Premiers Cours de Programmation avec Scheme

(du fonctionnel pur aux objets avec DrRacket)

2 octobre 2017

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 $1^2+2^2+\ldots+n^2$ et non $(1+2+\ldots+n)^2$.

page 129

En ligne 55, il faut lire `false` au lieu de `true`.

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 des pages 158-159.

page 157

À l'exercice 8.13.11, on ne fait figurer que les coefficients non nuls !

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

À 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

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

page 201

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

page 251

```
> (define mfib (make-memo-fib))
 ; void pour cacher le résultat
> (time (void (mfib 10000)))
cpu time: 22 ms
 ; le résultat est mémorisé !
> (time (void (mfib 10000)))
cpu time: 0 ms
> (time (mfib 50))
cpu time: 2 ms
12586269025
```

page 302-305

Les procédés (c'est vrai un peu rupestres) de la section 13.5 sont de moins en moins acceptés par les serveurs Web. Pour faire du *web scrapping* – du surf programmatique sur les pages web à la recherche d'informations – on peut demander un `(require net/url)` pour lire une page web d'un seul coup dans une énorme string. Il suffira ensuite d'analyser cette string à coups d'expressions régulières. Des méthodes plus professionnelles sont

disponibles pour lire du texte structuré comme HTML ou XML, mais ce n'était pas le but de ce livre.

```
> (require net/url)
> (define str                      ; une chaîne en mémoire centrale
    (port->string
      (get-pure-port
        (string->url "http://deptinfo.unice.fr/~roy/index.html"))))
> (regexp-match "Ar[a-z]*" str)
("Armstrong")
```

Si vous persistez à vouloir lire une page HTML ligne à ligne (par exemple si cette page est énorme), il est bon d'abandonner les requêtes `GET` et obtenir directement un port accessible en lecture sur la page HTML souhaitée. Exemple :

```
> (define p-in
    (get-pure-port
      (string->url "http://deptinfo.unice.fr/~roy/index.html")))
> (read-line p-in)
"<!DOCTYPE HTML>"
> (read-line p-in)
"<html lang=\"fr\">"                    ; etc
```

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? (cadddr 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 les trois lignes suivantes :

Programmons le flot de Fibonacci `FIBS` vu en Haskell en ligne 327 et en Lazy Racket aux lignes 337-338. Nous aurons besoin d'une fonction `map` sur les flots. Écrivons-en directement une version n-aire :