ISIMA 1ère année — 1er septembre 2011
Durée : 2 heures

Documents autorisés

PROGRAMMATION FONCTIONNELLE

Pensez à expliciter en français ce que doivent faire vos fonctions.
Vous pourrez bien sûr écrire des fonctions intermédiaires.

Exercice 1 :

Soit `mapc` une fonction ayant comme arguments une fonction `f` de deux arguments et une liste de couples `LC` du type `(`(x_1 y_1) `...` (x_n y_n)`)`, et telle que l'évaluation de l'expression `(mapc f` (x_1 y_1) `...` (x_n y_n)`)` retourne la liste des valeurs obtenues en évaluant `f` avec x_1 et y_1, x_2 et y_2, …, x_n et y_n.

Par exemple, l'évaluation de l'expression `(mapc + '((1 2) (3 4))` doit retourner la liste `(3 7)`.

1°) Ecrire la fonction `mapc` de manière récursive (`mapc` doit, en général, rappeler `mapc`).

Solution :

```
(define mapc (lambda (f LC)
   (if (null? LC) ()
       (cons (f (caar LC) (cadar LC)) (mapc f (cdr LC))) ) ))
```

2°) Ecrire la fonction `mapc` de manière non récursive en utilisant un schéma d'application itératif de type `map`.

Solution :

```
(define mapc (lambda (f LC)
   (map f (map car LC) (map cadr LC)) ))
```

Exercice 2 :

Ecrire une fonction `separer` ayant comme arguments un prédicat `P` et une liste `L` et telle que l'évaluation de l'expression `(separer P L)` retourne une liste contenant deux sous-listes : la sous-liste des éléments de `L` qui satisfont `P` et la sous-liste des éléments de `L` qui ne satisfont pas `P`.

Par exemple l'évaluation de l'expression `(separer number? '(2 (+ x 2) z 3))`

doit retourner la liste `((2 3) ((+ x 2) z))`.

Solution :

```
(define separer (lambda (P L)
   (if (null? L) '(() ())
       (let ((L2 (separer P (cdr L))))
         (if (P (car L)) (list (cons (car L) (car L2)) (cadr L2))
                         (list (car L2) (cons (car L) (cadr L2))) ) ) ) ))
```

Ou bien :

```
(define separer (lambda (P L) (sep P L () ())))
(define sep (lambda (P L L1 L2)
   (if (null? L) (list L1 L2)
       (if (P (car L)) (sep P (cdr L) (cons (car L) L1) L2)
                       (sep P (cdr L) L1 (cons (car L) L2)) ) ) ))
```

Exercice 3 :

Etant données deux fonctions *f* et *g* de $\mathbb{R}$ dans $\mathbb{R}$, on note usuellement *f* + *g* la fonction qui à tout nombre réel *x* associe *f* (*x*) + *g* (*x*). Le symbole "+" dans l'expression "*f* + *g*" est une opération sur les fonctions f et g :

1°) Ecrire une fonction `OF+` ayant comme arguments deux fonctions `f` et `g` de $\mathbb{R}$ dans $\mathbb{R}$, et telle que l'évaluation de l'expression `(OF+ f g)` retourne la fonction `f + g` : `OF+` est donc l'opérateur fonctionnel associé à l'opérateur +.

Solution :

```
(define OF+ (lambda (f g)
    (lambda (x) (+ (f x) (g x))) ))
```

2°) Généralisons la notation définie ci-dessus à un opérateur binaire quelconque, noté *op* : étant données deux fonctions *f* et *g*, on notera *f op g* la fonction qui à tout argument *x* associe *f* (*x*) *op g* (*x*). Le symbole "*op*" dans l'expression "*f op g*" est donc l'opérateur fonctionnel associé à l'opérateur *op*.

Ecrire une fonction `OF` ayant comme argument un opérateur binaire `op`, et telle que l'évaluation de l'expression `(OF op)` retourne l'opérateur fonctionnel associé à l'opérateur *op*.

Par exemple, l'évaluation de l'expression `(OF +)` doit retourner une fonction équivalente à la fonction `OF+` de la question précédente.

Donner un exemple d'utilisation de cette fonction pour calculer l'image d'un nombre réel x par la fonction f + g.

Solution :

```
(define OF (lambda (op)
   (lambda (f g)
      (lambda (x) (op (f x) (g x))) ) ))
```

Exemple d'utilisation de cette fonction pour calculer l'image d'un nombre réel x par la fonction f + g : `(((OF +) f g) x)`

Exercice 4 :

1°) Indiquer succinctement (5 lignes maximum) ce que réalise le programme suivant :

```
 (define FCT
  (lambda (C F L)
    (if (null ? L) F
      (let ((A (FCT C F (cdr L))))
          (cons (C (car L) (car A)) A) ) ) ) )
(define (decrire T)
     (apply + (FCT (lambda(x y) (+ T y))
                   '(1)
                   '(1 2 3 4 5 6 7 8 9 10) )) )
```

2°) Que retourne l'appel `(decrire 0.001` ?