COSC 455

Programming Languages: Design and Implementation

Fall 2016

**Lab Exercise #6**

Think left and think right and think low and think high.

Oh, the things you can think up if only you try!

-- Dr. Seuss

**Name: Garrett Bates / 15**

**Goals:** The intention of this lab exercise is to introduce you to the basic concepts in logic programming languages and Prolog. In doing so, you should develop an initial understanding of Prolog and start thinking about developing programs in a different manner. While general discussion is allowed, submitted work must reflect the work, understanding and solutions of the individual student.

**Language/Compiler:** For this lab exercise, we will use the SWI-Prolog environment for our Prolog programming. The SWI-Prolog environment is open source and freely available to download and install on various platforms at <http://www.swi-prolog.org/download/stable>. Alternatively, you may use the online Prolog engine at <http://swish.swi-prolog.org/>, but only use this as a last result.

**Deadline:** Submitted via Blackboard by 11:59pm on [Wednesday](https://www.youtube.com/watch?v=WRGOMyJeQrw), November 30, 2016

**Submission:** The submission of this lab exercise will be a single source file (e.g., \*.doc) via Blackboard and have a naming convention of *FirstNameLastNameLab6*.*doc*. For example, if your name is Homer Simpson, you would submit a single file via Blackboard named *HomerSimponLab6.doc*. ***Lab submissions not following this convention may not be graded.***

**Question 1.** (5 points) There are three forms in Prolog that we are interested in:

* Facts – normally atomic structures with only constant/literal terms
* Rules – implications that can contain variables or constants
* Goals/Questions – atomic or conjunctive structures with or without variables

Open the murder.pl Prolog source file. This is simple demonstration of the power of logic programming. For now, don’t be too concerned with how this program works, just download and run the program. This program is based on a classic Sherlock Holmes story in which he uses a computer, "The Engine", to investigate the murder at the Metropolitan Club. Read through the source and see if you can figure out who the murderer is by hand.

**From the murder.pl file, provide an example of source line and *your own* English explanation for:**

* Prolog fact:

room(mr\_holman, 12).

room(sir\_raymond,10).

room(mr\_woodley,16).

English translation: Gives concrete facts of where each person was. Mr. Holman was in room 12, Sir. Raymond in room 10, Mr. Woodley in room 16.

* Prolog rule:

hair(X,brown):-attire(X,pincenez).

English Translation: The person who wore the pince-nez had brown hair.

* Prolog goal/question:

14 ?- attire(mr\_holman, ring).

true.

15 ?- murderer(Who).

Who = sir\_raymond

English translation: For attire is Mr. Holman wearing a ring? Yes. Who is the murderer? Sir. Raymond.

Sherlock Holmes has trusted you to add a clue into “The Engine” to help solve the murder. Add a rule into the Prolog source code where I have indicated with a comment for the following clue: *The person who was in Room 12 had grey hair*. **Include your addition to the code below.**

**% Add your fact here. Means the person in room 12 had grey hair.**

**hair(X,grey) :-room(X,12).**

Now that we have this out of the way, it is time to investigate the murder. Issue the following commands at the SWI-Prolog command line. (Don't forget the periods, and the case of everything is lower case.)

listing(murderer).

listing(room).

listing(attire).

listing(hair).

**Briefly, describe what these commands do.**

**Answer**: These commands provide the data for each stored type. Using listing and the name of the type will show the attributed rules and facts for that type.

Now, it’s time to solve the murder! Try inputting the following query at the SWI-Prolog command line (don't forget the periods, and the case of X):

murderer(X).

X = sir\_raymond

So, Prolog *magically* solves the murder mystery. Unlike ever programming language you have likely used before (i.e., procedural languages), Prolog is a ***declarative language***. Declarative programming languages express the logic of a computation without describing its control flow. That is, in Prolog you don’t need to tell the language *how* to calculation a solution, but, rather, the structure of the solution. Prolog does so through its built-in, backward-chaining inference engine,

We can ask Prolog to show us his/her thinking and deduction process to solve the murder by issuing the following at the command line:

trace, murderer(X).

