CS 784 Programming Languages
Fall 2011/Home Work 1

---

1. (10 points) EOPL3 Exercise 1.12. Eliminate the one call to `subst-in-s-exp` in `subst` by replacing it by its definition and simplifying the resulting procedure. The result will be a version of `subst` that does not need `subst-in-s-exp`.
2. (10 points) EOPL3 1.20. `(count-occurrences s slist)` returns the number of occurrences of `s` in `slist`.
3. (10 points) EOPL3 Exercise 1.28. `(merge lon1 lon2)`, where `lon1` and `lon2` are lists of numbers that are sorted in ascending order, returns a sorted list of all the numbers in `lon1` and `lon2`.
4. (20 points) Define the procedure `compose` such that `(compose p1 p2)`, where p1 and p2 are procedures of one argument, returns the composition of these procedures, specified by this equation:

   ```
   ((compose p1 p2) x) = (p1 (p2 x))
   > ((compose car cdr) '(a b c d))
   b
   ```

5. (20 points) `(car&cdr s slist errvalue)` returns an expression that, when evaluated, produces the code for a procedure that takes a list with the same structure as `slist` and returns the value in the same position as the leftmost occurrence of `s` in `slist`. If `s` does not occur in `slist`, then errvalue is returned. Do this so that it generates procedure compositions.

   ```
   > (car&cdr 'a '(a b c) 'fail)
   car
   > (car&cdr 'c '(a b c) 'fail)
   (compose car (compose cdr cdr))
   > (car&cdr 'dog '(cat lion (fish dog ()) pig) 'fail)
   (compose car (compose cdr (compose car (compose cdr cdr))))
   > (car&cdr 'a '(b c) 'fail)
   fail
   ```

6. (30 points) EOPL3 Exercise 2.30 The procedure `parse-expression` as defined p53 is fragile: it does not detect several possible syntactic errors, such as `(a b c)`, and aborts with inappropriate error messages for other expressions, such as `(lambda)`. Modify it so that it is robust, accepting any s-exp and issuing an appropriate error message if the s-exp does not represent a lambda-calculus expression.

What to hand-in?

Submit on gandalf.cs.wright.edu files as follows:

```
% ~pmateti/CS784/turnin HW1 defs.scm
```

The `defs.scm` file should contain solutions to the above problems. Document your definitions well. Make sure the solution files can be loaded and run in DrRacket without any modification.

For examples of well-documented Scheme definitions, see: Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, Shriram Krishnamurthi, How to Design Programs, MIT Press, 2003. On-line version of the book is at http://www.htdp.org/ [HtDP book]
