The assignment is to be turned in before Midnight (by 11:59pm) on February 27. You should turn in the written parts of the solutions as a single pdf file through the TEACH website. The written solutions should be produced using editing software programs, such as LaTeX or Word, otherwise they will not be graded. You should turn in the source code and executable files for the programming assignment through the TEACH website.

1: Recovery and ARIES (4 points)

In this problem, you need to simulate the actions taken by ARIES algorithm. Consider the following log records and buffer actions:

time	LSN	Log	Buffer actions
0	00	update: T1 updates P7	P7 brought in to the buffer
1	10	update: T0 updates P9	P9 brought into the buffer; P9 flushed to disk
2	20	update: T1 updates P8	P8 brought into the buffer; P8 flushed to disk
3	30	begin_checkpoint	
4	40	end_checkpoint	
5	50	update: T1 updates P9	P9 brought into the buffer
6	60	update: T2 updates P6	P6 brought into the buffer
7	70	update: T1 updates P5	P5 brought into the buffer
8	80	update: T1 updates P7	P6 flushed to disk
9		CRASH RESTART	

(a) 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.

(b) Simulate Analysis phase to reconstruct XT and DPT after crash. Identify the point where the Analysis phase starts scanning log records and show XT and DPT after each action.

(c) 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.

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

2: Recovery (6 points)

Consider the following relations:

```
Dept (did (integer), dname (string), budget (double), managerid (integer))
Emp (eid (integer), ename (string), age (integer), salary (double))
```

Fields of types *integer*, *double*, and *string* occupy 4, 8, and 40 bytes, respectively. Each block can fit at most one tuple of an input relation. There are at most 22 blocks available to the join algorithm in the main memory. Assume that relations *Dept* and *Emp* are sorted according to attributes *managerid* and *eid*, respectively.

(a) Explain a join algorithm that efficiently computes $Dept \bowtie_{Dept.managerid=Emp.eid} Emp$ by

merging these relations. We do *not* want to repeat the join from the beginning if a crash happens during the join computation. Instead, your algorithm must avoid recomputing the joint tuples that have been computed before the crash and continue the join computation after the database system restarts. The final output of your algorithm must be exactly the same as the result of the join as if no crash has happened during the join computation. You should explain the steps that your algorithm will follow during the normal execution of join and the techniques that it uses after a restart.

(b) Implement your proposed algorithm in C++.
Requirements:

- Each input relation is stored in a separate CSV file, i.e., each tuple is in a separate line and fields of each record are separated by commas.
- The result of the join must be stored in a new CSV file. The files that store relations Dept and Emp are Dept.csv and Emp.csv, respectively.
- Your program must assume that the input files are in the current working directory, i.e., the one from which your program is running.
- The program must store the result in a new CSV file with the name join.csv in the current working directory.
- Your program must run on Linux. Each student has an account on *voltdb1.eecs.oregonstate.edu* server, which is a Linux machine. You may use this machine to test your program if you do not have access to any other Linux machine. You can use the following *bash* command to connect to *voltdb1*:

  ```
  > ssh your_onid_username@voltdb1.eecs.oregonstate.edu
  ```

  Then it asks for your ONID password and probably one another question. You can only access this server on campus.
- You can use following commands to compile and run C++ code:

  ```
  > g++ main.cpp -o main.out
  > main.out
  ```