Keep hitting return as each step is taken. (Note that Prolog isn’t trying to insult you at each step by calling you a “creep”. Rather, creep indicates that the system is taking a small step forward or backward towards satisfying the goal/question). At each step, you will see that Prolog makes one of four decisions: Call, Fail, Redo or Exit. Follow this process by hand while consulting the source code. **When you think you understand, write a few sentences describing what Prolog is doing.**

**Answer:** Prologue calls for who is the murderer and does a depth first search for the solution. In doing this it calls murderer with its unknown. The known is called and that is for brown hair. Brown hair calls for attire who is wearing the pince-nez. Attire attempts its first guess. If Sir Raymond wore tattered cuffs, then Mr. Woodley wore the pince-nez spectacles. The person in tattered cuffs was in room 16. Checks if Sir Raymond was in room 16. This fails it is not fact that Sir Raymond was in 16. The attire of Sir Raymond wearing tattered cuffs fails. A redo is called for the other situational statement. If Mr. Woodley wore the tattered cuffs, then Sir Raymond wore the pince-nez spectacles. Checks if Mr. Woodley was in room 16 which is true. Exits finding Mr. Woodley in room 16. The attire of Mr. Woodley is tattered cuffs. The attire of Sir Raymond is pince-nez. Sir Raymond has brown hair. This makes the murderer Sir Raymond.

**Question 2.** (5 points) One of the most useful applications for Prolog is its use as a database or for rapid database prototyping. Download and open the elementsDB.pl Prolog source file. This is a prototyped database of chemical elements from the periodic table. The database describes the chemical and physical properties of most of the known elements. Each element has a record structure named *element*, that lists the following information:

* Its name – an atom
* Its chemical symbol – an atomic string with a capital letter and an optional lowercase letter
* Its atomic number
* Whether it is radioactive or not radioactive
* What type of element it is: metal, non-metal, alkali, rare earth, etc.
* What its normal state is, assuming a temperature of 0C and normal pressure
* Its melting point (MP) in Celsius
* Its boiling point (BP) in Celsius

Additionally, there are some abbreviations for conveniently querying an element’s name, it’s name and symbol or to relate it’s name, symbol and number as follows:

element(Name).

element(Name, Symbol).

element(Name, Symbol, Number).

Further, the database also defines some properties of the atomic numbers: Is the element metallic or non-metallic, and what group and period in the periodic table is assigned to the element. There also three utility commands that print information about an element: period(Atomic\_number), group(Atomic\_Number), and print\_element(Name\_of\_an\_element).

Take some time to look through the Prolog source code. When you have an initial understanding, issue a query to retrieve the information of an element of your choice using the print\_element query described above. **Copy and paste your query and the results below.**

**Results:**

15 ?- print\_element(nitrogen).

symbol=N

atomic\_number=7

not\_radioactive

nonmetal

normally=gas

melts\_at= -210

boils\_at= -196

period=2

group=VA

true .

To see how this query works, at the command line, issue the following command:

listing(print\_element).

When you think you understand how this query works, **write a few sentences explaining the query including the meaning of write() and nl.**

**Answer:** A query accesses the facts stored for each element. Write prints out text that is lowercase and the element variable. nl orders to go to the next line. For example, print\_element (nitrogen) will assign nitrogen to A and then print out nitrogen. The facts about nitrogen will be accessed. Then there will be write statements which print out each fact then ordering a next line in between each of the facts.

Now try the following simple searches/queries of our knowledge base:

element(boron, Symbol, Number).

element(Name, 'F', Number).

element(Name, Symbol, 14).

element(barium, 'Ba', 56).

element(hydrogen, 'H', 24).

element([hydrogen](https://www.youtube.com/watch?v=Wi9b0HlD3zE), 'H', Nbr, Rad, Class, Normally, Melt, Boil).

These are all fairly simple queries that you should be somewhat comfortable with by now. The most important thing to recall is the difference between constants and variables. Now, let us issue and compare the following queries:

element(Name, \_, \_, radioactive, \_,\_,\_, \_).

element(Name, Sym, Num, radioactive, Class, Normally, Melt, Boil).

