"Prolog Parentage"
Programming Lab 5

Programming Language Fundamentals (CS 381)

Prolog

For this lab, you will use Prolog.
Download Prolog from here: http://www.swi-prolog.org/download/stable.

Dataset

Begin by downloading this file `royal.pl` from Canvas. This file is a set of Prolog facts that represents the genealogy of the royal House of Windsor.

Getting Started

Next, create a new file `Lab5.pl`.[1] This file should be in the same directory as `royal.pl`. In this file, you will include all your Prolog rules you will be asked to write in the lab.

Your program should begin with your name as a comment.

```
% First_name Last_name
% CSCI 381 Lab
```

Warmup

Check that everything is working. First start `prolog` at the command line.

Next, load the file `royal.pl` by typing:

```
:- consult('royal.pl').
```

[1] Note that `.pl` is an extension also used by Perl and may confuse syntax highlighting programs.

PROTIP: The period `.` indicates the end of your statement and is required.

Now, run the query:

```
?- parent(X, 'Queen Elizabeth II').
```

PROTIP: The symbol `?-` is the Prolog query prompt. You should not retype this.

This query asks who is the parent of Queen Elizabeth II. Prolog first returns the result `X = 'King George VI'`. Type a semicolon (;). This will prompt Prolog for the next result. Prolog then returns `X ='Lady Elizabeth Bowes-Lyon'`. The period following the second result indicates there are no additional results.

The entire query and result interaction should appear as:

```
?- parent(X, 'Queen Elizabeth II').
X = 'King George VI' ;
X = 'Lady Elizabeth Bowes-Lyon'.
```

Replicate this exchange on your machine.

Parents

In this lab you will create a number of Prolog rules that define a number of common familial relationships.

In Prolog parlance, we define a rule as its name and its arity. For instance the rule `mother/2` indicates a rule named `mother` with two arguments.

```
mother(M,C):- parent(M,C), female(M).
```

Add this rule to your Prolog file. Reload your file. Test your new rule with the query:

```
?- mother(X, 'Queen Elizabeth II').
```

which should return the single result:

```
X = 'Lady Elizabeth Bowes-Lyon'.
```

Now write the rule `father/2` and add it to your file.

More Rules

Now write a set of additional rules. In your rules, you are encouraged to make use of other rules you have written. This may effect the order you choose to implement these rules.

PROTIP: Make sure you get the order of the arguments correct as switching then will change the meaning of the function.

- `spouse/2`
- `child/2`
- `son/2`
- `daughter/2`
- `sibling/2`
- `brother/2`
- `sister/2`
- `uncle/2` — two rules: one by blood, one by marriage.
- `aunt/2` — two rules: one by blood, one by marriage.
- `grandparent/2`
- `grandfather/2`
- `grandmother/2`
- `grandchild/2`

For the next two rules, make use of rules you have already written. It may take more than one rule to define these functions.

Write the rules:

- `ancestor/2` — `ancestor(X, Y)` means `X` is the ancestor of `Y`.
- `descendant/2` — `descendant(X, Y)` means `X` is the descendant of `Y`.

PROTIP: You may find tracing useful. To debug, you can use the keyword `trace` to enable tracing and `notrace` to disable.

Numbers in Prolog

Now we will write some Prolog rules that require numeric comparisons. Write the rules:

- `older/2`
- `younger/2`

The rule `older(X, Y)` indicates person X is older than person Y. The rule `younger/2` should be written in a similar manner.

Lastly, write the rule `regentWhenBorn/2`. This rule, `regentWhenBorn(X, Y)`, should ask who was King or Queen (X) when person Y was born.

Testing

Output Format

Add this line to your code to alter the output formatting to work with the automatic tester.

```
portray(Term) :- atom(Term), format("~s", Term).
```

Single Test

This will run your code using a single test at the command line and print to `stdout`.

```
$ swipl -q lab5.pl < ./tests/t01.in
```

PROTIP: The prolog argument `--quiet` (or -q for short) runs prolog in quiet mode.

Testing Script

This will run a single test using the provided Python3 script. It runs the specific test given as the `-t` argument. In this example, it will run Test `t01`.

```
$ python lab5-tester.py lab5.pl -t t01
```

Or you can run all tests at once.

```
$ python lab5-tester.py lab5.pl
```

Troubleshooting

This lab requires an independent study of the Prolog language. You are encouraged to use any web tutorials and resources to learn Prolog. Given that only short code snippets are required, I will not debug your code for you or give hints.

Allow yourself plenty of time, and use patience, perseverance, and the Internet to debug your code. I will gladly answer clarifying questions about the goals and instructions of the Lab assignment.

Submission

NOTE: Given the end of term, a maximum of 1 late day may be applied for Lab5.

Each student will complete and submit this assignment individually. Do not consult with others. However, you are encouraged to use the Internet to learn Prolog.

Submit your program (`Lab5.pl`) on Canvas. This project is worth 100 points. Comment your program heavily.

PROTIP: If you do not finish in time, turn in whatever work you have. If you turn nothing, you get a zero. If you turn in something, you receive partial credit.

Extra Credit (10 pts)

Write rule(s) that define the first cousin relationship, `cousin/2`. You must determine the formal (logic) definition of first-cousin.

PROTIP: You do not need to support second or third cousins, nor cousins removed.