Université de La Réunion jeudi 30 août 2001

DEUG 1^{ere} année – MIAS 1 et MASS 1

UE4 – Informatique 2 – Examen terminal – 2e sesion

Durée : 2 heures – Aucun document n'est autorisé.

Répondre uniquement dans les cadres prévus à cet effet

Nom :	N° étudiant :	Signature
Prénom(s) :		
Date de naissance :		

(3 pts) **Exercice 1 :** Écrivez en Scheme une fonction **truc** qui prend une liste plate d'entiers et renvoie une liste formée du nombre d'entiers strictement positifs et du nombre d'entiers multiples de 3.
Exemple : `(truc '(1 -4 0 6 0 3 -9 7 12))` *renvoie* `(5 6)`.

Exercice 2 : Dans cet exercice, on représente des ensembles par des listes dans lesquelles chaque atome n'apparaît qu'un fois.

(1 pt) a) Écrivez un prédicat **est-ensemble?** qui teste si une liste représente un ensemble.

Tournez SVP $\Longrightarrow$

(2 pts) b) Écrivez une fonction **union** qui réalise l'union de deux ensembles **E** et **F**.

(2 pts) c) Écrivez une fonction **intersection** qui réalise l'intersection de deux ensembles **E** et **F**.

Exercice 3 : Étant donnés deux entiers a et b, on veut calculer la somme $S(n) = \sum_{i=0}^{n} a^i b^{n-i}$ qu'on peut définir par récurrence :

$$S(0) = 1 \quad ; \quad S(n) = a^n + bS(n-1) \quad \forall n \geq 1$$

(1 pt) a) Écrivez une fonction récursive directement inspirée de la définition par récurrence ci-dessus.

(2 pts) b) Écrivez une fonction récursive terminale calculant $S(n)$

(5 pts) **Exercice 4 :** Dans cet exercice, vous préciserez en fonction de quelle mesure des données vous évaluez les complexités, vous expliciterez les équations de récurrence permettant de calculer ces complexités et après les avoir résolues vous donnerez leur ordre de grandeur en utilisant la notation O.

Évaluer la complexité en temps des fonctions suivantes :

```
(define sauf
  (lambda (o l)
    (cond ((null? l) '())
          ((equal? o (car l)) (sauf o (cdr l)))
          (else (cons (car l) (sauf o (cdr l)))))))
```

```
(define enleve
  (lambda (l1 l2)
    (if (null? l1)
        l2
        (enleve (cdr l1) (sauf (car l1) l2)))))
```

Tournez SVP $\Longrightarrow$

(4 pts) **Exercice 5 :** Donnez le résultat de l'évaluation des expressions suivantes :

```
>(apply append '((1) ((a)) (a(b(c)))))


>(map cdr '((1 2) (2 (3 a))))


>(map car '((2 3 4) () (a (b))))


>(map * '(1 2) '(5 6) '(3 2))


>(map (lambda (x) (or (odd? x) (>0 x))) '(-7 9 4 0 ))


>((lambda (a b c) ( / (* a a ) (- b c) ) ) 4 6 4 )


>(apply * (map (lambda (x) 2) '(2 (1 1) 0 (4 (6 5))) ))


>(apply + (map (lambda (l) (+ (cadr l) 1) ) '((1 2) (2 3) (3 4))))
```