The last of these queries have several answer since we are essentially asking Prolog to *give all the elements that are radioactive*. To get each radioactive element in turn, hit ‘;’.

When you understand the difference between these two relatively similar queries, **write a few sentences to explain the differences of these queries and describe what ‘­\_’ is used for.**

**Answer: ‘**\_’ is used for ignoring a variable in a query. Without these there would not be a way to search through all the elements containing a selected few traits.

For our application, we may think that the ability to provide a listing of the names of all radioactive elements without needing to hit ‘;’ repeatedly is important. To accommodate this, **write a Prolog rule, called print\_radioactive\_elements, to do this and add it to the elementsDB source file.** Test your rule to ensure it functions as described. **Include your Prolog rule below.** Hint: your rule will look similar to print\_element and print\_periodic table. If you get a compiler warning regarding a “singleton” predicate, you should add

:- style\_check(-singleton).

to the top of the source file. This is simply a compiler directive to ignore any singleton predicates.

Rule for print radioactive elements:

print\_radioactive\_elements:-element(Name, Symbol, \_, radioactive, \_, \_, \_, \_),

write(name=Name), nl,

write(symbol=Symbol), nl,nl,

fail.

Your print\_radioactive\_elements rule should include ‘fail’ in it. **Describe below what ‘fail’ does in Prolog.** To best answer this, you may need to experiment with your rule by removing ‘fail’, compiling your program and running it to see what happens. You may also want to use the trace mode from the command line as we did in Task 1.

**Answer: ‘**fail’ tells the rule to run until the rule no longer is true for anymore facts. In the example of print\_radioactive\_elements the rule repeated looking for a next radioactive element until there weren’t any causing a false and exiting.

**Question 3.** (5 points) I have provided a large database, movies.pl, with information about some of my favorite movies. Using this database file, develop (and provide) the queries to answer the following questions:

* In what year was The Godfather released?
* Print all the movies that were released after [2001](https://www.youtube.com/watch?v=x6O2-WPcnjw).
* Find an actor or actress who has also directed a movie.
* Provide the name of a director of a movie in which [Gillian Welch](https://www.youtube.com/watch?v=jdYG-Nh_AxU), one of my favorite singers, was an actress.
* Print those actors or actresses who have played “himself” or “herself”.

**Include the answers and queries for these questions below.**

* In what year was The Godfather released?

Query:

movie(the\_godfather, X).

Answer:

X = 1972.

* Print all the movies that were released after [2001](https://www.youtube.com/watch?v=x6O2-WPcnjw).

Query:

movie(X, Y), Y<2001, write(X), nl, fail.

Answer:

