Project 4: Prolog Programming

Due Date: 11/28 by 11:59p

Important Reminder: As per the course Academic Honesty Statement, cheating of any kind will minimally result in receiving an F letter grade for the entire course.

Aims of This Project

The aims of this project are as follows:

- To give you an introduction to programming in Prolog.
- To expose you to using pattern matching for both constructing and accessing data-structures.
- To familiarize you with using backtracking in Prolog.

Project Specification

You are required to submit a `prj4.tar.gz` archive which contains a file `prj4-sol.pl` which can be run using the `swipl` implementation of Prolog on `remote.cs`.

The file should implement the functions specified in the skeleton file provided. Your solutions may include auxiliary top-level procedures unless explicitly forbidden.

Provided Files

The ./files directory contains the following:

Makefile
: This file contains a `submit` target such that typing `make submit` will create a `prj4.tar.gz` archive containing the files to be submitted. The `clean` target will remove the archive.

  You may edit this file if you choose to use a different organization for your project. When editing, watch out for tabs (the first character of any command-line **must be a tab character**).

README
: A template README; replace the `XXX` with your name, B-number and email. You may add any other information you believe is relevant to your project submission. In particular, you should document the data-structure used for your word-store.

prj4-sol.pl
: A skeleton file for the exercises.

Hints

You may choose to follow the following hints (they are not by any means required). They assume that you are using the project structure supported by the provided Makefile,.

- First work on your solutions using Prolog's declarative semantics. Write facts and rules for a problem which states something about the problem domain.

  Then think about the procedural semantics. It may be a good idea to include facts/rules corresponding to base cases before facts/rules corresponding to recursive cases (this should not affect correctness, but may result in different orders of solutions or a solution versus an infinite loop).

- The swipl version on `remote.cs` is quite old. In particular, the representations of Prolog strings (required for Exercise 5) has changed in newer versions.

Submission

You will need to submit a compressed archive file `prj4.tar.gz` which contains all the files necessary to build your jar file. Additionally, this archive **must** contain a `README` file which should minimally contain your name, B-number, email, the status of your project and any other information you believe is relevant.

If you are using the suggested project structure, then the provided Makefile provides a `submit` target which will build the compressed archive for you; simply type `make submit`.

Note that it is your responsibility to ensure that your submission is complete. To test whether your archive is complete, simply unpack it into a empty directory and see if it runs correctly.

To submit the above archive, please use blackboard by following the `Content->Projects` link.