PART B (BASIC)

1. (6 pts.) For each of the following hypothetical rules about functional dependencies, either show that it holds or give a counterexample. Each of A, B, C, and D stands for a single attribute.

   (a) If $AB \rightarrow C$ and $C \rightarrow B$, then $A \rightarrow B$.

   (b) If $A \rightarrow B$ and $BC \rightarrow D$, then $AC \rightarrow D$.

   Answer: For (a), a relation like

A	B	C
0	1	2
0	2	3

   suffices.

   For (b), we may compute $(AC)^+$ with respect to the given dependencies and note that it contains D.

2. (9 pts.) Suppose we have a relation $D(doc, word)$, which has those tuples (d, w) such that word w appears in document d. Write an expression of relational algebra for the set of pairs of documents (d_1, d_2) such that every word in document d_1 also appears in document d_2. To help write the expresssion, you may use a sequence of assignment steps of the form $T(a_1, a_2, \ldots, a_k)$:= E, where T is a temporary relation, and E is a relational algebra expression with a small number of operators. We shall accept that the resulting expression has attributes named $a_1, a_2, \ldots, a_k$, so you need not worry about renaming attributes. However, **you must explain** your work by telling what tuples belong in each of the temporary relations you define (a brief comment suffices).

   Answer:

   1) $R_1(d, w) \;:=\; (\pi_{doc}(D) \times \pi_{word}(D)) - D$
   2) $R_2(d1, d2) \;:=\; \pi_{d,doc}(D \underset{w=word}{\bowtie} R_1)$
   3) $R_3(d1, d2) \;:=\; D \times D$
   4) $R_4(d1, d2) \;:=\; R_3 - R_2$

   Explanation: R_1 is all docuemnt-word pairs such that the word is *not* in the document. R_2 is all document pairs (d_1, d_2) such that there exists a word w that is in d_1 but not in d_2. R_3 is all document pairs, and R_4, the answer, is all document pairs such that the first has *no* word that is not in the second.

PART B (ADVANCED)

3. (12 pts.) Suppose relations R and S have no duplicates, and are stored on 1000 and 500 disk blocks, respectively. Tuples of R and S are in no particular order, e.g., they cannot be assumed sorted.

   (a) We wish to compute $S - R$ using an algorithm that reads each block of R and S to a main-memory buffer only once. Describe an algorithm that uses as few buffers as possible. How many buffers do you need?

   (b) How would you modify your algorithm to compute $R - S$, again reading each block of R and S only once, while using no more buffers than for part (a)?

   (c) Suppose you only had 50 main-memory buffers. How would you compute $R - S$ to minimize the number of disk I/O's performed?

   Answer: For (a), copy S into memory. Then read each block of R, one-at-a-time, into the same buffer. For each tuple of R, "kill" the identical tuple of S if it exists. At the end, output all the tuples of S that have not been killed. The total buffer use is 501 blocks.

   For (b), again copy S into buffers, and read R one block at a time. Now, for each tuple of R, if it is in S, ignore it. If it is not in S, output it. Again, 501 buffers are used.

   For (c), create 10 sorted sublists of S and 20 sorted sublists of R, each by using all 50 buffers and doing an in-memory sort. Then, use 30 buffers to house one block of each sorted sublist, and perform a "merge." When tuple t is the least among all the buffers, if t is in R but not in S, then output t, otherwise ignore t. Remove all instances of t at the fronts of the sublists, and proceed to the next tuple in order.

4. (6 pts.) Suppose we are given predicates $red(X, Y)$ and $blue(X, Y)$, meaning that there is a red arc from node X to node Y or a blue arc from X to Y, respectively. Write a datalog program for the predicate $p(X, Y)$ meaning that there is a path from node X to node Y with no more than two red arcs in a row. Explain your answer by giving the intuitive meaning of each IDB predicate that you use.

   Answer:

```
zero(X,Y) :- blue(X,Y)
zero(X,Y) :- any(X,Z) & blue(Z,Y)

one(X,Y) :- red(X,Y)
one(X,Y) :- zero(X,Z) & red(Z,Y)

any(X,Y) :- zero(X,Y)
any(X,Y) :- one(X,Y)
```

   Intuitively, any legal path must either end in a blue arc or in a single red arc (i.e., either the path is of length 1, or the previous arc is blue). Thus, *zero* represents paths that end in blue; *one* represents paths that end in a single red, and *any* represents the union of these two kinds of paths. The answer is *any*.

5. (6 pts.) Suppose we have the following relation schemas and instances: $AB = \{a_1b_1\}$; $AC = \{a_1c_1, a_2c_2\}$; $BC = \{b_1c_1\}$. Let there also be the functional dependencies $A \rightarrow B$ and $B \rightarrow C$. Give the representative instance for this database. Show your reasoning for partial credit.

Answer: The weak instance formed from these relations is

A	B	C
a_1	b_1	$\perp_1$
a_1	$\perp_2$	c_1
a_2	$\perp_3$	c_2
$\perp_4$	b_1	c_1

When we chase with respect to the given FD's and eliminate subsumed rows, we get the representative instance:

A	B	C
a_1	b_1	c_1
a_2	$\perp_3$	c_2

6. (6 pts.) Below are three conjunctive queries:

```
C1: p(X,Y) :- q(X,Z) & r(Z,Y) & q(Y,Z)
C2: p(A,E) :- q(A,B) & r(B,C) & q(C,D) & r(D,E)
C3: p(F,H) :- q(F,F) & q(G,G) & r(F,G) & r(G,H)
```

Indicate all the containments between any two of these queries. Show your reasoning for partial credit.

Answer: The only containments are $C1 \subseteq C2$ and $C3 \subseteq C2$. For the first, the mapping $A \rightarrow X$, $B \rightarrow Z$, $C \rightarrow Y$, $D \rightarrow Z$, and $E \rightarrow Y$ suffices. For the second, use: $A \rightarrow F$, $B \rightarrow F$, $C \rightarrow G$, $D \rightarrow G$, and $E \rightarrow H$.