CS 357: Declarative Programming
Homework 2 (Spring '14)

1. Exercises 4.4, 4.6, 4.10, 4.11, 4.18, 4.19, 4.20 from Springer and Friedman.

2. Write a function, *calculator*, which takes an infix arithmetic expression and evaluates it. For example,

```
> (calculator 42)
42
> (calculator '(1 + 2))
3
> (calculator '(1 + (2 * 8)))
17
> (calculator '((((2 + 3) * 2) / 5) + (17 - 1))
18
```

   You may assume that all sub-expressions are parenthesized. In other words, you don't need to worry about precedence. Also, you need only implement the four basic arithmetic functions, namely, plus, minus, times and divide.

3. Write a function, *infix->prefix*, which takes an infix arithmetic expression and returns the corresponding prefix expression.

```
> (infix->prefix 42)
42
> (infix->prefix '(1 + 2))
(+ 1 2)
> (infix->prefix '(1 + (2 * 8)))
(+ 1 (* 2 8))
> (infix->prefix '((((2 + 3) * 2) / 5) + (17 - 1))
(+ (/ (* (+ 2 3) 2) 5) (- 17 1))
```

4. Define a function *iota-iota* which takes an integer *i* as its argument and returns a list of pairs of integers such that

```
> (iota-iota 1)
((1 . 1))
> (iota-iota 2)
((1 . 1) (1 . 2) (2 . 1) (2 .2))
> (iota-iota 3)
((1 . 1) (1 . 2) (1 . 3) (2 . 1) (2 . 2) (2 . 3)
(3 . 1) (3 . 2) (3 . 3))
```

All helper functions should be tail-recursive and should be defined within the body of *iota-iota* using *letrec*.

5. Define a tail-recursive function *digits->number* which takes a list of digits and returns the number represented by those digits. For example,

```
> (digits->number '(7 6 1 5))
7615
```

   Any helper functions you need should be defined within the body of *sin* using *letrec*. Note: There is a good way and a bad way to do this. The good way avoids computing the power of 10 from scratch each time. In other words, do not use or define *expt*.

6. Write a function, *cond->if*, which takes a *cond* expression, and transforms it into a set of nested *if* expressions. For example,

```
> (cond->if (cond ((> x y) (- x y)) ((< x y) (- y x)) (else 0)))
(if (> x y) (- x y) (if (< x y) (- y x) 0))
>
```

7. Write a tail-recursive function, *cos*, which takes a number, x, as its argument and returns $\cos(x)$. Your function should approximate $\cos(x)$ by summing the first 100 terms of the following Taylor series:

$$\cos(x) = \frac{x^0}{0!} - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \ldots$$

   Any helper functions you need should be defined within the body of *sin* using *letrec*. Note: There is a good way and a bad way to do this. The good way avoids computing the factorial and the power of x which appear in each term in the series from scratch each time. In other words, do not use or define *fact* or *expt*.