A: SIZE BOUNDS FOR JOINS [25%]

1. [15%] Compute the maximum possible output size for the following join queries. Assume that all relations have the same size $N$.

   (a) $q(x, y, z) : - R(x), T(y), U(z), S(x, y, z)$.
   
   (b) $q(x, y, z, w, t) : - R(x, y), S(y, z), T(z, w), U(w, x), V(x, t)$.
   
   (c) $q(x, y, z) : - R(x, y, z), S(x, z, w), T(x, y, w), U(y, z, w)$.

2. [10%] Suppose that relation $R_i$ has size $N_i$ (in tuples). Compute the maximum output size for the following query. (Hint: there will be different cases depending on the given $N_i$).

   $q(x_1, x_2, x_3, x_4, x_5) : - R_1(x_1, x_2), R_2(x_2, x_3), R_3(x_3, x_4), R_4(x_4, x_5)$.

B: DATALOG [75%]

1. [20%] Suppose $P$ is some boolean property of graphs (represented as an edge relation $E(x, y)$) that can be defined by a Datalog program. For example, $P$ can be: the graph $G$ has a cycle of odd length. Show that if $G$ is a graph that satisfies $P$, then:

   (a) every supergraph of $G$ also satisfies $P$.
   
   (b) if $h$ is a graph homomorphism, then $h(G)$ also satisfies $P$.

2. [20%] A Datalog program $P$ with a single recursive predicate is said to be bounded if there is a positive integer $n_0$ such that, on every database instance $I$, the bottom-up evaluation of $P$ terminates within at most $n_0$ steps. Otherwise, we say that $P$ is unbounded.

   (a) Prove that transitive closure is unbounded.
   
   (b) Give an example of a Datalog program that is bounded and has at least one recursive predicate.

3. [10%] Consider the following Datalog program:
\begin{verbatim}
T(x,y) :- R(x,y).
T(x,t) :- T(x,y), T(y,z), T(z,w), R(w,t).

Can you write an equivalent linear Datalog program? If yes, provide the program; otherwise, explain why this is not the case.

4. [20\%] Perform the magic set transformation for the following Datalog program:

\begin{verbatim}
T(x,y) :- F(x,y).
T(x,y) :- up(x,z1), T(z1,z2), F(z2,z3), T(z3, z4), down(z4,y).
q(y)   :- T(a,y).
\end{verbatim}

5. [5\%] Find all possible stratifications for the following Datalog program with negation:

\begin{verbatim}
T(x)   :- S(x), not R(x).
S(x)   :- T(x), not R(x).
U(x)   :- R(x), not T(x), not S(x).
V(x,y) :- V(x,y), not U(x), not U(y), .
\end{verbatim}
\end{verbatim}

**Deliverables**

Submit a single PDF document using Learn@uw (Homework 2). It is strongly suggested to use \LaTeX to write your solution.