Language Assignment #3: Prolog

Issued: Thursday, March 18
Due: Thursday, April 1

Purpose

This assignment asks you to begin using a logic programming language named Prolog. Prolog was designed by Alain Colmerauer, from Aix-Marseille University, in 1972.

Translator

In our lab, `onyx` is the home-directory file server for its nodes (e.g., `onyxnode01`). There is also a shared directory for "apps" mounted at `/usr/local/apps`. Nodes share a translator for Prolog, named `gprolog`, which is installed below `/usr/local/apps`, which is a non-standard location.

Due to shared-library constraints, `onyx` cannot execute `gprolog`. It can only be executed by a node.

Due to network constraints, `onyx` can be reached from the public Internet, but a node can only be reached from `onyx`. So, SSH and login to `onyx`, then SSH and login to a node.

An easy way to use `gprolog`, from a node, is to permanently add a line to the end of your `.bashrc` file. To do so, login to a random node, from `onyx`, by executing the script:

```
pub/bin/sshnode
```

Then, execute the script:

```
pub/bin/bashrc
```

Don't change your `$PATH`; just execute the script. Then, logout from the node and login to a node.

Documentation

Prolog lecture slides are at:

```
pub/slides/slides-prolog.pdf
```

Prolog is demonstrated by:

```
pub/sum/prolog
```

Links to manuals are at:

```
http://cs.boisestate.edu/~buff/pl.html
```

Prolog is described in Section 12.2 of our textbook.

Assignment

Write and fully demonstrate a program that selects a set of acceptable meeting times for a set of people. Each person provides a set of free time slots. For example:

```
pub/la3/data.pl
```

shows that Bob has three time slots that are free, one of which is 7:00AM–8:30AM. Bob's not very busy!

Hints and Advice

Start with a simpler problem. A skeletal solution is:

```
pub/la3/meetone.pl
```

When complete, this program will print the names of people who can meet from 8:30AM–8:45AM. This will be Ann, Carla, and Dave. If the "query" was for 5:30AM–6:45AM there would be no solutions.

Then, extend your solution to the complete problem. A skeletal solution is:

```
pub/la3/meet.pl
```

When complete, this program will print a list of compatible meeting times for Ann, Bob, and Carla: 8:00AM–8:30AM, 10:00AM–10:15AM, and 8:00PM–8:30PM.

Given two times slots, there are several ways in which they can overlap, but some are symmetric. Draw pictures. Your predicate will need a rule for each way. Use the `\==` predicate to avoid zero-length meetings. A big hint is for your predicate to "return" a new time slot modeling the overlap.

A general Prolog hint is to use one or more variables to pass data "into" a predicate, and another variable to "return" data.

You'll also need a predicate for comparing two times (e.g., `lte`).

Test your solution thoroughly, by modifying `data.pl` and the list of people who want to meet.