CSE636 Data Integration - Fall 2008

Assignment #3 - Due Wednesday, November 19th – 12:00pm (in class)

NAME:_____

- Please write your solutions in the spaces provided using a pen, **not a pencil**. Make sure your solutions are neat and clearly marked.
- *Simplicity and clarity of solutions will count.* You may get as few as 0 points for a problem if your solution is far more complicated than necessary, or if we cannot understand your solution.

Problem 1 (28 points)

Suppose we have EDB relations

```
frequents(Drinker,Bar)
serves(Bar,Beer)
likes(Drinker,Beer)
```

The first indicates the bars a drinker visits; the second tells what beers each bar serves, and the last indicates which beers each drinker likes to drink. Define the following predicates using safe datalog rules.

---

a. (7 points) **`happy(D)`** that is true if drinker **`D`** frequents at east one bar that serves a beer he likes.

```
happy(D) :- likes(D, R), serves(B, R), frequents(D, B)
```

---

b. (7 points) **`shouldVisit(D,B)`** if bar **`B`** serves a beer drinker **`D`** likes.

```
shouldVisit(D, B) :- serves(B, R), likes(D, R)
```

---

c. (7 points) **`veryHappy(D)`** if every bar that drinker **`D`** frequents serves at least one beer he likes. You may assume that every drinker frequents at least one bar.

```
tmpDrinker(D) :- frequents(D,B), not(shouldVisit(D,B))
veryHappy(D) :- happy(D), not(tmpDrinker(D))
```

d. (7 points) **`sad(D)`** if drinker **`D`** frequents no bar that serves a beer he likes.

```
sad(D) :- tmpDrinker(D), not(happy(D))
```

Problem 2 (28 points)

Assume an undirected graph is represented as a set of facts of the form **`node(x)`** for a node **`x`**, and **`edge(x,y)`** and **`edge(y,x)`** for an edge **`{x,y}`**. A graph is *connected* if for every two different nodes in the graph there is a path between them. A node **`x`** is an articulation point of a graph if (1) the graph is connected, and (2) the graph with **`x`** and its incident edges removed is no longer connected.

a. (7 points) Write a Datalog¬ (Datalog with negation) program P_1 that checks whether a given graph is connected.

```
node(X) :- edge(X,_)
node(X) :- edge(_,X)
path(X,Y) :- node(X), node(Y), edge(X,Y)
path(X,Y) :- node(X), node(Z), edge(X,Z), node(Y), path(Z,Y)
notConnected() :- node(X), node(Y), not path(X,Y)
connected() :- not notConnected()
```

b. (7 points) Write a Datalog¬ program P_2 that returns the set of articulation points of a given graph.

```
canAvoid(X,Y,P) :- node(X), node(Y), node(P),
                   X ≠ P, Y ≠ P, edge(X,Y)
canAvoid(X,Y,P) :- node(X), node(Y), node(P), node(Z),
                   X ≠ P, Y ≠ P, Z ≠ P,
                   edge(X,Z), canAvoid(Z,Y,P)
artPoint(P) :- connected(), node(X), node(Y), node(P),
               X ≠ P, Y ≠ P, not canAvoid(X, Y, P)
```

c. (7 points) Explain why the program P_2 is stratified.

It is stratified because there are no negative edges in the cycles of the dependency graph. Cycles originate only in **canAvoid** and **path**, and neither involves negation.

d. (7 points) Can the program P_2 be written without using negation?

It cannot be written without negation because it requires non-monotonic logic, i.e., adding new edges can remove the property of being an articulation point.

Problem 3 (20 points)

Determine the containment/equivalence relationship between the following pairs of queries. Show your work. Whenever one query is contained in another, show **all** containment mappings. Whenever containment does not hold in a certain direction, give a counterexample to containment.

---

a. (10 points)

Q1: `p(X,Y,Z) :- e(X,Y',Z'), e(X,Y,Z''), e(X',Y',Z'), e(X,Y'',Z)`

Q2: `p(A,B,C) :- e(A,B,D), e(A,E,C)`

Q2 contains Q1:

$A \rightarrow X$

$B \rightarrow Y$

$C \rightarrow Z$

$D \rightarrow Z''$

$E \rightarrow Y''$

Q1 contains Q2:

$X \rightarrow A$

$Y \rightarrow B$

$Z \rightarrow C$

$Y' \rightarrow B$

$Z' \rightarrow D$

$Z'' \rightarrow D$

$X' \rightarrow A$

$Y'' \rightarrow E$

Hence, Q1 and Q2 are equivalent.

---

b. (10 points)

Q1: `p(X,Y) :- a(X,Z), a(Z,W), a(W,Y), a(Y,U), a(U,X)`

Q2: `p(X,Y) :- a(X,Y), X ≥ Y`

Q1 does not contain Q2:

a	
X	Y
2	1

Q2 does not contain Q1:

a	
X	Y
1	2
2	3
3	4
4	5
5	6

Problem 4 (24 points)

Consider a Local-As-View integration system that has one global predicate **p**, and the following two views:

```
v(X,Y) :- p(X,Y), p(Y,Z)
w(U,V) :- p(W,U), p(U,V)
```

The query to be answered is:

```
q(A,B) :- p(A,C), p(C,B)
```

Find the maximally contained rewriting of query Q using the MiniCon algorithm. Make sure that you show the following:

1. The MiniCon descriptions (MCDs) and the containment mappings for each one
2. All possible rewritings, where the views are not unfolded
3. Optimize the rewritings, both individually and collectively, and give the final maximally contained rewriting

1. MiniCon Descriptions (MCDs):

View	Mappings	Subgoals Covered
v	Q.A → v.X Q.C → v.Y	1
v	Q.B → v.Y Q.C → v.X	2
w	Q.A → w.U Q.C → w.V	1
w	Q.B → w.V Q.C → w.U	2

2. Rewritings:

R1: `q(A,B) :- v(A,C), v(C,B)`
R2: `q(A,B) :- v(A,C), w(C,B)`
R3: `q(A,B) :- w(A,C), v(C,B)`
R4: `q(A,B) :- w(A,C), w(C,B)`

3. Final Rewriting:

```
q(A,B) :- v(A,C), v(C,B)
q(A,B) :- v(A,C), w(C,B)
q(A,B) :- w(A,C), v(C,B)
q(A,B) :- w(A,C), w(C,B)
```