CCM Programming Challenge: First Programming Languages KB

For this assignment you are asked to mimic the **programming languages KB demo** presented in class. This will assure that you are capable of creating a Prolog file in a suitable text editor, and that you have no trouble consulting the file. In abstract terms, you are asked to find a way to successfully interact with Prolog either on your machine or on one of the machines in a CS Department lab so that you can create a file containing the Prolog KB, load it into a Prolog process, list the KB, and exit the Prolog process. Just think of this as a "Hello World!" program within the realm of Prolog.

Specific Tasks

1. Establish a nice location for working on materials for this course. Within that location, establish a nice location for working on your programming assignments. Within that location, establish a nice location for working on this particular assignment.
2. In your favorite text editor, type the Prolog KB about about programming languages (the 9 facts) into a file called `proglang_v1.pro`.
3. From the directory that you have established, run Prolog, consult (load) the file, list the KB, and exit Prolog - like in the example provided.
4. Capture, in the form of a data file, the text generated by the Prolog process and your interactions with it for subsequent posting to your Web page, calling the file in which you store it `proglang_v1_demo.text`.
5. In the "Course Work" section of your web work site place nicely contextualized links to both the source file and the demo file.

The Programming Languages Knowledge Base Represented in English

- Smalltalk, Lisp, and Prolog are languages.
- The essence of Smalltalk is that the object is the fundamental data type and message passing is the featured mode of computation. The essence of Lisp is that the list is the fundamental data type and the application of recursive functions is the featured mode of computation. The essence of Prolog is that the relation is the fundamental data type and logical inference is the featured mode of computation.
- The history of Smalltalk is that it was invented by Alan Kay in 1980. The history of Lisp is that it was invented by John McCarthy in 1959. The history of Prolog is that it was invented by Alan Colmeraur in 1971.

The Programming Languages Knowledge Base Represented in Prolog

```
% file:  proglang_v1.pro
% line:  some knowledge about programming languages

% language(L) means L is a programming language

language(smalltalk).
language(lisp).
language(prolog).

% essense(L,DT,CF) means language L features datatype DT
```

```
% and computational formalism CF

essence(smalltalk,object,'message passing').
essence(lisp,list,'recursive functions').
essence(prolog,relation,'logical inferencing').

% history(L,invertor(I),date(D)) means language L was invented by I in year D

history(smalltalk,inventor('Alan Kay'),date(1980)).
history(lisp,inventor('John McCarthy'),date(1959)).
history(prolog,inventor('Alan Colmeraur'),date(1971)).
```

The Demo

```
bash-3.2$ pwd
/Users/blue/blues/2020Q/courses/2020/fall/CCM/PrologProgramming/programming_languages
bash-3.2$ ls
#proglang_v1.pro# proglang_v1.pro
bash-3.2$ swipl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.2.5)
Copyright (c) 1990-2012 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- consult('proglang_v1.pro').
% proglang_v1.pro compiled 0.00 sec, 10 clauses
true.

?- listing.

language(smalltalk).
language(lisp).
language(prolog).

essence(smalltalk, object, 'message passing').
essence(lisp, list, 'recursive functions').
essence(prolog, relation, 'logical inferencing').

history(smalltalk, inventor('Alan Kay'), date(1980)).
history(lisp, inventor('John McCarthy'), date(1959)).
history(prolog, inventor('Alan Colmeraur'), date(1971)).

:- thread_local thread_message_hook/3.
:- dynamic thread_message_hook/3.
:- volatile thread_message_hook/3.

true.

?- halt.
```

```
bash-3.2$
```