american\_beauty

the\_big\_lebowski

blade\_runner

the\_cotton\_club

down\_from\_the\_mountain

fargo

the\_firm

ghost\_busters

the\_godfather

the\_godfather\_part\_ii

the\_godfather\_part\_iii

groundhog\_day

hearts\_of\_darkness\_a\_filmmaker\_s\_apocalypse

the\_hudsucker\_proxy

inside\_monkey\_zetterland

lick\_the\_star

miller\_s\_crossing

mission\_impossible

o\_brother\_where\_art\_thou

the\_outsiders

peggy\_sue\_got\_married

raising\_arizona

rumble\_fish

spies\_like\_us

star\_wars\_episode\_i\_\_the\_phantom\_menace

torrance\_rises

the\_usual\_suspects

the\_virgin\_suicides

fall

ghost\_world

if\_lucy\_fell

home\_alone\_3

the\_horse\_whisperer

just\_cause

manny\_\_lo

my\_brother\_the\_pig

north

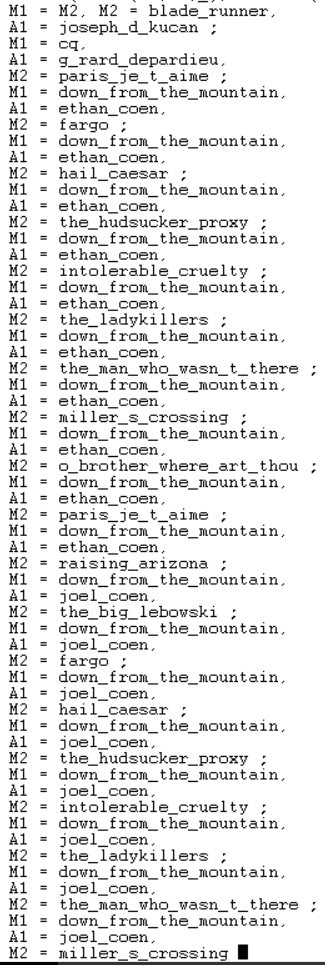
false.

* Find an actor or actress who has also directed a movie.

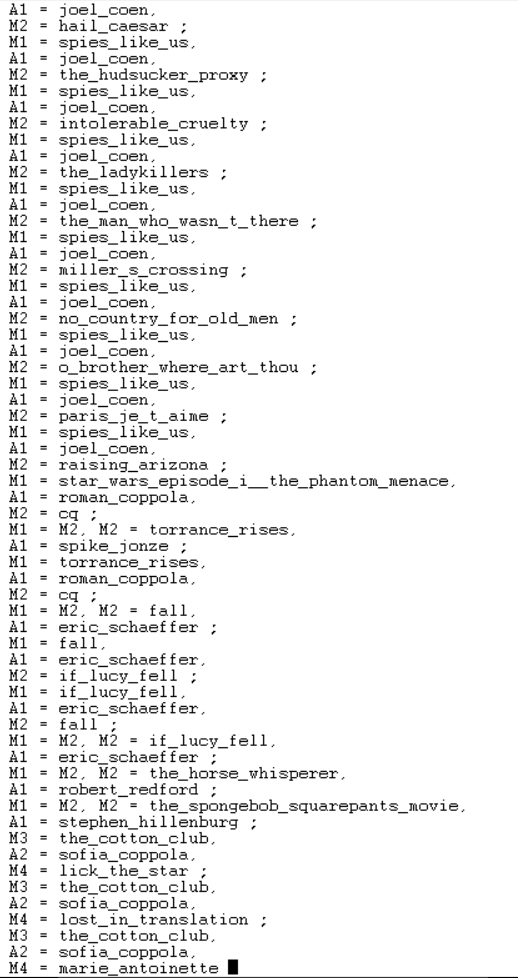
Query:

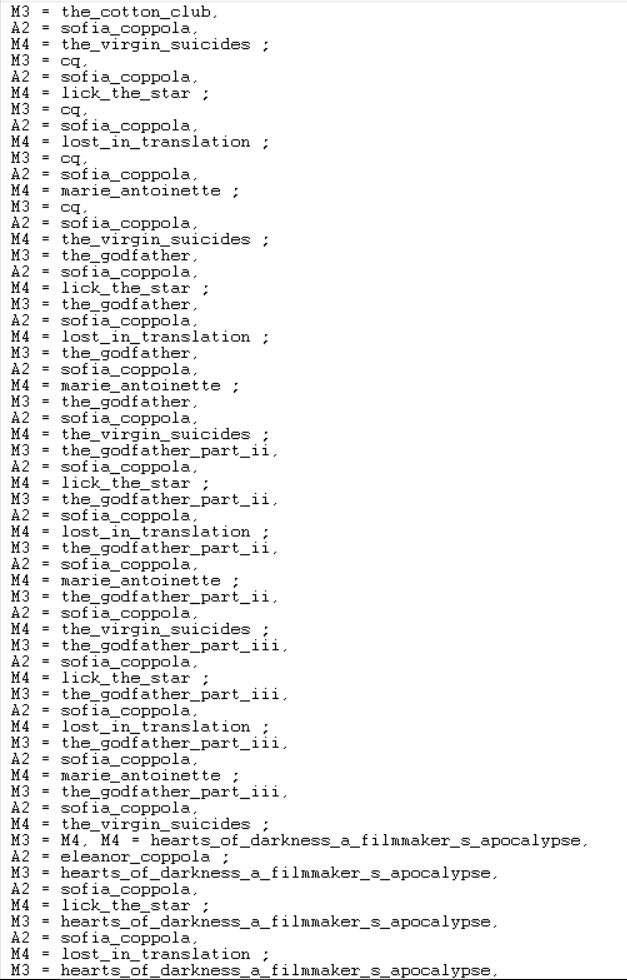
(actor(M1,A1,\_),director(M2,A1)); (actress(M3,A2,\_), director(M4,A2)).

