Homework 7: Save the Environment, Part II

Part II Due Thursday, April 9 at 9:55 am

This homework refers to the heavily commented implementation of the Eph language at http://cs.williams.edu/~morgan/www/cs334/code/eph/eph.zip, which is required reading.

6. **Elegance** (9 points)
   I used a larger set of Scheme functions and forms than you've seen before, and generally wrote in a more aggressively functional style than you are accustomed to. You will need to look some of them up in the Scheme manual (push F1 in DrScheme). Identify two design patterns that you find particularly "good" (elegant, clever, efficient, or clear) and for each:

   a. note one location where each is used,
   b. describe why it is good, and
   c. describe how the same task is generally accomplished in Java.

7. **Inelegance** (9 points)
   In some places I intentionally wrote in a very verbose or imperative style to make the code clearer to you on reading. There are likely even a few bugs still left in the code. Identify one area that could be made significantly more elegant, efficient, or correct. Give that code, your rewritten version, and an explanation of why your version is an improvement.

8. **LET** (10 points)
   Describe how to implement Scheme's LET for Eph. This should be the full-blown multi-variable version of LET, and it should use environments, not substitution. Your answer can be either the relevant pieces of an actual implementation, or a detailed description of the changes required at the design level. In either case, there are several details to which you must attend. There are many ways of implementing LET, and some are much better (i.e., *worth more points*) than others. Explain why you chose your method. Be careful not to implement LET* or LETREC by accident.

9. **Continuations** (10 points)
   Briefly describe the changes to the following areas to add continuations. Unlike previous questions, I am only looking for a high-level answer and not all of the details:

   a. Grammar
   b. eph-eval/eph-apply
   c. Library

Challenge: (glory and wisdom, but no points)

Add a new feature to the implementation. Some ideas are:

- One or more new library routines, like: foldl, reverse, eval, set-car!, set-cdr!, length
- String data type (and literals and utility functions)
- Return