CSE-6490B: Assignment #1

1. (5 points) **First-order Predicate Calculus.** *In theory, at least.*

   Consider first-order theory (program) $\mathcal{T}$.

   a. Is it possible that $\mathcal{T} \models \mathit{student}\,(\mathit{parke})$ *and* $\mathcal{T} \models \neg\mathit{student}\,(\mathit{parke})$?

      If it is possible, provide an example $\mathcal{T}$ such that it is the case. Otherwise, explain why it is not possible.

   b. Is it possible that $\mathcal{T} \not\models \mathit{student}\,(\mathit{parke})$ *and* $\mathcal{T} \not\models \neg\mathit{student}\,(\mathit{parke})$?

      If it is possible, provide an example $\mathcal{T}$ such that it is the case. Otherwise, explain why it is not possible.

   Consider a datalog database $\mathcal{D}$.

   c. Is it possible that $\mathcal{D} \models \neg\mathit{student}\,(\mathit{parke})$?

      Why or why not?

   d. Do $\mathcal{D} \vdash \neg\mathit{student}\,(\mathit{parke})$ $\mathcal{D} \not\vdash \mathit{student}\,(\mathit{parke})$ mean the same thing?

      Why or why not?

   e. Is every first-order propositional theory ("program") equivalent to some CNF propositional theory ("program")?

      If no, show an example of a first-order theory that cannot be written in CNF. If yes, are there any drawbacks to representing first-order theories in CNF?

2. (5 points) **Rules.** *Bar-flies.* (Thanks to Zahir Tari.)

   Suppose that we have the following predicates.

   - *frequents* (*Drinker, Bar*): The drinker frequently visits this bar.
   - *serves* (*Bar, Beer*): The bar serves this type of beer.
   - *likes* (*Drinker, Beer*)The drinker likes this type of beer.

   Define the following predicates via rules using the predicates above (and any that you define).

   a. *happy* (*D*): The drinker *D* frequents at least one bar which serves a beer that he / she likes.
   b. *very_happy* (*D*): Every bar that the drinker *D* visits serves at least one beer he / she likes.
   c. *should_visit* (*D, B*): The bar *B* serves at least one beer that the drinker *D* likes.
   d. *sad* (*D*): No bar that the drinker *D* visits serves a beer that he / she likes.
   e. *very_sad* (*D*): No bar serves a beer that he / she likes.

   You may assume that each drinker at least frequents one bar. Make certain that your rules are safe.

3. (5 points) **Queries in Datalog & Datalog¬.** *Enrol now in Datalog U.!* EXERCISE

Consider the following schema.

student(s#, sname, dob, d#)
    FK (d#) refs **dept** // Student's major
prof(p#, pname, d#)
    FK (d#) refs **dept** // Professor's home deparmtent
dept(d#, dname, building, p#)
    FK (p#) refs **prof** // Department's chair
course(d#, no, title)
    FK (d#) refs **dept** // Course offered by this deparmtent
class(d#, no, term, year, section, room, time, p#)
    FK (d#, no) refs **course** // Class is an offering of this course
    FK (p#) refs **prof** // Instructor of class
enrol(s#, d#, no, term, year, section, grade)
    FK (s#) refs **student** // This student is enrolled in
    FK (d#, no, term, year, section) refs **class** // this class

'FK' above stands for *foreign key.* These indicate foreign-key constraints in the schema.

Write the following queries in Datalog (and Datalog¬). You may use auxiliary predicates and rules. (You may reuse auxiliary predicates and rules in following sub-questions.)

A common convention is to use '_' as a variable name when the variable is unimportant for the query; e.g., *class* (*D*, *N*, _, _, _, _, _, _). By convention, two occurrences of '_' are *different* variables and may take on different values (even though they seem to have the same "name"). You may find this convention useful.

Be careful that all your rules are *safe*, including rules that you write that use negation.

a. Which students have taken some course twice?
b. Which students have taken a course with *a* department chair?
   Note that a professor may teach classes outside of his or her department. Also note that a student may take classes in a department outside of his or her major's department.
c. Which students have never taken a course in his or her major (dept)?
d. Which students have taken all of the courses offered by a department?
e. Which students have taken at least five courses in their major (dept)?
   You shall need to use arithmetics (e.g., '$\neq$', '$<$') here. Assume that course numbers (no) can be compared; e.g., $M < N$. Use the predicate *is* to equate numbers; e.g., J is $I + 1$.

4. **Datalog Modeling.** *As easy as rolling off a log.* (5 points)

The puzzle *Sū Doku*—or just *sudoku*—is to fill in the blank cells of a 9×9 matrix with the numerals 1, . . . , 9 such that no row has the same numeral twice, no column has the same numeral twice, and no *block* has the same numeral twice. The 9×9 matrix is tiled by nine 3×3 matrices, each called a block.

A typical sudoku puzzle has some of the cells already filled in (the *givens*) so that there exists exactly one solution. For example,

5	**3**			**7**				
6			**1**	**9**	**5**			
	9	**8**					**6**	
8				**6**				**3**
4			**8**		**3**			**1**
7				**2**				**6**
	6					**2**	**8**	
			4	**1**	**9**			**5**
				8			**7**	**9**

$\Rightarrow$

5	**3**	4	6	**7**	8	9	1	2
6	7	2	**1**	**9**	**5**	3	4	8
1	**9**	**8**	3	4	2	5	**6**	7
8	5	9	7	**6**	1	4	2	**3**
4	2	6	**8**	5	**3**	7	9	**1**
7	1	3	9	**2**	4	8	5	**6**
9	**6**	1	5	3	7	**2**	**8**	4
2	8	7	**4**	**1**	**9**	6	3	**5**
3	4	5	2	**8**	6	1	**7**	**9**

with the solution shown on the right.

Write a Datalog program for sudoku. Let each cell in the sudoku matrix be represented by a variable:

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}
X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}
X_{27}	X_{28}	X_{29}	X_{30}	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}
X_{36}	X_{37}	X_{38}	X_{39}	X_{40}	X_{41}	X_{42}	X_{43}	X_{44}
X_{45}	X_{46}	X_{47}	X_{48}	X_{49}	X_{50}	X_{51}	X_{52}	X_{53}
X_{54}	X_{55}	X_{56}	X_{57}	X_{58}	X_{59}	X_{60}	X_{61}	X_{62}
X_{63}	X_{64}	X_{65}	X_{66}	X_{67}	X_{68}	X_{69}	X_{70}	X_{71}
X_{72}	X_{73}	X_{74}	X_{75}	X_{76}	X_{77}	X_{78}	X_{79}	X_{80}

Do not use negation, arithmetics (e.g., "$\neq$", "$<$"), or function symbols (which are not in Datalog proper anyway).

One predicate should be *sudoku* that takes arguments $X_0, \ldots, X_{80}$. For a given puzzle, one could then query for the solution; e.g.,

$$\leftarrow \textit{sudoku}\,(5,\ 3,\ X_2,\ X_3,\ \ldots,\ 9).$$

Note that your sudoku "program" need not be efficient in any way. It just needs to *specify* logically and correctly the problem. So try to keep it quite simple. You may use ellipses (e.g., "$\ldots$", "$\vdots$") where appropriate and when easily understood to make your answer briefer.

Hint: Be clever in defining the permutations of 1, . . . , 9.