CSE 599F1, Winter 2017
Constraint Logic Programming Assignment
Due: ~~Wed Jan 18, 1:30pm~~ Mon Jan 23, 10:00pm

This assignment is intended to give you some experience in writing CLP programs and derivation trees. The first three questions involve solving constraints over the reals (Question 1), finite domains (Question 2), and trees (Question 3).

25 points total (5 points per question).

1. Write a Prolog rule `average` that computes the average of a list of numbers, using the clpr library so that arithmetic works multi-directionally. It should fail on the empty list. Here are some example goals for which your rule should work:

   ```
   average([1,2,3],N).
   average([1,X,3],3).
   average(Xs,10).
   average(Xs,A).
   ```

2. A cryptarithmetic puzzle consists of an equation involving several numbers whose digits are represented by letters. Each digit can map to at most one letter. A solution consists of identifying the digit corresponding to each letter. The classic `SEND+MORE=MONEY` puzzle, by Henry Dudeney, was published in 1924 in *Strand Magazine*. There is a sample Prolog program on the 599 website to solve this puzzle.

   ```
     S E N D
   + M O R E
   ---------
   M O N E Y
   ```

   For this question, write a Prolog program, using the `clpfd` library, to solve the following cryptarithmetic *multiplication* problem:

   ```
         B O B
   x     B O B
   -----------
   M A R L E Y
   ```

   (The lower-case `x` denotes multiplication - it is not a digit in the problem.) The leading digits (`M` and `B`) cannot be 0. There are two solutions: find them both.

3. This question is based on the symbolic differentiation program linked from the Constraint Logic Programming lecture notes page, which in turn adapts an example in Chapter 2 of the book *Structure and Interpretation of Computer Programs*. First, load the symbolic differentiation program into SWI Prolog and try it, to make sure it is working OK and that you understand it. Note that this program justs standard Prolog without the constraint logic programming extensions. For example, the `+` in the expression `A+B` in the `basic_deriv` rule is an uninterpreted function symbol, and so this expression matches `3*x+5`, resulting in `A=3*x` and `B=5` — it doesn't try to evaluate `+`.

   You can try a particular goal, like this:

   ```
   ?- deriv(10*x+5,x,D).
   ```

and also run the provided unit tests, like this:

```
?- run_tests.
```

Now add the following extensions, by allowing the expressions to be differentiated to include:

- `-` (i.e., the minus operator)
- sin and cos
- raising an expression to an integer power

Minus should be really easy — use plus as a model. The rules for the others are as follows:

$$\frac{d(\sin u)}{dx} = \cos u(\frac{du}{dx})$$

$$\frac{d(\cos u)}{dx} = -\sin u(\frac{du}{dx})$$

$$\frac{d(u^n)}{dx} = nu^{n-1}(\frac{du}{dx})$$

For sin and cos, just simplify applying these to a number, e.g. $\sin 0$ should simplify to 0. For simplification of the power function build in the rules that anything to the 0 power is 1, and anything to the power 1 is itself. Also simplify a number raised to another number, e.g. 3^2 should simplify to 9.

Add suitable unit tests for your new rules.

4. Draw a simplified derivation tree for the following goal:

   ```
   sum([3,4],S)
   ```

   You'll need to provide the rules for `sum`, which should be easy. (Or hint: this is the last question for the second mini-exercise, on the CLP web page, and the answer is there is well.)

5. Draw a simplified derivation tree for the following goal that includes at least two answers. (The tree is infinite.)

   ```
   sum([3|As],10)
   ```

Turnin: Turn in your program in a single file using the 599 dropbox. For the trees (Questions 4 and 5), you can either turn in the answer on paper (either leave it in my mailbox or slip it under my door), or electronically using the dropbox, as you prefer.