COMP 325 **Programming Languages**

Program 3B (Interpeter 2B)

Assigned: October 11, 2021 *Due: October 20, 2021 by 11:59 PM*

This assignment is optional. If you're satisfied with your interpreter 2 score, then you can skip this and keep that score.

There were lots of problems in the second interpreter (third program) and understanding that material, both conceptually and in terms of Pyret-based language implementation, is vital to your long term success in this course. So, you have the chance to recoup some lost points. You'll be given a reduced set of expressions definitions for which you must write the appropriate parser, desugarer, and interpreter.

Interpreter 2 will be graded out of 40 possible points. By doing this assignment you can add at most 10 to that score.

Details

I am again giving you some starter code, but this time it's just data definitions. You are given an `ExprExt` and `ExprC` and a few other things. You can add other data definitions as necessary, but do not modify the ones I give you.

You need to build a parser, desugarer, and interpreter for a language that uses those data definitions. You are free to define the functions you need as necessary, with one exception (defined below).

A program in this language consists of a list of function definitions, one of which must be "main" (à la C).

Requirements

- You must have a function named "run" which takes in a string (the program) and a list of arguments. It should produce the final output of calling "main" with those arguments, i.e., it reads the S-expression, parses, desugars, and interprets.
- Have complete test coverage, including testing for runtime errors. Don't forget about applying a function to too many or too few arguments!
- Your interpreter must be environment-based, not substitution-based.
- `andExt` and `orExt` should ultimately reduce to `ifC`. **Notice the difference here!**. You need to properly carry about short-circuiting semantics.
- The multi-branching `condExt` should reduced to nested `ifC` expressions. Pay attention to the preconditions on the structure of the conditional's list of clauses. You are free to decide the top-level syntax for this as long as something parses into `condExt`, which later desugars into `ifC`.
- For full credit, I want you to use Pyret's higher-order functions like *map* and maybe even *fold* wherever possible. This would mean that you shouldn't be doing your own recursion on Lists. You can still get most of the points without this, but I think it would be most instructive if you did.