ECS 140A – Programming Languages
Project 3: Prolog

Due: December 01, Wednesday, 2021, 11:59pm PT

This project specification is subject to change at any time for clarification.

Getting Started

We will be using SWI-Prolog for the project: `https://www.swi-prolog.org`.

You may either download the SWI-Prog app and run your program using the app (please see the SWI-Prolog's website for more details), or you may install and run your program using the terminal.

To install in terimal on MacOS, use homebrew by running the command "brew install swi-prolog". To run you program, cd into the directroy where your program is stored and run the command "swipl [filename]", where "[filename]" is the name of your Prolog file. The extension for Prolog programs is ".pl".

For Windows and Linux: please check `https://wwu-pi.github.io/tutorials/lectures/lsp/010_install_swi_prolog.html`.

There are two (unrelated) parts of the project. Submit one file per part. There is no starting code for the project.

Part 1: Warm Up

You will need to define five predicates in this part.

Write all of them in a single file named "part1.pl".

1. `concat(L1, L2, L)`: meaning the concatenation of two lists L_1 and L_2 is L.
   Examples:
   - `concat([1, 2, 3], [4], [1, 2, 3, 4])` is true.
   - `concat([1, [2, 3]], [4], [1, 2, 3, 4])` is false.
   - `concat([1, [2, 3]], [4], [1, [2, 3], 4])` is true.

   You may assume the given arguments are always lists.
2. `element_at(X, N, L)`: meaning the element X is the Nth element in list L.
   Examples:
   - `element_at(2, 1, [2, 3, 4])` is true.
   - `element_at(2, 1, [3, 2, 4])` is false.
   - `element_at(2, 1, [[2], 3, 4])` is false.
   - `element_at(5, 2, [3, 2, 4])` is false.

You may assume $N \geq 1$ and L is non-empty.

3. `my_reverse(L1, L2)`: meaning the reverse of list L_1 is L_2.
   Examples:
   - `my_reverse([1,2,3], [3,2,1])` is true.
   - `my_reverse([1,2,3], [2,1,3])` is false.
   - `my_reverse([1,[2,3],4,[5]], [[5],4,[2,3],1])` is true.
   - `my_reverse([1,[2,3],4,[5]], [[5],4,[3,2],1])` is false.
   - `my_reverse([], [])` is true.
4. `my_flatten(L1, L2)`: Given a list L_1, its flattened version is L_2.
   Examples:
   - `my_flatten([1, 2, 3], [1, 2, 3])` is true.
   - `my_flatten([1, [2, 3]], [1, 2, 3])` is true.
   - `my_flatten([1, [2], [3, 4]], [1, 2, 3, 4])` is true.
   - `my_flatten([1, [2, 3, [4, 5], 6]], [1, 2, 3, 4, 5, 6])` is true.
5. `compress(L1, L2)`: Given a list L_1, L_2 is its compressed version by eliminating the duplicates.
   Examples:
   - `compress([1, 2, 3], [1, 2, 3])` is true.
   - `compress([1, 2, 2], [1, 2])` is true.
   - `compress([1, 2, [2]], [1, 2])` is false.
   - `compress([1, 2, [3, 4], [5], [5], [3]], [1, 2, [3, 4], [5], [3]])` is true.

Part 2: N-Queens Problem

Given an $N \times N$ chessboard, we want to place N queen on the board so that no two queens can attack one another. In Chess, a queen can attack another piece vertically, horizontally and diagonally. For more details, please check `https://en.wikipedia.org/wiki/Eight_queens_puzzle`. In other words, no two queens can be placed on the same row, on the same column, or on the same diagnal.

Define a predicate `queens(N, Q)` where N is the number of rows and columns of the chessboard, and Q is a list N numbers such that the ith number represents the position of the queen in column i. For example, $Q = [4, 2, 7, 3, 6, 8, 5, 1]$ means that the queen in the first column is in row 4, the queen in the second column is in row 2, etc.

Examples:

- It is impossible to place three queens in a 3×3 board in such a way.
- Given a 4×4 board, there are two ways to place the 4 queens: $[2, 4, 1, 3]$ and $[3, 1, 4, 2]$.

Write your program in a file named "part2.pl".

Submission

Please submit both files and a README file in a zipped folder.

You should avoid using existing source code as a primer that is currently available on the Internet. You are also not allowed to use the parser tools found on the Internet.

You must specify in your readme file any sources of code that you have viewed to help you complete this project. All class projects will be submitted to MOSS to determine if students have excessively collaborated. Excessive collaboration, or failure to list external code sources will result in the matter being referred to Student Judicial Affairs.