CS345A Final Examination

Friday June 3, 1994, 12:30–3:15 pm

Directions

The exam is *open book*; any written materials may be used. Answer all 12 questions on the exam paper itself. The total number of points is 300. You may make additional comments at the bottoms of pages or at the end if you feel the need to clarify or justify your answers. Do not forget to **sign the pledge** below.

I acknowledge and accept the honor code.

____

Print your name here: ____

1	2	3	4
5	6	7	8
9	10	11	12

Problem 1: (30 points) Consider the following stratified rules:

```
minpath2(X,Y) :- arc(X,Z) & arc(Z,Y) & not arc(X,Y)
minpath3(X,Y) :- arc(X,Z) & arc(Z,W) & arc(W,Y) &
                    not arc(X,Y) & not minpath2(X,Y)
```

a) What are the strata?

b) Suppose the EDB consists of the following *arc* tuples: $arc(1,2)$, $arc(2,3)$, and $arc(3,4)$ (i.e., the graph looks like $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$). Give the stratified (perfect) model for this EDB and rules. You can state only the IDB, since the EDB is fixed.

c) For the same EDB as in part (b), find another minimal model for the EDB and rules.

Problem 2: (35 points) Consider the rules

```
p(X) :- a(X,Y) & not q(Y)
q(X) :- a(X,Y) & not p(Y)
```

Let the EDB consist of the "cycle" of tuples $a(1,2)$, $a(2,3)$, and $a(3,1)$.

a) Write below all the instantiated rules that do not have a false EDB subgoal. You may omit true EDB subgoals from the instantiated rules.

b) In the space below, draw the dependency graph for the instantiated IDB predicates.

c) Are the rules with this EDB locally stratified? ________________ Give the strata if so, and give a cycle if not.

d) These rules may or may not be locally stratified, depending on the EDB. Give an example of an EDB where the answer to "are these rules and EDB locally stratified?" is different from your answer to part (c).

Problem 3: (25 points) For the rules and EDB $\{a(1,2),\ a(2,3),\ a(3,1)\}$ given in Problem 2:

a) Give all stable models.

b) Give the well-founded model.

Problem 4: (20 points) The following rules and EDB are related to those of Problem 2 by the addition of a "basis" rule for p.

```
p(X) :- b(X)
p(X) :- a(X,Y) & not q(Y)
q(X) :- a(X,Y) & not p(Y)
```

The EDB consists of the same three tuples for a: $\{a(1,2),\ a(2,3),\ a(3,1)\}$ and the tuple $b(1)$ for EDB predicate b. In the table on the next page, use the alternating fixedpoint method to find the well-founded model for these rules and EDB.

$p(1)$	
$p(2)$	
$p(3)$	
$q(1)$	
$q(2)$	
$q(3)$	

Problem 5: (15 points) Suppose that R and S are two binary relations, and we define ternary relation T to be the union of the joins of R and S in either order, i.e.,

$$T(X,Y,Z) = \big(R(X,Y) \bowtie S(Y,Z)\big) \cup \big(S(X,Y) \bowtie R(Y,Z)\big)$$

Write a fragment of GLUE code that will set T to be the desired relation, assuming that relations R and S are already available.

Problem 6: (10 points) Suppose that there is a unary EDB relation $a(X)$ that has as tuples some constants that are the names of binary relations. Write a HiLog-style rule that defines $p(X,Y)$ to be the union of all the relations named in a.

Problem 7: (25 points) For the two conjunctive queries

Q_1: **panic** `:- a(X,Y) & a(Y,X) & a(X,X) & a(Y,Y)`
Q_2: **panic** `:- a(A,B) & a(B,C) & a(C,D)`

a) Give a containment mapping from Q_2 to Q_1.

b) Briefly explain why there is no containment mapping from Q_1 to Q_2.

c) How many containment mappings from Q_2 to Q_1 are there? ______________

Problem 8: (15 points) Consider the conjunctive query

```
Q: p(X,Y) :- arc(X,Z) & arc(Z,Y)
```

and the datalog program P defined by the rules:

```
p(X,Y) :- p(X,Z) & p(Z,W) & p(W,Y)
p(X,Y) :- arc(X,Y)
```

Is $Q \subseteq P$? ______________ Briefly explain your answer. No credit will be given for a correct answer without explanation.

____

____

____

____

Problem 9: (30 points) Multiple choice. Circle the letter of the correct answer. Score = 3 times number right minus number wrong.

a) Wich of the following is a *disadvantage* of using virtual memory addresses as object ID's?

