Homework 4 — Prolog — due Wednesday 6 March

Total number of points available on this homework is 400. Full credit is equivalent to 100 points.

4.1 Numerals (10 pts)

Define numerals as follows:

```
%% num(X) means: X is a numeral.
num(0).
num(s(X)) :- num(X).

%% sum(X, Y, Z) means: X, Y, Z are numerals
%%          such that Z is the sum of X and Y.
sum(X, 0, X) :- num(X).
sum(X, s(Y), s(Z)) :- sum(X, Y, Z).
```

Why can't we write simply `sum(X, 0, X).` for the first rule?

What are the answers to the following queries:

1. `?- sum(s(s(0)), s(s(s(0))), Z).`
2. `?- sum(X, Y, s(s(s(0)))).`

4.2 Multiplication (10 pts)

Continuing the preceding exercise, write the rules for multiplication `mult(X, Y, Z)`, meaning that `Z` is the product of `X` and `Y`. Multiplication is defined by the following axioms:

- $x \cdot 0 = 0$
- $x \cdot s(y) = (x \cdot y) + x$

4.3 Lists (50 pts)

The following are the rules for `islist(L)`, which means that `L` is a list.

```
islist([]).
islist([_|T]) :- islist(T).
```

1. (10 pts) Write the rules for `listlength(L, X)`, which means that `X`, a numeral *as defined in the preceding exercises*, is the length of the list `L`.
2. (15 pts) Write the rules for `listappend(L1, L2, L3)`, which means that the concatenation of lists `L1` and `L2` is the same as list `L3`.
3. (15 pts) Write the rules for `listreverse(L1, L2)`, which means that list `L2` is the reverse of list `L1`.
4. (10 pts) Write the rules for `listpalindrome(L)`, which means that list `L` is a palindrome.

4.4 Arithmetic evaluator (30pts)

The relation `ev` is between an expression in a very restricted arithmetic language described below, and a number which that expression evaluates to, in this little language. You can assume that the first argument is fully grounded.

```
?- ev(3,N).
N=3

?- ev([+,3,4], N).
N=7

?- ev([+,3,[*,8,0.25]], N).
N=5
```

Expressions in the "little language" are defined as

```
E ::= number
    | [+, E, E]
    | [*, E, E]
    | [exp, E]
    | [sin, E]
    | [cos, E]
```

These have the obvious numeric meanings when evaluated. Write the rules for `ev`.

4.5 Tic-tac-toe (40pts)

Write a Prolog program that allows a human user to play tic-tac-toe against the computer. Either the computer or the human can move first. The computer may choose its moves using any

policy whatsoever, including random. The first player to move is the X player, the second is the O player.

Suggested representations of a tic-tac-toe board include these possibilities: `[x,o,o,b,x,b,b,b,x]` or `[[x,o,o],[b,x,b],[b,b,x]]`. Both of these examples are lists intended to represent a winning board for X, in which X has conquered the bend (the diagonal from top left to bottom right).

Hints: Consider writing a predicate `xwins(L)`, meaning that the board L is a winning board for X, and a similar predicate `owins(L)`. Consider writing a predicate `okmove(L1,L2)`, meaning that from the board L1 it is legal to get to the board L2 in the very next move.

You do not have to worry about interactive input-output. It suffices to write a predicate `computersmove(L1,L2)`, which, when given an instantiated board L1, produces a unique board L2 as the outcome of the computer's move. Assume that the user will repeatedly query this predicate.

4.6 Smarter tic-tac-toe (10pts)

Same as above, but the computer should use a policy that guarantees it will not lose. Use the policy you developed in Exercise 3.2.

4.7 Analysis of tic-tac-toe (30 pts)

Consider the game of tic-tac-toe. A move maps a current board to the next board. A strategy maps a current board to a move. Continue this formalization. How can the moves and strategies be represented in a finite-state machine formalism?

4.8 Extra credit: Diplomacy (220 pts)

This is an especially challenging exercise, and will take much more time than the rest of the problems on this homework set. If you decide to work on this problem, you have until 27 March to finish it.

The rules of the game *Diplomacy* are available at the following URL:
`http://www.hasbro.com/instruct/Diplomacy.PDF`.

A wealth of other information about the game can be found at:
`http://www.diplomacy-archive.com/`.

Obtain the rules and familiarize yourself with the game.

In each move of the game, each of the seven players writes a list of *orders* for the units controlled by the player. All the orders are revealed at once, and at that point any conflicts between orders are resolved according to the rules of the game. Some moves succeed, while others fail.

Devise a representation for a configuration of the game in Prolog: represent the map (provinces and adjacency; supply centers), the units (armies and fleets), and the orders.

Write a Prolog predicate which, given a complete list of orders for one move, determines which of the orders succeed and which fail.

How to turn in

Turn in your code by running

~dmykola/handin your-file

on a regular UNM CS machine.

You should use whatever filename is appropriate in place of your-file. You can put multiple files on the command line, or even directories. Directories will have their entire contents handed in, so please be sure to clean out any cruft.

Remember to submit extensive tests of your programs!