COSC 455

Programming Languages: Design and Implementation

Fall 2016

**Project #3 – Logic Programming Using Prolog**

**Directions:** This project has 7 individual problems worth 45 possible points. However, the project will be scored out of a maximum of 30 points. You may choose to do as many, or as few, problems as you’d like. Please read through the entire project description before starting it.

**Goals:** The intention of this project is to gain experience in programming with a logic programming language and compare the implementation to that of an imperative language. In doing so, you should develop a better understanding of logic programming, nonprocedural languages, Prolog and the concepts behind this class of programming languages. This project will give you development experience in the following areas of programming languages that were covered in class:

* Predicate calculus
* Logic programming
* Terms, facts, rules, goals and inferences
* List data structures

Further, the implementation of these concepts will give you further experience in essential programming concepts including:

* Problem solving
* Recursion

**Deadline:** Monday, December 12, 2016, 11:59pm to Blackboard

**Language Requirements:** Prolog and the SWI-Prolog compiler (<http://www.swi-prolog.org/>) should be used for this project. For those having difficulty installing SWI-Prolog on Macs, you may alternatively use the online Prolog interpreter at <http://swish.swi-prolog.org/> for some problems (note that this will not work for all programs).

**Grading:** Each developed program will be graded as follows: 40% allocated for code development; 40% for correct solution; and, 20% for quality of your code (i.e., structure, comments, etc.).

**Submission:** You must submit: 1. all Prolog source code; 2. screenshots of example output; and 3. a readme file describing how to run each Prolog file. You should name the files *prog1.pl, prog2.pl*, etc. and separate the *programs into their own respective directories* (e.g., Program1, Program2, etc.). Submit all files as a zip file to Blackboard. ***Failure to submit the project follow these guidelines* *may result in your project not being graded.***

**Program 1 – Language Translation (5 points)**

*Hanyu pinyin* is the most commonly used Romanization system for the Standard Mandarin Chinese language. *Hanyu* means “Chinese language” and *pinyin* means “spell sound” or “phonetic”. Consider the following table:



Use this table as a set of facts that can relate a Chinese pinyin word with an English word. For example, one might envision a fact written in Prolog as:

translation(ling, zero).

As one who has studied Mandarin Chinese, I would like a Prolog program that would translate a list of English numbers and produce the corresponding list of Chinese numbers, written in pinyin. To do this, I should be able to query at the Prolog command line with the predicate translate as follows:

?- translate([qi, ling, si, si, wu, san, liu], English).

which should produce the output:

?- English = [seven, zero, four, four, five, three, six].

Your translation program, however, should be able to go in both directions. For example, the query

?- translate(Chinese, [four, five, five]).

should output:

?- Chinese = [si, wu, wu].

**Program 2 – Prolog for Prototyping Digital Circuits (5 points)**

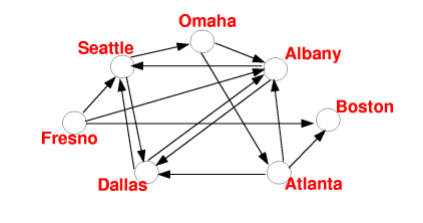
Basic digital logic gates, including NOT, OR, XOR and AND, can be combined in many ways to design simple circuits. In Prolog, develop the facts and rules for the following circuits:

* Half-Subtractor (<http://www.flintgroups.com/2012/10/half-subtractor-and-full-subtractor.html>)
* 1-to-2 Line Decoder (<http://www.play-hookey.com/digital/combinational/decoder_demux_two.html>)
* 2-to-4 Line Decoder (<http://www.play-hookey.com/digital/combinational/decoder_demux_four.html>)
* 2-Input Multiplexor (<http://www.play-hookey.com/digital/combinational/multiplexer_two_input.html>)
* 4-Input Multiplexor (<http://www.play-hookey.com/digital/combinational/multiplexer_four_input.html>)

When your program is compiled, it should display instructions on how to use it. I should be able to call each of the above circuits to produce the truth tables for input/output.

**Program 3 – Prolog as a Constraint Checker (10 points)**

Consider the following graph of flight paths between 7 US cities:



Encode this information and:

1. (5 points) Write a Prolog program that checks if there is a route between two given cites. For example, a sample test run could be:

?- flight(fresno, omaha).

true.

