6.090 IAP '05 - Homework 2

Assigned Tuesday January 11th.
Due 10am Wednesday January 12th.
Submit a print-out of your .scm file. Should you encounter difficulty printing, the file may be emailed to the 6.090 staff.

Problem 1: Evaluation

The usual: type of expression, guess the value, then evaluate.

```
(lambda (x y z) x)
((lambda (x y) (+ x y)) 4 (+ 3 4))
((lambda (happy tiger) (if (= tiger 4) (+ happy tiger) happy))
 4 5)
((lambda (wow this works) (wow this works))
 - 7 5)
((lambda (wow this works) (works wow this))
 7 - 5)
((if (= 4 5)
     (lambda (x) (+ x 3))
     (lambda (x) (+ x 5)))
 7)
(define x 2)
((lambda (x) (+ x x)) 5)
((lambda (yummy) (* yummy yummy)) 5)
yummy
```

Problem 2: Writing Procedures

For each problem, write the specified procedure while obeying the given constraints on what primitive procedures are available. Additionally, test the procedure with a couple of inputs and include these test cases and their results in your submission to demonstrate that your procedure works.

1. If you didn't finish problem 2 in the lecture handout, do so (through positive-root).
2. Write two procedures that use the following definitions to convert days into minutes and minutes into days:

```
(define seconds-per-minute 60.0)
(define minutes-per-hour 60.0)
(define hours-per-day 24.0)
(define days-per-year 365.25)
(days->minutes 1)
;Value: 1440.0
(minutes->days 1)
;Value: 0.0006944444444444445
```

3. Given the lengths of the sides of a right triangle, compute the length of the hypotenous. You may use square and sqrt.

```
(define pythagoras
  (lambda (a b)
```

4. Write the procedure XOR. XOR is a a logical operator like AND or OR, except that it has the following function: If both of the inputs are true, or both of the inputs are false, it returns false. If the inputs differ, it returns true. The truth table for XOR is as follows:

X	Y	Output
#t	#t	#f
#t	#f	#t
#f	#t	#t
#f	#f	#f

You can use `boolean=?` if you want (look it up in the reference manual).

```
(define xor
  (lambda (x y)
```

5. Write a procedure `censor` which takes as input a string, and returns either the string or the string "BEEP", depending on whether the input string contained any offensive material. For this problem, any string which contains the phrase "scheme sucks" will be deemed offensive. Thus:

```
(censor "I love scheme; it's so cool!")
;Value: "I love scheme; it's so cool!"
(censor "Dunno, but I think scheme sucks")
;Value: "BEEP"
```

You may find the primitive procedure `substring?` handy.

```
(define censor
  (lambda (s)
```

Problem 3: Writing Recursive Procedures

For each problem, first write down your **plan** (base case, recursive case) in english/math, then write the specified procedure while obeying the given constraints on what primitive procedures are available. Additionally, test the procedure with a couple of inputs and include these test cases and their results in your submission to demonstrate that your procedure works.

1. Write a procedure `slow-mul` that multiples two non-negative numbers together using only addition and subtraction (ie not * or /).

   ```
   (slow-mul 3 4)
   ;Value: 12
   Plan - Base case:
          Recursive case:  x * y = ((x-1) * y) + y
   (define slow-mul
   ```

2. Write a procedure `even?` that returns true if it's input is even. You may only use numerical comparisons and subtraction in your solution.

   ```
   (even? 35)
   ;Value: #f
   Plan - Base case:      if n < 2, n is even if n=0
          Recursive case:
   (define even?
   ```
