Devoir sur table – Novembre 2001
MIAS 1ère année – 1er semestre
Durée : 1h30

Aucun document ni machine électronique n'est permis à l'exception de la carte de référence de Scheme.

Répondre sur la feuille même, dans les boîtes appropriées. La taille des boîtes suggère le nombre de lignes de la réponse attendue. Le barème (total sur 30) apparaissant dans chaque boîte n'est donné qu'à titre indicatif.

La clarté des réponses et la présentation des programmes seront appréciées. Toutes les fonctions apparaissant dans les réponses doivent être accompagnées de leur spécification. Ne pas désagrafer les feuilles.

## Exercice 1

**Question 1.1** – Quelle est la valeur de `(cons 1 (cons 2 (list 5 4)))` ?

Réponse [2/30]
C'est (1 2 5 4).

**Question 1.2** – Donnez une expression en Scheme ayant pour valeur `((1 2) (3))` en n'utilisant que des appels à `list` sans usage de la forme spéciale `quote`.

Réponse [2/30]
Par exemple :

```
(list (list 1 2) (list 3))
```

**Question 1.3** – Donnez une expression en Scheme ayant pour valeur `((1 2) (3))` en n'utilisant que des appels à `cons`. Il est possible de faire usage de la forme `quote`.

Réponse [3/30]
Par exemple :

```
(cons (cons 1 (cons 2 '()))
      (cons (cons 3 '())
            '() ) )
```

*ou*

```
'((1 2) (3))
```

*ou*

```
(cons '(1 2) '((3)))
```

Et de nombreuses autres possibilités s'offrent.

Copyright © 2001 by UPMC/DEUG/MIAS. 

| Section | Groupe | Nom | Prénom |
|---|---|---|---|
| | | | |

**Question 1.4** – Indiquer, en mentionnant leur nature (constructeur, reconnaisseur ou sélecteur), les fonctions que vous connaissez sur le type `LISTE[]`.

Réponse [3/30]
On trouve les accesseurs (`car` et `cdr`), les constructeurs (principalement `cons` mais aussi `list` et `append`) et les reconnaisseurs (`pair?` et `list?`).

# Exercice 2

Un nombre naturel non nul $n$ de représentation décimale $\overline{n_i n_{i-1} \ldots n_1 n_0}$ (avec $n_i \neq 0$) est dit *théosophiquement équilibré* lorsque la somme des chiffres de rang pair qui le représentent en base décimale est égale à la somme des chiffres de rang impair ou, en termes plus formels, lorsque $n_0 + n_2 + \ldots + n_{2j} = n_1 + n_3 + \ldots + n_{2j+1}$ lorsque le nombre a un nombre pair $(2j+2)$ de chiffres ou $n_0 + n_2 + \ldots + n_{2j+2} = n_1 + n_3 + \ldots + n_{2j+1}$ lorsque le nombre a un nombre impair $(2j+3)$ de chiffres.

Aucun nombre à un seul chiffre n'est théosophiquement équilibré. Les multiples de 11 inférieurs à 100 sont les seuls nombres théosophiquement équilibrés entre 10 et 100, à savoir 11, 22, 33, ... 99. Parmi les nombres théosophiquement équilibrés à 3 chiffres, on trouve 121 ou 154 ou encore 352. Donnons simplement comme exemples de nombres théosophiquement équilibrés à 4 chiffres : 5445 ou 5742.

**Question 2.1** – Écrire une fonction `somme` prenant une liste d'entiers quelconques et calculant la somme de ces nombres. Ainsi :

```
(somme (list 1 2 3 4))  → 10
```

Réponse [4/30]

```
;;;somme: LISTE[int] -> int
;;;(somme L) renvoie la somme des entiers présents dans L
(define (somme L)
  (reduce + 0 L) )

(define (somme L)
  (if (pair? L)
      (+ (car L) (somme (cdr L)))
      0 ) )
```

**Question 2.2** – Écrire une fonction nommée `representation-liste` prenant un entier naturel non nul $n = \overline{n_i n_{i-1} \ldots n_1 n_0}$ et retournant la liste des chiffres qui composent sa représentation décimale c'est-à-dire `(`$n_0$ $n_1$ ... $n_i$`)`. Ainsi

```
(representation-liste 245)  → (5 4 2)
```

Réponse [4/30]

```
;;;representation-liste: nat -> LISTE[Chiffre]
;;;(representation-liste n) renvoie la liste des chiffres composant
;;;la représentation décimale de n.
(define (representation-liste n)
  (if (< n 10)
      (list n)
      (cons (remainder n 10)
            (representation-liste (quotient n 10)) ) ) )
```

| Section | Groupe | Nom | Prénom |
|---|---|---|---|
| | | | |

**Question 2.3** – Écrire les fonctions nommées `pairs` (resp. `impairs`) prenant une liste et retournant la liste des termes de rang pair (resp. impair). On compte le rang à partir de zéro. Ainsi :

```
(pairs (list 1 2 3 4 5))  → (1 3 5)
(pairs (list 1 2 3 4 5 6))  → (1 3 5)
(impairs (list "un" "deux" "trois"))  → ("deux")
(impairs (list "un" "deux" "trois" "quatre"))  → ("deux" "quatre")
```

Réponse [6/30]

Ces deux fonctions peuvent s'épauler mutuellement :

```
;;;pairs: LISTE[a] -> LISTE[a]
;;;(pairs L) renvoie la liste des termes de rang pair de L.
(define (pairs L)
  (if (pair? L)
      (cons (car L) (impairs (cdr L)))
      (list) ) )

;;;impairs: LISTE[a] -> LISTE[a]
;;;(impairs L) renvoie la liste des termes de rang impair de L.
(define (impairs L)
  (if (pair? L)
      (pairs (cdr L))
      (list) ) )
```

On peut également les rendre autonomes et écrire :

```
;;;pairs: LISTE[a] -> LISTE[a]
;;;(pairs L) renvoie la liste des termes de rang pair de L.
(define (pairs L)
  (if (pair? L)
      (cons (car L)
            (if (pair? (cdr L))
                (pairs (cddr L))
                '() ) )
      '() ) )

;;;impairs: LISTE[a] -> LISTE[a]
;;;(impairs L) renvoie la liste des termes de rang impair de L.
(define (impairs L)
  (if (pair? L)
      (if (pair? (cdr L))
          (cons (cadr L) (impairs (cddr L)))
          '() )
      '() ) )
```

**Question 2.4** – Écrire un prédicat nommé `theosophiquement-equilibre?` prenant un entier naturel non nul $n$ et vérifiant que $n$ est théosophiquement équilibré.

Section | Groupe | Nom | Prénom

Réponse [6/30]

On a maintenant tous les utilitaires nécessaires pour l'écrire directement :

```
;;;theosophiquement-equilibre?: nat/>0/ -> bool
;;;(theosophiquement-equilibre? n) vérifie qu'un entier naturel est théosophiquement équilibré.
(define (theosophiquement-equilibre? n)
  (if (< n 10)
      #f
      (let ((chiffres (representation-liste n)))
        (= (somme (pairs chiffres))
           (somme (impairs chiffres)) ) ) ) )
```

Copyright © 2001 by UPMC/DEUG/MIAS. 