This exam is a take-home, open-book, open-notes exam. You are free to use not only your notes and the textbook, but also supplementary material in the form of books, papers, and documents on the Web. However, all work must be done individually and all used resources must be prominently acknowledged in your submission. If you are unsure if something is allowed, please check with me.

Two very important criteria in evaluating your submission are *clarity* and *rigor*. A technically correct answer that is poorly written or that uses dubious reasoning will earn very few points, if any, so please pay particular attention to these criteria.

You may submit your exam in either handwritten or typeset form. However, if you chose to type your submission, please make sure all mathematical notation is properly typeset. A neat handwritten submission is much better than a poorly typeset one. You may submit your hardcopy to Ellen in NV 237, noting the time of submission on the submission itself. For electronic submission, use the procedure outlined in the homework assignments.

1. (1 pt.) Write your name on your submission.

2. (4 pts.) Explain the difference, if any, between the terms *sort* and *type* as used by the textbook.

3. (5 pts.) Prove or disprove: Conjunctive queries are monotonic and satisfiable over finite domains.

4. (10 pts.) Prove or disprove: The running time of the *improved seminaive algorithm* is no greater than that of the *basic seminaive algorithm*.[1] Assume that the principal determinant of the running time is the number of facts generated by rule evaluations, including repeated generation of the same fact by one or many rules.

5. (10 pts.) Provide a detailed example of a Datalog program and a database instance for which the improved seminaive algorithm performs at least an order of magnitude less work than does the basic seminaive algorithm. Provide the detailed traces of both algorithms on the input example, showing all steps. Describe how your example may be generalized to inputs of arbitrary sizes.

6. (10 pts.) Provide either (a) a bound on the ratio of the running times of the basic and improved seminaive algorithms (on the same input) or (b) a method that, given an arbitrary number N, computes an input for which this ratio is greater than N. The term input refers to a Datalog program and a database instance on which the program is evaluated using the two methods. The bound may be expressed using any reasonable characterization of the input.

[1] Serge Abiteboul, Richard Hull, and Victor Vianu, *Foundations of Databases* (Addison-Wesley, 1995). 315–316..

7. (10 pts.) In the *reverse same generation* example,[2] the order of the subgoals in the body of the rule for rsg^{fb} was altered from that in the rule for rsg^{bf}. Demonstrate the impact of this change by providing a detailed trace of the QSQR algorithm on both versions of the adorned rules on the sample instance $\mathbf{I}_0$ used by the example.[3]

8. (20 pts.) Provide the details of an algorithm that takes a *nr-Datalog*$^{\neg}$ program (non-recursive Datalog with negation and a single output IDB *ans*) as input and produces an equivalent *named algebra*[4] query as output. Describe your algorithm using pseudocode at a level of detail that would permit a typical Computer Science junior to easily implement the algorithm. Explain why the algorithm is correct and state, and prove, its time and space complexity.

9. (10 pts.) Consider a query language L that is identical to the SPJR algebra except that L does not permit unary singleton constants. Describe the expressive power of L as precisely as possible (and justify your answer).

10. (20 pts.) Consider a relation *SvcLoc(station, milemark)* that describes the locations of service stations along a long interstate highway. A tuple (s, m) in this relation indicates that the station with name s (a string) is located near mile-marker m (a floating-point number) on this highway. Consider another relation *Stops(id, milemark)* that similarly describes the locations of planned stops along this highway. A tuple (i, m) in this relation indicates that a stop, identified by i (a string), is planned near mile-marker m.

    (a) Write a standard SQL query to produce a relation *StopSvc(id, station)* that pairs each stop with the nearest service station. That is, this relation contains a tuple (i, s) for each stop identifier i in *Stops* such that s is the name of the service station nearest stop i, based on distance along the highway, computed using the mile-markers.

    (b) Characterize the performance of the above query using any reasonable characterization of the input (such as the cardinalities of the input relations, the distributions of mile-marker values, the presence of any indexes, and the plan chosen by the query optimizer).

    (c) Provide brief experimental support for your claims by conducting simple experiments using the PostgreSQL system.

[2] *Idem*, p. 320.
[3] *Idem*, p. 312.
[4] *Idem*, p. 71.