CIS560 – Database System Concepts **Name:_______________**

Homework Assignment 6 [15 points] – due October 25 (by midnight)

1. [5 points] After a system's crash, the undo-log using nonquiescent checkpointing contains the following data.

<START T1>
<T1, X1, 1>
<START CKPT ????>
<START T2>
<T2, X2, 2>
<T1, X1, 3>
<START T3>
<COMMIT T1>
<END CKPT>
<START CKPT ???>
<T2, X2, 4>
<T3, X3, 5>
<START T4>
<COMMIT T2>
<T4, X4, 6>
<COMMIT T3>
<END CKPT>
<START T5>
<T5, X5, 7>
<START CKPT ????>
<T4, X4, 8>
CRASH !!!

i. What are the correct values of the three <START CKPT ????> records? You have to provide three correct values for the three "????"s.

```
First: <START CKPT T1>
Second: <START CKPT (T2, T3)>
Third: <START CKPT (T4, T5)>
```

ii. Assuming that the three <START CKPT ????> records are correctly stored in the log, according to your answer at i., show which elements are recovered by the undo recovery manager and compute their values after recovery.

```
X4 = 6, X5 = 7
```

iii. Indicate what fragment of the log the recovery manager needs to read.

Recovery manager must read up to `<START CKPT (T2, T3)> [ we can actually stop at <START T4> because we know that the other two transactions have committed].`

2. [5 points] After a system's crash, the redo-log using nonquiescent checkpointing contains the following data.

`<START T1>`
`<T1, A, 10>`
`<START T2>`
`<T2, B, 5>`
`<T1, C, 7>`
`<START T3>`
`<T3, D, 12>`
`<COMMIT T1>`
`<START CKPT ????>`
`<START T4>`
`<T2, E, 5>`
`<COMMIT T2>`
`<T3, F, 1>`
`<T4, G, 15>`
`<END CKPT>`
`<COMMIT T3>`
`<START T5>`
`<T5, H, 3>`
`<START CKPT ????>`
`<COMMIT T5>`
`CRASH !!!`

i. What are the correct values of the two <START CKPT ????> records? You have to provide two correct values for the two ????s.

```
First: < START CKPT (T2, T3) >
Second: < START CKPT (T4, T5) >
```

ii. Indicate and explain what fragment of the log the recovery manager needs to read.

Since the second START CKTP does not have an associated END CKPT, we cannot be sure that committed transactions prior to the start of this checkpoint had their changes written to disk. Thus, we must search for the previous checkpoint. In the previous START CKPT, T2 and T3 were the two active transactions. Both transactions committed and must thus be redone. T2 was the first one to start. The recovery manager must thus read the log record starting from < START T2 > and must read until the end of the log file.

iii. Assuming that the two < START CKPT ??? > records are correctly stored in the log, according to your answer above, show which elements are recovered by the redo recovery manager and compute their values after recovery.

We must redo the changes made by all committed transactions there were either active during the first START CKPT or that started after that point. T2 and T3 were active during the first START CKPT and committed. T4 and T5 started after the checkpoint but only T5 committed. We must thus redo the changes by T2, T3, and T5. Elements B, D, E, F, and H are thus recovered. Their values after recovery are as follows:

B=5
D=12
E=5
F=1
H=3

3. [5 points] The SuperSQL database system stores its undo log file in a table, with the following schema:

Log(N, T, E, V)

where **N** is the entry number (0, 1, 2, 3, …), **T** is the transaction id, **E** is the element id, and **V** is the old value. A log entry of the form <T, E, V> is simply represented by the tuple (N, T, E, V), where N is the entry number and E>0 for an ordinary element id. The log entries <START T>, <COMMIT T>, and <ABORT T> are represented by a tuple (N, T, E, null), where E=-1 for START, E=-2 for COMMIT, and E=-3 for ABORT. For example, the log:

<START T1>
<T1, X1, 55>
<START T2>
<T2, X2, 99>
<COMMIT T1>
. . . .

is represented by the table:

N	T	E	V
0	T1	-1	
1	T1	X1	55
2	T2	-1	
3	T2	X2	99
4	T1	-2	
. . .			

Recall that each transaction starts and ends at most once; for example, a sequence <START T> ... <COMMIT T> ... <START T> ... will not occur in the log. Moreover, any update by the transaction T will occur between its <START T> and <COMMIT T>, or between <START T> and <ABORT T> respectively. Finally, once a transaction has ended in COMMIT or ABORT and the corresponding log record is on disk, the transaction has completed and does not need to be undone.

Write a SQL query that can be run during database recovery, after a system crash. The answer to your query should return a table with two attributes, **E**, and **V**, indicating which elements have to be updated with what values. You should include each element **E** at most once in your answer: otherwise it would be ambiguous how to update it.

Here are a few solutions for the query:

```
 Select L1.E, L1.V
 From Log L1
 Where L1.T not in
               (Select distinct L3.T
                From Log L3
                Where L3.E in (-2, -3))
       and L1.N <= all (Select L2.N
                    From Log L2
                    Where L2.E = L1.E and L2.T not in
                           (Select distinct L3.T
                            From Log L3
                            Where L3.E in (-2, -3)))

      and L1.E <>'-1'
```

```
SELECT E, V
FROM Log y
WHERE V IS NOT NULL
and N = (SELECT MIN(x.N) FROM Log x WHERE x.E = y.E and x.T NOT IN ( SELECT
c.T FROM Log c WHERE c.E = -2 OR c.E = -3)  )
and T NOT IN ( SELECT c.T FROM Log c WHERE c.E = -2 OR c.E = -3)
```