**1: Recovery and ARIES**

1. For the actions listed above, show Transaction Table (XT) and Dirty Page Table (DPT) after each action. Assume that DPT holds pageID and recLSN, and XT contains transID and lastLSN.

* Transaction Table

|  |  |
| --- | --- |
| transID | lastLSN |
| T1 | 80 |
| T0 | 10 |
| T2 | 60 |

* DPT

|  |  |
| --- | --- |
| pageID | recLSN |
| P7 | 00 |
| P9 | 10 |
| P8 | 20 |
| P6 | 60 |
| P5 | 70 |

1. Usually the Analysis phase starts scanning log from the where the last checkpoint ends.

* Transaction Table

|  |  |
| --- | --- |
| transID | lastLSN |
| T1 | 80 |
| T2 | 60 |

* DPT

|  |  |
| --- | --- |
| pageID | recLSN |
| P9 | 50 |
| P6 | 60 |
| P5 | 70 |
| P7 | 80 |

* LAST ENTRY HAS P6 FLUSHED

1. Simulate Redo phase: first identify where the Redo phase starts scanning the log records. Then, for each action identify whether it needs to be redone or not.

* Start from smallest LSN 50 (smallest recLSN in DPT)

|  |  |
| --- | --- |
| **LSN** | **Action** |
| 50 | Add (T1,50) to XT and (P9,50) to DPT |
| 60 | Add (T2,60) to XT and (P6,60) to DPT |
| 70 | Change (T1,50) to (T1,70) in XT and Add (P5,70) to DPT |
| 80 | Change (T1,70) to (T1,80) and Add (P7,80) |

|  |  |
| --- | --- |
| **LSN** | **Action** |
| 50 | Changes to P9 are redone |
| 60 | Changes to P6 are redone |
| 70 | Changes to P5 are redone |
| 80 | Changes to P7 are redone |

1. Simulate Undo phase: identify all actions that need to be undone. In what order will they be undone?

* The Undo phase starts from the lastLSN (starting from the end of the log) which will be 80 in this case

|  |  |
| --- | --- |
| **lastLSN** | **Action** |
| 90 | Undo (T1,80) |
| 100 | Undo (T1,70) |
| 110,115 | Undo (T2,60) and end T2 |
| 120 | Undo (T1,50) and end T1 |

**2: Recovery**

1. Assuming the data is already sorted, we first take the primary key of one relation and the foreign key of the other relation and compare them to create a set of data which will have the join functionality. Although I have coded only till the join is executed into a join.csv file. But I assume these would be the further steps:

I would have used the flag system, once there is a join it would change the flag of that join to one and the remaining are by default 0. After there is a crash it will restart the recovery process and will find the transaction with the changed flag which were executed before the server failed and then continue with all the transaction with the flag 0.

1. The program creates a join.csv file with the joined data but couldn't figure out how to do the crash and continuation part. I hope I get the partial grades for creating the join, to run enter “g++ main.cpp -o main.out” .  
   Running the "main.out" file would create a csv file named join.csv.