CCM Programming Challenge: Dotsville Implementation, Part 2

This programming challenge pertains to the implementation of "low level semantics" for the Dotsville interpreter that you will soon be asked to code.

Please ...

1. Be sure that you have completed Part 1 of the Dotsville implementation exercise prior to commencing this part, Part 2, of the Dotsville implementation exercise.
2. Please think of this programming challenge as a **puzzle** of sorts, and see how much of the puzzle you can complete. I am thinking that most of you should be in a position to complete most, if not all, of it!
3. Please think of this programming challenge as a **game** of sorts, and do your best to striclty adhere to the rules of the game!

Tasks

Simply perform the following tasks in order to implement the "low level semantics" for the Dotsworld interpreter that is to come:

1. Return to the a nice place that you made in which to do all of your Dotsville programming, presumably in a directory housed within your Prolog programming directory for this course.
2. Carefully read through the `dotsville_lls_goal_demo.text` file that I am providing so that you will have a good idea about what you are intended to accomplish in this assignment.
3. In a file called `dotsville_lls.pro` within the directory that you created for all of your Dotsville assignments, please enter the code that I am distributing for this assignment by means of a file of the same name. But note: the code that I am distributing is a **redacted** version of the complete low level semantics code. Your job is largely to reconstruct the file, that is, to write the code that I have redacted.
4. Please note that I am also providing you with a "global variable data type" in the form of a file called `gv.pro`. We will talk about this data type in class. With respect to the programming challenge, it comes into play just on your last task, Task 20. Please place it next to your list processing library (`lp.pro`), and organize your files so that the `dotsville_lls.pro` code as distributed loads both files without "coughing".
5. Please load your `dotsville_lls.pro` file, noting that it will be loading:
   (a) Your `dotsville.pro` code from Part 1 of this Dotsville implementation exercise.
   (b) Your lisp processing library (`lp.pro`) from your list processing programming assignment.
   (c) The global variable library (`gv.pro`) that I am providing with this assignment.

   Also, please test this program to make sure that it works just like the program worked after you completed Part 1 of this multipart implementation assignment.
6. Perform each task, in order from Task 1 throught Task 20. **You want to do these in order, because in some cases a task will not work unless you have done the previous tasks.** Please note that:
   (a) In some cases, performing a task amounts to:
      i. Identifying were the task is presented in the file of redacted Prolog code (`dotsville_lls.pro`).
      ii. Writing the missing Prolog code.
      iii. Running the given demo to test your Prolog code. **Please do not alter the demo!**
   (b) In some cases, performing a task amounts to:

i. Identifying were the task is presented in the file of redacted Prolog code (`dotsville_lls.pro`).

ii. Reading and studying the Prolog code that I have provided for the task, to the point where you could explain how it works to someone.

iii. Running the given demo to test your Prolog code.

7. Consider the given "demos" program that is contained in the `dotsville_lls.pro` file. Note that it mostly contains lines which reference the individual demo/test programs, and that each of these lines begins with a leading comment indicator. **For those demos that worked for you, and only those demos that worked for you, remove the leading comment indicator.** Ideally, this will be all of them. Regardless, once you have removed the lead comment indicators associated with the working demos, please run the `demos` program in a Prolog shell. Then, save the results to a file called `dotsville_lls_demo.text`.

8. In the entry on your web work site that you previously established for all of the Dotsville work that you will be doing, please add (1) a reference to your reconstructed `dotsville_lls.pro` file (2) a reference to a Prolog session in which you ran the `demos` program, that is, a reference to the `dotsville_lls_demo.text` file.

Due Date

Wednesday, October 28, 2020