Examen – Module d'informatique 1 – Janvier 2000
MIAS 1ère année – 1er semestre

Aucun document ni machine électronique n'est permis.

Répondre sur la feuille même, dans les boîtes appropriées. La taille des boîtes suggère le nombre de lignes de la réponse attendue (utiliser le dos de la feuille précédente si la réponse déborde des boîtes). Le barème (sur 50) 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 désagrapher les feuilles.

Exercice 1

Écrire une fonction, nommée `cube`, calculant le volume d'un cube d'arête a.

[3/50]

Exercice 2

DANS CET EXERCICE, ON SUPPOSE NE PAS AVOIR D'ENTIERS EN SCHEME. On représentera les entiers naturels par des *EntierBaton*s c'est-à-dire des listes de bâtons. Ainsi 0 sera-t'il représenté par la liste vide, 1 par la liste `(I)`, 2 par la liste `(I I)` et ainsi de suite.

Question 2.1 – Écrire un prédicat, nommé `positif?`, testant si un *EntierBaton* est strictement positif.

[3/50]

Section | Numéro d'anonymat

Question 2.2 – Écrire une fonction, nommée `plus`, additionnant deux *EntierBaton*s et retournant leur somme sous la forme d'un *EntierBaton*. Par exemple,

`(plus '(I I I) '(I I))` → `(I I I I I)`

[4/50]

Question 2.3 – Écrire une fonction, nommée `moins1`, qui prend un *EntierBaton* strictement positif et lui retranche l'*EntierBaton* correspondant au nombre 1.

[3/50]

Section Numéro d'anonymat

Question 2.4 – Écrire une fonction, nommée `fois`, multipliant deux *EntierBatons* et retournant leur produit sous la forme d'un *EntierBaton*. On pourra utiliser la relation suivante :

$$\text{pour } p >= 1,\ pq = q + (p-1)q$$

[5/50]

Exercice 3

Le but de cet exercice est de définir un prédicat, nommé `equilibre?`, qui prend en arguments un prédicat p et une liste l. Le prédicat `equilibre?` retourne vrai s'il y a, dans la liste l, autant de termes qui satisfont p que de termes qui ne le satisfont pas. Par exemple,

```
(equilibre? zero?  '(1 0 0 5 4))    → #f
(equilibre? number? '(1 a c 3 e 5)) → #t
```

Question 3.1 – Dans un premier temps, on demande d'écrire une fonction, nommée `compter`, qui prend les mêmes arguments que `equilibre?` et qui retourne la différence entre le nombre de termes de la liste l qui satisfont le prédicat p et le nombre de termes de la liste l qui ne satisfont pas le prédicat p.

[4/50]

Section Numéro d'anonymat

Question 3.2 – Définir maintenant le prédicat `equilibre?`.

[3/50]

Exercice 4

Soit la propriété booléenne définie par le prédicat ternaire *coeurCarreau* définie sur les entiers naturels par les règles suivantes :

$$\begin{array}{ll} coeurCarreau(a,0,c) = \text{ Vrai} & \text{si } a = c \\ coeurCarreau(a,0,c) = \text{ Faux} & \text{si } a \neq c \\ coeurCarreau(a,b,c) = coeurCarreau(a-b,b,c) & \text{si } a > b \\ coeurCarreau(a,b,c) = coeurCarreau(a,b-a,c) & \text{si } a <= b \end{array}$$

Question 4.1 – Que vaut $coeurCarreau(24,8,4)$? Que vaut $coeurCarreau(24,16,8)$?

[2/50]

Section Numéro d'anonymat

Question 4.2 – Écrire un prédicat ternaire, que l'on invoquera comme `(coeur-carreau` a b c`)`, implantant en Scheme le calcul de l'expression $coeurCarreau(a, b, c)$.

[4/50]

Exercice 5

Soit le petit langage, nommé CC, défini par la grammaire suivante (qui comporte quatre règles) :

<*programme*>	→	(<*programme*> `OU` <*programme*>)	Règle 1
		(<*expression*> `COEUR` <*expression*> `CARREAU` <*expression*>)	Règle 2
<*expression*>	→	<*naturel*>	Règle 3
		(<*expression*> - <*expression*>)	Règle 4

On rappelle qu'un entier naturel est un entier positif ou nul.

Question 5.1 – Donner deux exemples de S-expressions satisfaisant cette grammaire et utilisant au moins une fois les quatre règles qu'elle définit.

[2/50]

Question 5.2 – Qu'est-ce qu'une barrière d'abstraction ?

[3/50]

Section | Numéro d'anonymat

Dans le langage CC, la notation p `OU` q représente la disjonction (le « ou » des booléens) des valeurs de vérité p et q. Les expressions composées avec les symboles `COEUR` et `CARREAU` utilisent pour être évaluées, la fonction `coeur-carreau` de l'exercice précédent. Plus précisément, la valeur du programme `(`a `COEUR` b `CARREAU` c`)` est la valeur de $coeurCarreau(a, b, c)$. La soustraction est repérée par le symbole `-` en position infixe.

L'évaluateur du langage CC utilise une barrière d'abstraction syntaxique dont voici quelques éléments. On suppose que programmes et expressions sont syntaxiquement corrects.

```
;;; Programme[correct] -> bool
(define (disjonction? p)
  (equal? (cadr p) 'OU) )

;;; Expression[correcte] -> bool
(define (nombre? e)
  (number? e) )

;;; Expression[nombre] -> int
(define (nombre-valeur n)
  n )

;;; Programme[coeur-carreau] -> Expression
(define (coeur-carreau-operande-1 p)
  (car p) )

;;; Programme[coeur-carreau] -> Expression
(define (coeur-carreau-operande-2 p)
  (car (cddr p)) )

;;; Programme[coeur-carreau] -> Expression
(define (coeur-carreau-operande-3 p)
  (car (cddr (cddr p))) )
```

Question 5.3 – Définir les quatre sélecteurs manquant dans la précédente barrière d'abstraction syntaxique.

[2/50]

Section	Numéro d'anonymat

Question 5.4 – Compléter les définitions suivantes.

```
;;; Évalue un programme
;;;   Programme -> booléen
(define (programme-eval p)
  (if (disjonction? p)
      cas de la règle 1
      cas de la règle 2 ) )

;;; Évaluer une expression
;;;   Expression -> entier
(define (expression-eval e)
  (if (nombre? e)
      cas de la règle 3
      cas de la règle 4 ) )
```

[7/50]

Section | Numéro d'anonymat

Exercice 6

Examiner la fonction suivante :

```
(define (mystere o)
  (define (secret o)
    (if (pair? (cdr o))
        (secret (cdr o))
        (mystere (car o)) ) )
  (if (pair? o)
      (secret o)
      o ) )
```

Question 6.1 – Évaluer à la main l'expression `(mystere '((a c e) f (b d)))`. Quelles sont les (environ) sept étapes principales de cette évaluation ?

[2/50]

Question 6.2 – Donner une spécification aux fonctions `mystere` et `secret`.

[3/50]