?- flight(omaha, fresno).

false.

1. (5 points) Write a program using an imperative language of your choice that checks if there is a route between two given cites. The cities will be provided as runtime arguments for an executable file (e.g., .exe, .jar). For example, if used Java, a sample test run could be:

> java -jar prog3b.jar flight(fresno, omaha)

true.

> java -jar prog3b.jar flight(omaha, fresno)

false.

**Program 4 – Prolog as an Interactive Database (10 points)**

To emphasize the use of Prolog as a database, we will design an intelligent front-end to a database. Develop a set of Prolog facts of the following form, describing courses offered at Towson University:

course(Name,

location(Building, Room),

time(Days, StartTime),

instructor(LastName).

For example, the courses I teach this semester could be described using the facts:

course(cosc455,

location(yr7800, 204),

time(t, 1100),

instructor(dehlinger)).

course(cosc455,

location(yr7800, 402),

time(r, 1100),

instructor(dehlinger)).

course(cosc612,

location(yr7800, 125),

time(w, 1900),

instructor(dehlinger)).

course(cosc465,

location(yr7800, 202),

time(t, 1230),

instructor(davani)).

course(cosc465,

location(yr7800, 202),

time(r, 1230),

instructor(davani)).

course(cosc439,

location(yr7800, 401),

time(m, 1900),

instructor(azadegan)).

course(cosc578,

location(yr7800, 202),

time(m, 1900),

instructor(zimand)).

course(cosc175,

location(yr7800, 205),

time(t, 1100),

instructor(taylor)).

course(cosc175,

location(yr7800, 205),

time(r, 1100),

instructor(taylor)).

Note that the times are listed on a 24 hour time scale (i.e., 1900 is 7:00pm). Additional facts could be developed from your own schedule of classes. Note that when I test your program, I will use my own set of facts that will be much larger than the example given above.

You should initially load a set of facts from a file called fall16sched.pl (or fall16sched.pro, depending on your version) using a *load\_courses* command. Your program should be able to additionally output all courses (including course name, location, time and instructor) in a nicely formatted output using a *list\_courses* command; save the courses to the fall16sched.pl (or fall16sched.pro, depending on your version) file using the ­*save\_courses* command; and add and delete courses using the commands *add\_course(Name, Location, Time, Instructor)* and *delete\_course(Name, Location, Time, Instructor)*, respectively. Adding and deleting should physically add/delete the course from the fall16sched.pl (or fall16sched.pro, depending on your version) file. These functions will be demonstrated in class (and also described in Blackboard under Projects/Project 3/ Prolog Resources / Interactive Database Tutorial and my example source code can be found on Blackboard under Projects/Project 3/ Prolog Resources / Interactive Prolog Database Source Files).

In order to interface with this database, your task is to define by writing Prolog facts and/or rules to define the following relations:

* is\_teaching(LastName, Period) which is true if LastName teaches a class at the specified Period. For example, consider the following:

?- is\_teaching(dehlinger, Period).

Period = time(t, 1100);

Period = time(w, 1900);

Period = time(r, 1100);

* is\_instructor(LastName, Course) which is true if Course is taught by a person whose last name is LastName. For example, consider the following:

?- is\_instructor(WhoIs, cosc455).

WhoIs = dehlinger

?- is\_instructor(dehlinger, Class).

Class = cosc455 ;

Class = cosc612 ;

* is\_busy(LastName, Days, Hour, Location) which is true if LastName is busy at Location on Days at time Hour. For example, consider the following:

?- is\_busy(dehlinger, w, 1900, Where).

Where = location(yr7800, 125)

?- is\_busy(WhoIs, t, 1100, location(yr7800, 205)).

WhoIs = taylor

Note that you may assume that the day and time will be fully instantiated in the queries for is\_busy (i.e., it will contain no variables).

* cannot\_meet(Time, Instructor1, Instructor2) a time that two instructors cannot meet because one or both are teaching a course. For example:

?- cannot\_meet(time(t,1100),dehlinger, taylor).   
true

* schedule\_conflict(Course1, Course2) that relates the times that two given courses are taught. For example:

?- schedule\_conflict(cosc612, cosc455).   
false.

