CSC 351L – Comparative Programming Languages Lab
Project 1 – Prolog Introduction
Due: February 18, 2021 – 11:59 pm

Purpose:

The objectives of these exercises are for you to become familiar with the Prolog system you will be using, including its trace/debug features, and for you to practice writing Prolog rules and querying the Prolog database. By tracing Prolog's execution of your queries, you will be able to see backtracking in action.

Description:

The first part of this exercise requires that you look through a set of Prolog rules and find the syntax errors in them. You will be provided with a file of facts and rules that have some errors. In lab, you will look through the file and identify as many of the errors as possible. You will also look at a set of queries relative to this file and decide on what you think the answer should be. You will later test your expectations on the computer. It is important that you think about the answers first, because a difference in the computer results might be due to a subtle error that you missed.

The second part requires you to write and test some simple rules.

Steps:

1) Look though the first print out of facts and rules attached and identify/correct as many errors as possible. Some errors in Prolog can be rather subtle because of the structure of the language.
2) Determine what you think the answers are for the set of queries below. You should do this with the expectation that all errors have been removed.
3) Copy the file `~mcconnel/CSC351/Prolog/Project1/sample.pl` into your own directory and use a text editor to make the corrections you identified in step 1.
4) Ask the following set of queries and compare the results with your expectations from step 2. If there are differences determine whether your expectations or the Prolog is wrong.

   Queries:

   Is Tom happy?

   Who is happy?

   Who is playing what game/sport?

   Who has supplies?

   Who has pop?

   Who has what?

5) Look through the second print out of facts and rules attached. You will find facts with the following names and structure: `father(Father, Child)`, `male(Person)`, `mother(Mother, Child)`, `female(Person)`, `citizen_of(Person, Country)`.
6) Write Prolog statements to represent the following English sentences (you will have to complete some of these definitions yourself):

   A person is a citizen of a country if the person's parent is a citizen of that country.

   One person is the sibling of another if they have the same parents.

   One person is the sister of another if …

   One person is the brother of another if …

   One person is the aunt of another if …

   One person is the uncle of another if …
7) Copy the file `~mcconnel/CSC351/Prolog/Project1/people.pl` into your own directory and use a text editor to enter the rules you developed in step 6.
8) Query the extended database to get Prolog to determine who are sisters, brothers, aunts, and uncles. Also, find out from the database the names of persons who are citizens of the US and Canada.

Deliverables:

When you complete the program, you will prepare two things: a project report and a zip file containing all of your code.

The project report must use the format given in the sample file on D2L. This report can be prepared as a text file, MS Word document, or PDF. The project report should not be included in the zip file.

The zip file should include all prolog files. You can create the zip file in one of two ways – using zip on brahe or on your own computer.

The two items must be uploaded to the Project 1 drop box on D2L. No other form of submission will be accepted.

```
:- discontiguous(has/2).
:- discontiguous(playing/2).

happy(tom):- watching(tom, bills), has(Tom, supplies).
happy(mary) :- can_pay(mary, bills).
happy(mary) :- \+ (owes_for(mary, bills)).
happy(dave) :- weather(sunny), temperature(hot).
happy(anne) :- happy(dave), \+ (happy(tom)), happy(mary).
happy(anne) - \+ (happy(dave)), happy(tom), \+ (happy(mary)).
happy(anne) :- \+ (happy(dave)), happy(tom), happy(mary).
happy(steve) :- competing(steve, jim, tennis).
happy(steve) :- competing(steve, sue, golf).
happy(steve) :- competing(steve, Anyone, baseball).
happy(carol) := watching(carol, sabres), has(carol, supplies).
has(X, supplies):- has(X, pop), has(X, pretzels).
has(X, supplies):- has(X, pop), has(X, chips).
watching(tom, bills):- is_on(toms_tv), playing(bills, football).
watching(carol, sabres):- is_on(carols_tv), playing(sabres, hockey).
can_pay(X, Y) :- has(X, money), owes_for(X, Y).
competing(X, Y, Sport) :- playing(X, Sport), playing(Y, Sport),
              opponents(x, y).
playing(steve baseball).
playing(bob, baseball).
opponents(steve, bob)
playing(jim, tennis).
playing(sue, golf).
is_on[toms_tv].
is_on(carols-tv).
playing(bills, football).
has(tom, pop).
has(mary, money).
weather(Sunny).
temperature(mild).
has(tom, pretzels).
playing(sabres, hockey).
```

```
has(carol, chips).
has(carol, pop).
owes_for(dave, dinner).
```

```
father(david, peter).
father(peter, harold).
father(peter, helen).
father(peter, sherry).
father(harold, christopher).
mother(maude, peter).
mother(martha, harold).
mother(martha, helen).
mother(martha, sherry).
mother(helen, arnold).
mother(sherry, sonya).
male(david).
male(peter).
male(harold).
male(christopher).
male(arnold).
female(maude).
female(martha).
female(helen).
female(sherry).
female(sonya).
citizen_of(peter, unitedstates).
Citizen_of(maude, canada).
```