Open Book.

*Attach extra pages as needed. Please, write neatly.*
Problem 1

Let $P$ be the following program:

$$c \leftarrow a, \neg c.$$  
$$a \leftarrow \neg b.$$  
$$b \leftarrow \neg a.$$  

(1) Give all the models and minimal models for $P$.  
(2) Find all the fixpoints and minimal fixpoints for $P$.  
(3) Find all stable models for $P$.  
(4) Compute $T_P \uparrow \omega(\emptyset)$.

Answer: Discussed in Class.
Problem 2: 24 points

You are given the database table:

student(Name char(10), Course char(10), Quarter char(10), Grade char(10))

1. Write an ATLaS program to return the name of each student with the courses he/she took and a progressive sequence number for each different course taken by the student—thus repeated courses are ignored. For instance, say that Joe, in his first quarter as a freshman, took cs11 and pe10, then, in the next quarter, he repeated cs11, and also took pe30. Then, your program should return:

   joe   cs11   1
   joe   pe10   2
   joe   pe30   3
   . . .

Answer: Discussed in Class.

2. Is the aggregate function that you expressed monotonic w.r.t. the student relation?
   Answer: it is obviously monotonic with respect to set containment.

3. Is the program that you wrote blocking?
   Yes, if the UDA aggregate you wrote returns all the results at the end.
You have the following query:

\[
\text{?append([a, b], [c, d], X)} \\
\text{append([], X, X).} \\
\text{append([H|T], X, [H|Y]) } \leftarrow \text{ append(T, X, Y).}
\]

A Show the binding passing analysis, and explain why the magic set method is applicable here, while left/right linear rules are not. (Hint, you can view the second rule as follows:
\[
\text{append(Z, X, W) } \leftarrow Z = [H|T], \text{ append(T, X, Y), W = [H|Y].}
\]

B Show the rewritten rules generated by the supplementary magic method.

C Show the content of the supplementary magic relation at the end of the computation.

Answer: Discussed in Class.
Minimal Model vs. Stable Model

For the program below, write a minimal model that is not a stable model.

\[
\begin{align*}
\text{plant}(X) \leftarrow & \quad \text{live}\_\text{organism}(X), \neg \text{animal}(X). \\
\text{live}\_\text{organism}(\text{oak}). \\
\text{animal}(\text{horse}). & \quad \text{animal}(\text{dog}). \\
\text{animal}(\text{cat}). & \quad \text{animal}(\text{eagle}).
\end{align*}
\]

**Answer:** Discussed in Class.
Problem 4: Fibonacci

1. Write an efficient Datalog program to check whether a given integer is a Fibonacci number. (Hint, for efficiency, you should use linear rules.)
Answer: linear rules to compute a value not to exceed the given max.

\[
\text{isfib}(\text{Given}; \text{Given}; \text{Fib1})
\]

\[
\text{isfib}(\text{Max}, 1, 1) \leftarrow 1 < \text{Max}
\]

\[
\text{isfib}(\text{Max}, \text{Fib}, \text{Fib1}) \leftarrow \text{isfib}(\text{Max}, \text{Fib1}, \text{Fib2}), \text{Fib} = \text{Fib1} + \text{Fib2}, \text{Fib1} < \text{Max}.
\]

2. Answer: The goal \text{isfib}^{bf} the recursive rule with the same pattern. But only the first argument is bound in the recursive predicate in the body. Then we get the pattern \text{isfib}^{bff} that becomes a stable pattern—i.e., the program has the binding passing property, where \text{isfib}^{bff}. Then the program is compiled by specializing the left-linear rules—i.e., by replacing \text{Max} with \text{Given}. Finally the delta transformation is used. Thus the transformed program is as follows:

\[
\text{isfib}(\text{Given}, \text{Fib}, \text{Fib1})
\]

\[
\delta \text{isfib}(\text{Given}, 1, 1) \leftarrow 1 < \text{Given}
\]

\[
\delta \text{isfib}(\text{Given}, \text{Fib}, \text{Fib1}) \leftarrow \delta \text{isfib}(\text{Given}, \text{Fib1}, \text{Fib2}), \text{Fib} = \text{Fib1} + \text{Fib2}, \text{Fib1} < \text{Given}.
\]
\textbf{p Until q:} \( p(1, 6) \) denotes that property \( p \) holds from point 1, included, till point 6, excluded.

From the bottom. \texttt{cep} (covered end of \( p \)) is true if \( K \) is covered by another interval. \texttt{break(I1, J2)}: \( I1 \) is the beginning of an interval, and \( J2 \) is the end of an interval to its right. Between them there is an interval ending with a \( K \) that is not covered. Where there is no such a break, then we have a coalesced interval starting at \( I1 \) and ending at \( J2 \). Given a \( q(I, _) \) \( p \cup q \) the coalesced \( p \) yields an interval spanning from 0 to \( I \).

\begin{align*}
p &\cup q(\textbf{yes}) \leftarrow q(0, J). \\
p &\cup q(\textbf{yes}) \leftarrow \texttt{coalscp}(0, I), q(J, _), I \geq J. \\
\texttt{coalscp}(I1, J2) \leftarrow p(I1, J1), p(I2, J2), J1 < J2, \\
&\neg \texttt{break(I1, J2)}. \\
\texttt{break(I1, J2)} \leftarrow p(I1, J1), p(I2, J2), p(_, K), \\
&J1 \leq K, K < I2, \neg \texttt{cep(K)}. \\
\texttt{cep}(K) \leftarrow p(_, K), p(I, J), I \leq K, K < J.
\end{align*}