**Program 5 – A Murder Mystery, part II (5 points)**

Four guests (Colonel Mustard, Professor Plum, Miss Scarlett and Ms. Green) attend a dinner party at the home of Mr. Boddy. Suddenly, the lights go out! When the lights come back on, Mr. Boddy lies dead in the middle of the table. Everyone is a suspect.

Upon further examination, the following clues come to light:

* Mr. Boddy was having an aﬀair with Ms. Green
* Professor Plum is married to Ms. Green.
* Mr. Boddy was very rich.
* Colonel Mustard is very greedy.
* Miss Scarlett was also having an aﬀair with Mr. Boddy.

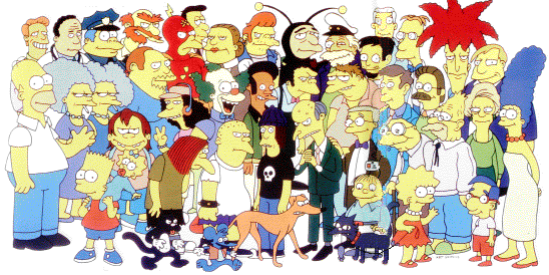
The guests remaining conclude that there are two possible motives for the murder: hatred and greed. Someone hates someone else if that other person is having an aﬀair with his or her spouse. Someone is willing to commit murder if they are greedy and not rich, and the victim is rich.

1. Write a Prolog program that encodes the above facts and rules. When encoding these rules and facts, be careful about how you encode symmetric relationships (e.g., marriage) so that you don’t get Prolog into an infinite loop. Hint: The “not” or “not provable” symbol in Prolog is “\+”. For example the following code defines a rule, named female, that is defined as anyone who is not a male.

female(Person) :- \+(male(Person)).

1. Given these facts and rules, use Prolog to determine who murdered Mr. Boddy. Who are the possible suspects? Provide a screenshot of this answer.
2. Add a single fact to your database that will result in there being a unique suspect. Mark this additional fact in your code with a preceding comment. Hint: comments in Prolog start with the “%” symbol.

**Program 6 – The Simpsons Prolog (5 points)**



My favorite TV show, The Simpson, has a big cast of characters with a diverse set of jobs and salaries. In Prolog, I have set up a database, *prog6.pl*, of characters along with their salaries in the following predicate form:

salary([first\_name, last\_name], salary\_amount).

where the name is specified in a list with two elements. Complete my *prog6.pl* program with a predicate that would compute the total income for a given family where a family is identified by having the same last name. For example,

?- family\_income(simpson, Income).

Income = 50500.

In my solution, I used Prolog’s *bagof()* predicate (not covered in class). Briefly, the bagof() predicate collects elements into a list. For example:

child(bart, homer).

child(lisa, homer).

child(maggie, homer).

all\_children(X, C) :- bagof(M, child(M, X), C).

?- all\_children(homer, Children).

Children = [bart, lisa, Maggie]

**Program 7 – Drunk Homer (5 points)**

Homer Simpson, after a stop at Moe’s, went to the Springfield Mall to buy Marge, Lisa, Bart and Maggie a gift in anticipation that they will be upset with him when he gets home. While at the mall, Homer buys 4 gifts: a green dress for Marge, a saxophone book for Lisa, a slingshot for Bart and a new pacifier for Maggie. He recalls buying the gifts at the following stores: The Leftorium, Sprawl-Mart, Try-N-Save, and King Toots.

Somewhere along the way, Homer lost his car keys and had to walk home carrying the gifts. Wanting to retrace his steps and find his lost car keys, the family asks Homer where he bought the gifts and the order in which he bought the gifts. Being partly inebriated however, Homer can’t remember which stores he bought the gifts at and in which order he visited the stores. After some sobering up, Homer remembers the following:

* He bought the saxophone book at King Toots
* The store he visited immediately after buying the slingshot was not Sprawl-Mart
* The Leftorium was his second stop
* Two stops after leaving Try-N-Save, he bought the pacifier

Develop a Prolog program that will help Homer solve the mystery of the stores where he bought each gift and the order in which he bought them.