Lab 7

Due: July 14, 2023 at the end of lab

1 Software setup

This section consists of Prolog setup that you should only have to follow once.

Download and install a common compiler for Prolog, **SWI-Prolog**, from `http://www.swi-prolog.org/download/devel`. Note that this is the developer version, not the stable version. Accept the defaults for installation.

You may get some security warnings upon installation or the first time you run the program. Give the program permission to install/run.

If you are running SWI-Prolog on a Mac, you have to install xQuartz from `https://www.xquartz.org/` as well. (If you forget, your IDE will have no menu items.)

1.1 Importing project

The skeleton code at `https://classroom.github.com/a/RgIou1M8` is the start of this week's lab. SWI-Prolog doesn't have a built-in git interface like more robust IDEs. For the amount that we are using Prolog, it makes the most sense to download the source file manually and update it with your solution on the Github website manually.

1. In Github, go to your Lab 7 project
2. Click the **Repository** link in the horizontal navigation bar.
3. Click the **Download zip** button to download all the project files.
4. Extract the .zip file into an appropriate place on your computer.

At this point, you should have Lab 7 available to load into SWI-Prolog.

1.2 Completing project

Refer to section **2** for the programming part of the project.

1.3 Exporting project

Once you have completed your code (or you have updated it since last pushing it to Github), follow these steps to make sure that I can access your most recent code:

1. In Github, go to your Lab 7 project
2. Click the **Repository** link in the horizontal navigation bar.
3. Click on the file that you want to update.
4. Click on the **edit** button.
5. Paste your Prolog code into the text box.
6. Click the **Commit Changes** button at the bottom of the page.

Once you have committed your code, I have access to it and I can download and grade it. I look at the date of the most recent changes in github to determine timeliness of your lab, so make sure go through these last steps before the due date.

2 Lab 7 Specification

2.1 Your very own knowledge base

Implement a knowledge base of your devising and provide example queries on that knowledge base. Your knowledge base can entail any information you would like, with any relationships between the atoms that makes sense.

Write your clauses in **lab7a.pl**, using any text editor you like.

Note: Your computer may recognize **.pl** files as Perl files, as they use the same file extension as Prolog files. Don't let the code highlighting distract you.

Provide at least **8 facts** and **4 rules** based on those facts. Write at least **3 queries** for your knowledge base. The rules must have more than one predicate on the right side. The queries must reference a rule or use a variable.

2.1.1 Running your queries

Load your program by launching SWI-Prolog, going to **File** → **Consult ...** and selecting the file from your computer. You should get feedback that the file loaded successfully, or that there were errors. Test that your queries return the results you expect by running them from the prolog console. After your query returns a result, you can hit '**.**' to terminate the query or '**;**' to keep looking for more true statements. If you change your file in a text editor, reload the contents by going to **File** → **Reload modified files**.

Paste your 3 queries within the comments section at the bottom of the **lab7a.pl** file.

2.2 Database Relations

```
classtime(1000,10).      where(1000,dobbs102).      enroll(mary,1200).
classtime(1200,12).      where(1200,dobbs118).      enroll(john,3400).
classtime(3400,11).      where(3400,wentw216).      enroll(mary,3350).
classtime(3350,12).      where(3350,wentw118).      enroll(john,1000).
classtime(2350,11).      where(2350,wentw216).      enroll(jim,1000).
```

Define the following predicates by writing a rule or rules for them. The examples and results use the facts given above.

2.2.1 schedule/3

The **schedule** predicate should take the arguments of (student, classroom, time). With this predicate, we can form queries such as `?- schedule(mary, P, T).` and get results: `P = dobbs118, T = 12; P = wentw118, T = 12.` This predicate can also query a classroom's usage with something like `?- schedule(S, wentw216, T).` This will show all the students in a classroom and the times that they are there.

2.2.2 usage/2

The **usage** predicate should take the arguments of (classroom, time) and give all the times that a classroom is in use. For example, `?- usage(dobbs102, T).` would result in: `T = 10.` The query `usage(X, 12).` should return all of the classrooms that are in use at 12.

3 Grading

Be professional. Make results easy to understand and grade. Include only those parts to be graded. Leave comments where necessary, especially if it aids in grading.

Your knowledge base is worth 50% of the lab grade. **schedule** and **usage** are each worth 25% of the lab grade.