CSSE304 Exam 1 Paper Review

Instructions

Please complete the following problems. If you have questions come up and ask. I will send out an answer key after the review session is over.

1. **Problem 1:** Write a function list-masher that takes three scheme procedures as arguments and returns a procedure that takes two lists as arguments. The procedure it returns runs procedure one on list one, procedure two on list two and procedure three on the results. Hint: think about map and apply.

```
(define important-math (list-masher abs sqrt +))

(important-math '(3 -5 -4) '(4 9 16)) -> '(5 8 8)
(important-math 1 2) -> Error
((list-masher symbol? list cons) '(a b 3) '(4 5 6)) ->
'((#t 4) (#t 5) (#f 6))
```

2. **Problem 2:** Write a function product-of-evens that takes in a list of integers and returns the product of only the even values. Use some combination of map, apply, and filter to complete the problem (no explicit recursion).

```
(product-of-evens '(1 2 3 4 5)) -> 8
```

3. **Problem 3:** Write a function product-incremented-numbers that takes in a list of values (numbers and symbols) and returns the product of only the numbers incremented by 1. Use some combination of map, apply, and filter to complete the problem (no explicit recursion).

```
(product-incremented-numbers '(1 a 3 4 f s)) -> 40
```

4. **Problem 4:** Write a function find-valid-data that takes in a list of lists of numbers and returns a list of only the positive averages. Use some combination of map, apply, and filter to complete the problem (no explicit recursion).

```
(find-valid-data '((1 2 3) (-2 -4 -2) (0 3 5 4))) -> '(2 3)
```

5. **Problem 5:** Here is some code that uses let and lambda to show closures. Write what will be displayed by the code when it is run.

```
(define factory
  (let ((global-val 100))
    (lambda ()
      (let ((local-val 0))
        (lambda ()
          (let ((temp-val 5))
            (set! global-val (- global-val 5))
            (set! local-val (+ local-val 10))
            (set! temp-val (+ temp-val 1))
            (display (list global-val local-val temp-val))
            (newline)))))))

(define f1 (factory))
(define f2 (factory))
(f1)
(f1)
(f2)
(f1)
```

6. **Problem 6:** Here is a lambda calculus expression. list which values are bound and which are free.

```
(lambda (a b)
 (lambda (c)
  (lambda (d e)
   (+ e a f (apply c g)))))
```

7. **Problem 7:** Use the following grammar and expression to create a grammar tree. Use E for expr, B for base_val, S for symbol, and I for integer.

```
<expr> ::= (<symbol> <expr> <expr>) | [<expr> <expr>]
           | <base_val>
<base_val> ::= <integer> | <symbol>

(+ a [3 (* 4 5)]))
```