

# Premiers Cours de Programmation avec Scheme

(du fonctionnel pur aux objets avec DRACKET)

(c) Editions ELLIPSES, 2010

27 octobre 2011

# Errare humanum est !

*Les inévitables (?) erreurs et coquilles... Envoyez-moi celles que vous voyez, merci!*

## page 35

Pour obtenir un entier aléatoire de  $[10, 20]$ , on utilise `(+ 10 (random 11))` et non `(+ 10 (random 21))`.

## page 108

Le membre gauche de la formule à prouver est  $1^2 + 2^2 + \dots + n^2$  et non  $(1 + 2 + \dots + n)^2$ .

## page 143

`(map (lambda (x) (+ x 1)) '(1 2 3 4))` donne `(2 3 4 5)` et non `(3 5 7 9)`.

## page 147

Cette page n'a rien à faire dans ce chapitre puisque les fonctions d'arité variable ne sont acceptées qu'en vrai langage SCHEME et non – hélas – en niveau *Etudiant avancé*. Elle devrait être située après la page 229, en vrai langage SCHEME (chapitre 11)! A moins que les implémenteurs de RACKET (*the racketeers*) accèdent dans une version ultérieure à ma demande de les faire reconnaître en niveau *Etudiant avancé*... Wait and see, et voir l'erratum qui suit.

## pages 158-159

Ces exercices sont mal placés, en vertu de l'erratum précédent. Ils sont donc rejetés au chapitre 11 et leur solution se trouve au début des solutions du chapitre 11.

**page 167**

A la ligne 77, remplacez `'valeur` par `'simplif-const`. On ne méfiera jamais assez du copier-coller ! A la ligne 79, il manque l'appel à `simplif-const`.

**page 173**

A la ligne 142, remplacer `P` par `pile`.

**page 201**

A la troisième ligne, remplacez : *Les deux solutions ne produisent pas le même code*, par : *Qu'en pensez-vous ?*.

**page 251**

<pre>90   &gt; (define mfib (make-memo-fib)) 91   ; void pour cacher le résultat 92   &gt; (time (void (mfib 10000))) 93   cpu time: 22 ms 94   ; le résultat est mémorisé !</pre>	<pre>95   &gt; (time (void (mfib 10000))) 96   cpu time: 0 ms 97   &gt; (time (mfib 50)) 98   cpu time: 2 ms 99   12586269025</pre>
--	---

**page 309**

La structure cachée des modules ayant changé pendant la rédaction du livre, *Racket* est passé de `(module foo racket ...)` à `(module foo racket (%module-begin ...))`, ce qui demande une modification de la ligne 459 en :

```
(length (filter def? (caddr x)))
```

Néanmoins, le livre est bien correct en haut de la page 301...

**page 312**

Dans l'exercice 13.7.9 question b), il n'y a pas de tag fermant `</img>`, seulement le chevron `>`.

**pages 401 et 410**

Dans cette section, la fonction `take` a pour format `(take n L)` comme en HASKELL et non pas `(take L n)` par endroits [ce qui est le choix de la SRFI-1].

**page 410**

*Remplacer les deux lignes qui précèdent la ligne 479 par :*

Programmons le flot de Fibonacci `FIBS` vu aux pages 401 et 402. Nous aurons besoin d'une fonction `map` sur les flots. Écrivons-en directement une version `n`-aire :