Homework 9: Implementation of Recursive List Processing Languages

This homework is due Monday, April 17. Each weekday that the homework is late you lose 10% of the possible points. One point from the total for each of the problems will be awarded for how well your answer is presented (i.e., logic and style).

1. (5 points) Do exercise 9-23 on page 369 in the text.

2. (8 points) Suppose `oh-no` and `test` are defined as follows.

   ```
   (defun oh-no ()
          (oh-no))
   (defun test (x y)
          (if (eq 0 x) 0 y))
   ```

   Then what happens when the expression `(test 0 (oh-no))` is executed by interpreters that use (i) applicative-order evaluation or (ii) normal-order evaluation?

3. This problem is intended to familiarize you with meta-circular interpreters for LISP. Code for the interpreter handed out in class is in the file `/users/class/cs342/public/interp.lsp` on Zaphod. For each part of this problem, keep track of the functions that you modify and turn in only these modified functions. (Send mail to `matthews`, with each part clearly labeled.) Please program your modifications in the same style as the interpreter itself.

   (a) (5 points) Modify the interpreter so that `square`, a function that returns the square of its numeric argument, is built-in to the interpreter (a primitive).

   (b) (20 points) Modify the interpreter to recognize the special form `define`. That is, the user code `(define` *name exp*`)` should bind the value of *exp* to *name* in the active scope. (Thus, `define` does not have to be executed at the top-level). Note that *name* is not evaluated. The interpreter should produce an error message if the symbol *name* is already bound in the current environment. The result of `define` is the symbol that is its second argument.

   Your program will have to use `rplaca` to mutate the environment.

   (c) (20 points) Modify the interpreter to recognize the special form `break`. When the user evaluates `(break` *message*`)`, the value of *message* will be printed, and then a new read-eval-print loop whose prompt should be `debug>` will be started. (The XLISP function `print` can be used to print the value of *message*. When this new read-eval-print loop starts running, its environment should be the envrionment in which `break` was executed (so that the user can examine the bindings of variables).

4. (10 points) Briefly explain the costs and benefits of a language where storage is explicitly freed (i.e., one with explicit erasure) as compared with LISP.