TIPLPA Programming Language Paradigms
Functional Programming Assignment 2

Due: 23:59 Thursday 15 May 2014
Version 1

Task 1

Write a procedure called `zip` that takes two lists and returns a single list of the corresponding pairs of elements from the input lists. If the lists are of unequal length, the result should be the same length as the shorter list.

```
(zip '(1 2 3) '(a b c))
=> ((1 . a) (2 . b) (3 . c))
(zip '(1 2) '(a b c))
=> ((1 . a) (2 . b))
```

This procedure *should* be tail-recursive. (Non tail-recursive versions will only be accepted with a strong rationale.)

Task 2

More folding — use `foldl` to implement:

1. Reimplement `filter`
2. Implement a procedure, `sumprod`, that calculates the sum of the product of the number in two lists. For example,

   ```
   (sumprod '(1 2 3) '(4 5 6))      => 32
   ```

3. Implement a procedure, `dedup`, to deduplicate a list.

   ```
   (dedup '(1 1 2 2 3 3 4 5 5))     => (1 2 3 4 5)
   (dedup '(1 1 2 1 1))             => (1 2 1)
   ```

Task 3

Consider a variation on fold:

```
(define foldr
  (lambda (proc lst acc)
    (if (null? lst)
        acc
        (proc (car lst)
              (foldr proc (cdr lst) acc)))))
```

Part 1: Use `foldr` to (re)implement `map` and `reverse`
Part 2: How does this version of fold differ from `foldl`? Answer in commented block in your `.rkt`/`.scm` file.
Part 3: Why do fold functions take an accumulator/initial value parameter? Answer as above.

Task 4

TBD

Other Notes

Remember that your submitted file(s) must start with a header like the one below. If it does not, your assignment will not be accepted. It is up to you as to whether you wish to hand in a single file, or one file per Task.

One submission per group of two; remember to include all both people in the CampusNet submission.

```
;; FP2.rkt
;; Joey Coleman <jwc@eng.au.dk> 970072541
;; Stefan Hallerstede <sha@eng.au.dk> 987654321
;;
```