Data Management – AA 2014/15 – exam of 20/2/2015

Problem 1
Let `R(A,B)` and `S(B,C)` be two relations (sets of tuples), stored in 180.000 pages, and 60.000 pages, respectively. Note that `B` is a key both in the relation `R` and in the relation `S`. Suppose our buffer has 500 frames available. Describe in detail (step by step) the algorithm you would use to compute the natural join of `R` and `S`, and tell which is the cost of the algorithm in terms of page accesses (ignoring, as usual, the cost of producing the output).

Problem 2
A schedule is said to be "symmetric" if the precedence graph G associated to it satisfies the following conditions:

- every node in G has at least one outgoing edge;
- it is symmetric, meaning that for each edge $\langle n_i, n_j \rangle$ in G, the edge $\langle n_j, n_i \rangle$ is also in G.

Let S be a symmetric schedule on transactions $T_1, \ldots, T_n$, such that every transaction T_i has only "write" actions. Prove or disprove that S is not view serializable.

Problem 3
Consider the following schedule S, where we assume that the timestamp of each transaction T_i is i:

$$r_1(x)\, r_2(y)\, w_1(y)\, w_3(t)\, r_3(x)\, c_3\, w_2(t)\, c_1\, c_2\, w_4(y)\, w_4(z)\, a_4\, w_5(z)\, r_5(v)\, c_5\, w_6(v)\, c_6$$

You are asked to:

- Illustrate the various steps carried out by the timestamp-based scheduler when analyzing the above schedule.
- Tell whether the above schedule is conflict serializable.
- Tell whether the above schedule is in 2PL with shared and exclusive locks.
- Tell whether the above schedule is in ACR.

Problem 4
Consider the relation `R` with 15.000.000 tuples, where each tuple has 4 attributes. We assume that every attribute and every pointer has the same size. We know that 10 tuples of `R` fit in one page. Consider a sorted index `I` using alternative 2 for `R`, with one attribute as a search key, which is also a key of the relation. Tell which is the number of page accesses required for searching for a tuple in `R` given a value of the search key, both in the case where `I` is dense and in the case where `I` is sparse.

Problem 5
Consider the relation `Driving(dcode,ccode,time,numcrash,grade)`, where `dcode` and `ccode` form a key, which stores information about how much time a driver (`dcode`) has used a car (`ccode`), how many crashes he had using that car, and which is the grade that the driver assigned to the car. The most frequent queries posed to such relation are (`C` is a constant):

Query 1	**Query 2**
select `dcode,ccode` from `Driving`	select `time,numcrash,grade` from `Driving` where `dcode=C`

We know that the size of every field (value or pointer) is 10B, every page has 900B, the number of tuples of `Driving` is 36.000.000, and in the average every driver has used 20 cars. Tell which method would you choose for representing the relation in secondary storage, taking into account that you can define at most one index, and that your goal is to execute the above queries efficiently. Also, for each query, tell which algorithm would you choose for executing the query, and how many page accesses would be needed for computing the answer to the query, given the chosen algorithm.