Programming Language Concepts

4003-450-01

Spring Quarter 20113

Project 3: Prolog

Due

- As shown on myCourses Project 3 submission dropbox. Team project.

Objective

- Gain additional experience with Prolog by doing two unrelated mini-projects.

Task A. Old British Money (Bob's your Uncle)

Before 1971, British currency did not use the current decimal system, where a British pound has 100 British pennies. To use typical British understatement, their old system of coins was fairly convoluted, as described at:

http://h2g2.com/dna/h2g2/A506350

Adapt the coin-changing problem for US currency discussed in class to find the smallest number of notes/coins needed to represent a given amount expressed in pre-1971 British pennies. For this assignment, assume that the pre-1971 British currency conversions were set up as follows.

- 1 pound = 20 shillings
- 1 shilling (aka bob) = 12 pennies (also called pence)

The notes and coins to be used are as follows:

- 1 tenner (note) = 10 shillings
- 1 half-a-crown (coin) = 2½ shillings
- 1 florin (coin) = 2 shillings
- 1 sixpence (coin) = 6 pennies (as in "Sing a song of sixpence, a pocketful of rye, ..."
- 1 joey (coin) = 4 pennies
- 1 tickie (coin) = 3 pennies
- 1 tuppence (coin) = 2 pennies
- 1 copper (coin) = 1 penny
- 1 ha-penny (coin) = ½ penny
- 1 farthing (coin) = ¼ penny

If you were wondering whether half-a-crown should imply there should be a crown. Yes, the good news that there were crowns (5 shillings), which soon became obsolete so the bad news is: you can omit them from this project.☺

In a file named `coins.pl`, write predicate(s) to find the least number of coins for a given amount in pence. See the Submission area for a description of what files need to be submitted.

Task B: Flighty Tales

A flight database contains the following facts about flights and layover times at airports:

```
flight(roc, syr, 25).   flight(roc, jfk, 55).   flight(jfk, bos, 65).
flight(bos, syr, 40).   flight(jfk, syr, 50).   flight(bos, roc, 50).
layover(roc, 25).       layover(jfk, 55).       layover(syr, 30).
layover(bos, 40).
```

The first flight indicates that there is a non-stop 25-minute flight from `roc` to `syr`; a layover time is the time it typically takes to connect from one flight to another at that airport, i.e., the layover time at `BOS` is 40 minutes.

You are to write a rule:

```
          route(X, Y, R, D) :- ...
```

that will find a route R of duration D (if one exists) from airport X to airport Y via zero or more layover (hub) airports. The intent is that the user will invoke route as a query, specifying X and Y as constants and R and D as variables. For example, the following interaction occurs when the user types the following to the Prolog interpreter:

```
?- route(roc, syr, Routing, Duration).

Routing = [roc, syr],
Duration = 25 ;

Routing = [roc, jfk, syr],
Duration = 160 ;

Routing = [roc, jfk, bos, syr],
Duration = 255 ;

false.
```

(Note: the ";"s above represent user requests for additional matches from the Prolog interpreter.)

Flights are unidirectional, that is, a flight from airport `roc` to `jfk` does not imply there is a flight from `jfk` and `roc`; moreover, even if there is a flight, the duration of the onward and return flights need not be the same. For example, flying from New York's `jfk` to Los Angeles's `lax` takes more time than from `lax` to `jfk`.

Flight paths may involve cycles so your program must handle them by ensuring that your route does not loop around the same airport multiple times.

See the Submission area below for a description of the files to be submitted.

Submission

Submit a file:
`project3.zip`
which should contain five files:

`coins.pl`	*the facts and rules for Task A.*
`cQueries.pl`	*the test queries you used to test Task A*
`flights.pl`	*the flight and airport facts you set up for Task B.*
`routes.pl`	*the rules you developed to meet the routing requirements for Task B.*
`fQueries.pl`	*the test queries (or goals) you used to test Task B.*

Note that I will use my own facts and queries to test your rules.

As in earlier projects, the instructor's test script requires you to name your files as stated; if you don't, the script will simply generate a failing grade! So do take care to avoid severe penalties.

Please submit only the named zipped file before the deadline. As before, emailed or hardcopy submissions will not be accepted.