(A) OID's are typically 32 bits long.
(B) Swizzling is avoided.
(C) OID's can be treated as integers.
(D) It is hard to find objects.

b) Which of the following is a *disadvantage* of using physical (secondary-storage) addresses as OID's?

(A) OID's are typically longer than 32 bits.
(B) It is hard to insert objects.
(C) It is hard to delete objects.
(D) It is hard to find objects.

c) A purpose of IDL stubs in the CORBA architecture is:

(A) To translate ORB OID's into the ID's of an object implementation.
(B) To validate accesses to objects.
(C) To answer queries about the schema.
(D) To optimize method invocations.

d) A predicate in CORAL can be:

(A) A GLUE procedure.
(B) A C++ procedure.
(C) Defined in Prolog.
(D) None of the above.

e) Which of these BLOB's is most likely to require a piece stream for efficient implementation?

(A) An encrypted password.
(B) The source text for a book being edited.
(C) A stored video clip available for viewing.
(D) The code for an operating system kernel.

f) Which of the following is *not* a problem with multiple inheritance?

(A) Method names can be ambiguous.
(B) Attribute names can be ambiguous.
(C) We cannot compute the offsets of attributes in a record representing the object.
(D) None of the above.

g) Which of the following CORBA components is not accessed *directly* by client (application) programs?

(A) ORB interface.
(B) IDL stubs.
(C) Dynamic invocation interface.
(D) Object adapters.

h) In a functional query language:

(A) All attributes are implemented by functions.
(B) Attributes are referred to as if they were functions.
(C) All queries are written as functions.
(D) OID's are implemented as if they were functions.

i) Which of the following is *not* a reason for implementing objects by property lists?

(A) OID's can be swizzled.
(B) Attributes can have variable-length values.
(C) Attributes can have sets of objects as values.
(D) Attributes can have bags (multisets) of objects as values.

j) An advantage of database programming languages over extended database systems is:

(A) Database programming languages support persistent data.
(B) Database programming languages are (typically) computationally complete.
(C) No conversion between data models is needed.
(D) Database programming languages (typically) support ad-hoc queries.

Problem 10: (15 points) Write a GLUE procedure `eval()` to compute the stratified model for the following rules.

```
p(X) :- a(X) & not q(X)
q(X) :- b(X) & not r(X)
r(X) :- c(X,Y)
```

You may assume that all predicates mentioned are EDB relations declared elsewhere in the module. You need not write the complete module — just the procedure.

Problem 11: (40 points) Given the logic program P_1:

```
p(X,Y) :- a(X,Y)
p(X,X) :- a(X,Y) & p(Y,Y)
```

with query $p(0, Z)$, we might evaluate the query by first rewriting P_1 as the equivalent program P_2:

r_1: `p(X,Y) :- a(X,Y)`
r_2: `p(X,X) :- a(X,Y) & q(Y)`
r_3: `q(X) :- a(X,X)`
r_4: `q(X) :- a(X,Y) & q(Y)`

a) Give two benefits of making this transformation:

(1)

(2)

b) Apply the basic magic-sets transformation to the program P_2, repeated here as:

r_1: `p(X,Y) :- a(X,Y)`
r_2: `p(X,X) :- a(X,Y) & q(Y)`
r_3: `q(X) :- a(X,X)`
r_4: `q(X) :- a(X,Y) & q(Y)`

c) There are a number of simplifications that can be made to the magic rules from (b). Below, rewrite the rules to remove any predicates that are defined by only one rule and used only once. Also, eliminate any predicates that are defined by a single rule with a single subgoal, no matter how many times the predicate is used. (The foregoing conditions are the standard reasons for eliminating predicates as discussed in Section 13.4 of the text.)

Problem 12: (40 points) We wish to use ODL to define two classes ("interfaces") *Account* and *Depositor*. The following facts need to be taken into account in your specification.

1. Accounts have a *balance*, which is a real number, and a set of *holders*. The holders of an account are *Depositor*'s.
2. *Depositor*'s have a *name*, which is a string, and an *address*, which is a pair of strings. *Depositor*'s also have a set of *accounts*, which are the *Account*'s of which the depositor is a holder.
3. There are no indexes desired for *Account* or *Depositor*; hence no extents or keys need to be declared.
4. *Depositor* is a subclass of class *Person*.
5. There is an operation *overdrawn* that takes an *Account* as argument and returns a Boolean, presumably true iff the account has a negative balance.
6. There is an operation *assets* that takes a *Depositor* as argument and returns a bag of reals, presumably the bag of balances on the accounts held by the depositor.

Write your specification on the next page.