GENERAL INSTRUCTIONS

- Answer each question in a separate book.
- Indicate on the cover of each book the area of the exam, your code number, and the question answered in that book. On one of your books list the numbers of all the questions answered. Return all answer books in the folder provided. Additional answer books are available if needed.
- **Do not write your name on any answer book.**
- Answer **all four (4)** questions. Before beginning to answer a question make sure that you read it carefully. If you are confused about what the question means, state any assumptions that you have made in formulating your answer.
- The grade you will receive for each question will depend on both the correctness of your answer and the quality of the writing of your answer.

Policy on misprints and ambiguities: The Exam Committee tries to proofread the exam as carefully as possible. Nevertheless, the exam sometimes contains misprints and am- biguities. If you are convinced a problem has been stated incorrectly, mention this to the proctor. If necessary, the proctor can contact a representative of the area to resolve problems during the first hour of the exam. In any case, you should indicate your inter- pretation of the problem in your written answer. Your interpretation should be such that the problem is nontrivial.

QUESTION 1: JOIN ALGORITHMS

Consider the equijoin of two relations R and S, as in the SQL query:

```
SELECT *
FROM R, S
WHERE R.A = S.B
```

Furthermore, suppose that R is sorted on the A attribute, and S is sorted on the B attribute. You may find that some of the following is under-specified; please explicitly list any assumptions you think you need to make in your answer.

(1) Give the merge-join algorithm for such a problem (the equijoin of two relations sorted on their join attribute.)

(2) While the merge algorithm in Part 1 is generally fast, it is somehow not satisfying when there are large "gaps" in the matching tuples. So, for example, suppose the i-th tuple in R is r_i, and the j-th tuple in S is s_j. Suppose that r_i does not match $s_{(j+1)}$, and furthermore $r_{(i+1)}$ does not match any tuple in S until s_k, where k is substantially larger than $j+1$. Then the merge join from Part 1 will compare $r_{(i+1)}$ with many S tuples with no output. It would be nice if the algorithm could skip to the next match. Fortunately, there is a data structure that lets you do this: a B+-tree index. So suppose there are indexes on $R.A$ and $S.B$, modify your algorithm from Part 1 to "skip" to the next match. Note that the opportunities for skipping can be symmetric and your algorithm should handle this.

(3) How would you expect the performance of your new algorithm to compare with the performance of hybrid hash join?

(4) Now suppose that S has no index on it. You would now have the opportunity to first build an index on S and then do your new skipping merge join. How would you expect this index-building algorithm to compare with hybrid hash join?

QUESTION 2: DATALOG Consider the following Datalog program:

$$\texttt{t(X,Y)} :- \texttt{b(X,Y)}.$$
$$\texttt{t(X,Y)} :- \texttt{a(X,Z)}, \texttt{t(Z,Y)}.$$

(1) Give the magic sets rewriting of this program for the goal `t(1,Y)`

(2) In addition to Magic Sets, people talk about Naive vs. SemiNaive evaluation as options for evaluating recursive Datalog programs. If we begin with Naive evaluation of the program above without the Magic Sets rewriting, we can make the evaluation more efficient by changing from the Naive evaluation of the program without the Magic Sets rewriting to the SemiNaive evaluation of the program without the Magic Sets rewriting. We can also make the evaluation more efficient by moving from the Naive evaluation of the original program to the Naive evaluation of the program with the Magic Sets rewriting.

Describe a data set (that is, instances for a and b) such that moving from the Naive evaluation of the original program to the SemiNaive evaluation of the original program is expected to improve efficiency more than moving from the Naive evaluation of the original program to the Naive evaluation of the Magic Sets rewriting of the original program.

QUESTION 3: DATA EXCHANGE

Let S be a relational database with two tables:

- `HOUSES(location,price,agent-id)`
- `AGENTS(id,name,city,state,fee-rate)`

Let T be a relational database with one table:

- `LISTINGS(area,list-price,agent-address,agent-name)`

These databases describe house listings for sale. A sample tuple of `HOUSES` is `(Atlanta, GA; 360,000; 32)`. A sample tuple of `AGENTS` is `(32; Mike Brown; Athens; GA; 0.03)`. A sample tuple of `LISTINGS` is `(Denver, CO; 550,000; Boulder, CO; Laura Smith)`. The list price in T is computed by multiplying the price in S with $(1 + \texttt{fee-rate})$.

(1) Suppose we want to copy all data from S to T. Write a single SQL query that when executed over database S would transform all data of S into the format of T. That is, the query would create tuples for table `LISTINGS` of T from the data in S.

(2) In practice, writing such SQL queries to copy data from one database to another is very time consuming. To save time, a user can employ a schema matching tool to find semantic matches between S and T. Examples of such matches are: `location = area` and `name = agent-name`. The user then employs another tool to elaborate these matches into SQL queries (that can then be executed to copy data).

Using these tools, can the above process be completely automated? If yes, why? If not, why not? In that latter case, discuss at which points in the process the user must be involved, what the user must do, and why.

(3) Suppose the user has copied data from the database S to the database T, and has also copied data from the database T to another database U. While doing this, the user has established that attribute x of S matches y of T, and attribute y of T matches z of U. Can the user conclude that attribute x of S matches z of U? If yes, why? If not, why not?

QUESTION 4: QUERY EQUIVALENCE

Consider the following Conjunctive Query q: `q(X) :- R(X,Y), S(Y,Z), T(Z,X), T(Z,W)`

(1) Find an equivalent CQ to q that is minimal. Explain why the query you find is equivalent to q.

(2) Suppose that we are also given the following two views over `R,S,T`:

- `v1(A,B,C,D) :- R(A,B),S(C,D)`
- `v2(D,E) :- T(D,E)`

Express q using only the two views v1, v2 (and not any of the base tables `R,S,T`).