead>
<tr>
<th>time</th>
<th>t₁</th>
<th>t₂</th>
<th>t₃</th>
<th>t₄</th>
<th>t₅</th>
<th>t₆</th>
<th>t₇</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁</td>
<td>S(A)</td>
<td></td>
<td></td>
<td></td>
<td>S(B)</td>
<td>S(C)</td>
<td></td>
</tr>
<tr>
<td>T₂</td>
<td></td>
<td>S(B)</td>
<td></td>
<td></td>
<td>X(C)</td>
<td></td>
<td>X(B)</td>
</tr>
<tr>
<td>T₃</td>
<td></td>
<td></td>
<td>X(A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LM</td>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Lock requests of 3 transactions

i. **[3 points]** For the lock requests in Table 2, determine which lock will be granted or blocked by the lock manager. Please write ‘g’ in the LM row to indicate the lock is granted and ‘b’ to indicate the lock is blocked. For example, in the table, the first lock (S(D) at time t₁) is marked as granted.

ii. **[4 points]** Give the wait-for graph for the lock requests in Table 2. List each edge in the graph like this: Tₓ → Tᵧ because of Z (i.e., Tₓ is waiting for Tᵧ to release its lock on resource Z). Order the edges in ascending order with respect to x.

iii. **[3 points]** Determine whether there exists a deadlock in the lock requests in Table 2, and briefly explain why.

(b) Deadlock Prevention:
Consider the following lock requests in Table 3. Again,
• S(·) and X(·) stand for ‘shared lock’ and ‘exclusive lock’, respectively.
• T₁, T₂, T₃, and T₄ represent four transactions.
• LM represents a ‘lock manager’.
• Transactions will never release a granted lock.

i. **[3 points]** For the lock requests in Table 3, determine which lock request will be granted, blocked or aborted by the lock manager (LM), if it has no deadlock prevention policy. Please write ‘g’ for grant, ‘b’ for block and ‘a’ for abort. Again, example is given in the first column.

ii. **[4 points]** Give the wait-for graph for the lock requests in Table 3. List each edge in the graph like this: Tₓ → Tᵧ because of Z (i.e., Tₓ is waiting for Tᵧ to release its lock on resource Z). Order the edges in ascending order with respect to x.

---

Question 2 continues...
Table 3: Lock requests of 4 transactions

<table>
<thead>
<tr>
<th>time</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
<th>$t_6$</th>
<th>$t_7$</th>
<th>$t_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>S(A)</td>
<td></td>
<td>S(B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_2$</td>
<td>X(B)</td>
<td></td>
<td></td>
<td></td>
<td>X(D)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_3$</td>
<td></td>
<td>S(C)</td>
<td>X(D)</td>
<td>X(A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_4$</td>
<td></td>
<td></td>
<td></td>
<td>X(C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>g</td>
</tr>
</tbody>
</table>

iii. **[3 points]** Determine whether there exists a deadlock in the lock requests in Table 3, and briefly explain why.

iv. **[5 points]** To prevent deadlock, we use the lock manager ($LM$) that adopts the Wait-Die policy. We assume that in terms of priority: $T_1 > T_2 > T_3 > T_4$. Here, $T_1 > T_2$ because $T_1$ is older than $T_2$ (i.e., older transactions have higher priority). *Determine which lock request will be granted ('g'), blocked ('b') or aborted ('a').* Follow the same format as the previous question.

v. **[5 points]** Now we use the lock manager ($LM$) that adopts the Wound-Wait policy. We assume that in terms of priority: $T_1 > T_2 > T_3 > T_4$. Here, $T_1 > T_2$ because $T_1$ is older than $T_2$ (i.e., older transactions have higher priority). *Determine which lock request will be granted ('g'), blocked ('b') or aborted ('a').* Follow the same format as the previous question.
Question 3: Hierarchical Locking - A Blogging Website . . . . . . . . [30 points]

Consider a Database (D) consisting of two tables, Users (U) and Posts (P). Specifically,

- Users(uid, first_name, last_name), spans 300 pages, namely U1 to U300
- Posts(pid, uid, title, body), spans 600 pages, namely P1 to P600

Further, each page contains 100 records, and we use the notation $U_3:20$ to represent the 20th record on the third page of the Users table. Similarly, $P_5:10$ represents the 10th record on the fifth page of the Posts table.

We use Multiple-granularity locking, with S, X, IS, IX and SIX locks, and four levels of granularity: (1) database-level (D), (2) table-level (U, P), (3) page-level (U1 − U300, P1 − P600), (4) record-level (U1 : 1 − U300 : 100, P1 : 1 − P600 : 100).

For each of the following operations on the database, please determine the sequence of lock requests that should be generated by a transaction that wants to efficiently carry out these operations by maximizing concurrency.

Please follow the format of the examples listed below:

- write “IS(D)” for a request of database-level IS lock
- write “X(P2 : 30)” for a request of record-level X lock for the 30th record on the second page of the Posts table
- write “S(P2 : 30 − P3 : 100)” for a request of record-level S lock from the 30th record on the second page of the Posts table to the 100th record on the third page of the Posts table.

(a) [6 points] Fetch the 10th record on page P100.

(b) [6 points] In remembrance of our TA (RIP Christopher “Inf” Wallace), set the last_name of all the users to be “Inf”.

(c) [6 points] Scan all the records on pages P1 through P20, and modify the record $P_5 : 10$.

(d) [6 points] Order all the posts by their title.

(e) [6 points] Delete ALL the records from ALL tables.

Homework 5 continues...
Question 4: B+ tree Locking ...................................... [20 points]

Consider the following B+ tree:

![B+ tree diagram]

To lock this B+ tree, we would like to use the Bayer-Schkolnick algorithm. Important: we use the version as presented in the lecture, which does not use lock upgrade. For each of the following transactions, give the sequence of lock/unlock requests. For example, please write \( S(A) \) for a request of shared lock on node A, \( X(B) \) for a request of exclusive lock on node B and \( U(C) \) for a request of unlock node C.

Important notes:

- Each of the following transactions is applied on the original tree, i.e., ignore any tree changes described in other problems.
- For simplicity, ignore the changes on the pointers between leaves.

(a) [5 points] Search for data entry “15*”
(b) [5 points] Delete data entry “28*”
(c) [5 points] Insert data entry “9*”
(d) [5 points] Insert data entry “45*”