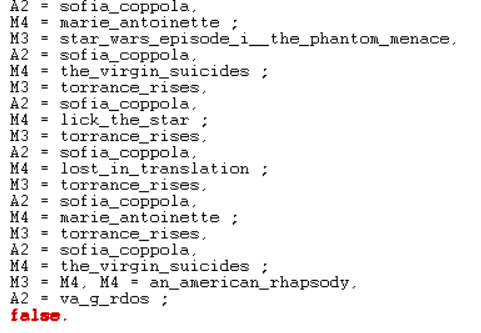
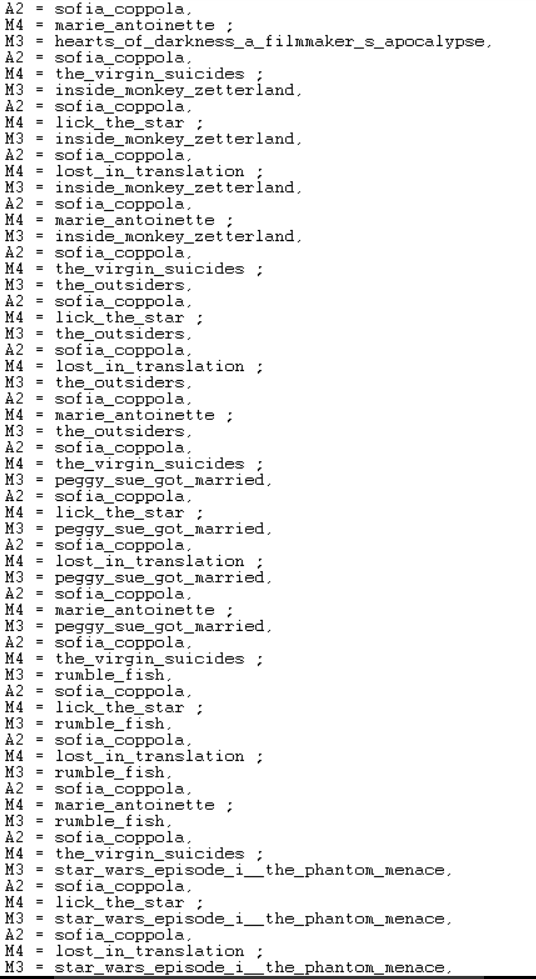
Answer:











* Provide the name of a director of a movie in which [Gillian Welch](https://www.youtube.com/watch?v=jdYG-Nh_AxU), one of my favorite singers, was an actress.

Query:

actress(M, gillian\_welch, \_), director(M, D).

Answer:

M = down\_from\_the\_mountain,

D = nick\_doob ;

M = down\_from\_the\_mountain,

D = chris\_hegedus ;

M = down\_from\_the\_mountain,

D = d\_a\_pennebaker ;

M = o\_brother\_where\_art\_thou,

D = ethan\_coen ;

M = o\_brother\_where\_art\_thou,

D = joel\_coen.

* Print those actors or actresses who have played “himself” or “herself”.

Query:

actress(\_,N, N); actor(\_,N,N).

Answer:

N = janeane\_garofalo ;

N = madonna ;

N = roman\_coppola ;

N = eminem ;

N = regis\_philbin ;

N = chris\_rock ;

N = michael\_rooney ;

N = fatboy\_slim ;

N = will\_smith ;

false.

**Note:** There are 2-5 [Easter eggs](https://en.wikipedia.org/wiki/Easter_egg_(media)) (this isn’t one of them) in this lab. If you find one, let me know!