CMPE 260
Principles of Programming Languages
Spring 2015
Prolog Project

Due Date : 04.04.2016

March 19, 2016

1 European Cities and Languages

In this project, you will be traveling between the famous European cities and speaking the local languages.

2 Knowledge Base

You will be given a knowledge base with the following predicates.

2.1 `road( SourceCity, TargetCity, Distance ).`

The `road` predicate defines a direct connection from 'SourceCity' to 'TargetCity' that is 'Distance' kms long.

This predicate is bi-directional and it implies a reverse connection from 'TargetCity' to 'SourceCity' with the same distance.

2.2 `language( City, Language ).`

The `language` predicate represents that the language 'Language' is spoken in the 'City'.

'Language' is not a list and one stand-alone predicate would appear in the knowledge base for each valid 'City' 'Language' tuple.

2.3 Example Knowledge Base

Here is an example knowledge base that you can use during the development of your predicates.

```
road( london, paris, 459 ).
road( paris, berlin, 1054 ).
road( paris, barcelona, 1036 ).
road( barcelona, milano, 981 ).
road( milano, budapest, 960 ).
road( berlin, budapest, 874 ).
road( budapest, istanbul, 1319 ).

language( london, english ).
language( paris, french ).
language( paris, arabic ).
language( berlin, german ).
language( berlin, turkish ).
language( barcelona, spanish ).
language( barcelona, catalan ).
language( barcelona, italian ).
language( milano, italian ).
language( budapest, hungarian ).
language( istanbul, turkish ).
language( istanbul, arabic ).
```

3 Queries

For the project, you have to define the following predicates.

3.1 `communicate_with( +City1, +City2, +Language ).`

Define `communicate_with/3` predicate such that it is true for two different cities 'City1' and 'City2' that share the common 'Language'.

3.2 `communicate( +City1, +City2 ).`

Define `communicate/2` predicate such that it is true for two different cities 'City1' and 'City2' that share a common language.

3.3 `cities_of_language( +Language, -CityList ).`

`cities_of_language/2` predicate finds all the cities in which 'Language' is spoken and returns a list 'CityList'.

3.4 `languages_of_city( +City, -LanguageList ).`

`languages_of_city/2` predicate finds all the languages that are spoken in 'City' and puts them into 'LanguageList'.

3.5 `is_connected( +City1, +City2 ).`

Two cities 'City1' and 'City2' are connected if there is road between them. Define `is_connected/2` predicate such that it is true for all connected cities.

3.6 `distance( +City1, +City2, -Distance ).`

The 'Distance' between the connected cities 'City1' and 'City2'.

3.7 `connected_cities( +City, -CityList ).`

`connected_cities/2` predicate finds all the cities that are connected to 'City' and puts them into 'CityList'. Just consider direct connections.

3.8 `minimum_distance( +City1, +City2, -Distance ).`

`minimum_distance/3` predicate finds the minimum 'Distance' between 'City1' and 'City2'. To implement this predicate, just consider the direct distance between City1 and City2, and the distance of City1-CityX-City2, i.e. going to City2 from City1 over a different City. You may need to implement auxiliary predicates for this question.

3.9 Test Cases

You can control the accuracy of your predicates by testing them over the given example knowledge base.

```
?- communicate_with(berlin,istanbul,Language).
Language = turkish ;
false.

?- communicate_with(City1,City2,italian).
City1 = barcelona,
City2 = milano ;
City1 = milano,
City2 = barcelona ;
false.

?- communicate(City1,City2).
City1 = paris,
City2 = istanbul ;
City1 = berlin,
City2 = istanbul ;
City1 = barcelona,
City2 = milano ;
City1 = milano,
City2 = barcelona ;
City1 = istanbul,
City2 = berlin ;
City1 = istanbul,
City2 = paris ;
false.

?- cities_of_language(arabic,CityList).
CityList = [paris, istanbul].

?- languages_of_city(barcelona,LanguageList).
LanguageList = [spanish, catalan, italian].

?- is_connected(london,istanbul).
```

```
true ;
true ;
false.

?- is_connected(milano,berlin).
false.

?- is_connected(paris,City2).
City2 = berlin ;
City2 = barcelona ;
City2 = budapest ;
City2 = istanbul ;
City2 = milano ;
City2 = budapest ;
City2 = istanbul ;
false.

?- distance(london,istanbul,Distance).
Distance = 3706 ;
Distance = 4755 ;
false.

?- minimum_distance(london,istanbul,Distance).
Distance = 3706.

?- connected_cities(paris, CityList).
CityList = [barcelona, berlin, budapest, istanbul, milano].
```

4 Submission

You have to submit a single .pl file through Moodle with the exact file name cities_STUDENTID.pl where you replace STUDENTID with your student id.