Université Lyon 1 - Licence STS - UE LIFAP2 – 15 mars 2021 – 45mn
Aucun document autorisé – Calculatrices et téléphones interdits

Nom : **Prénom :**

Numéro d'étudiant-e :

Premier exercice (2 points)

Donner les résultats de l'évaluation par l'interpréteur Scheme des expressions suivantes :

```
(cons (car '(a (b))) (list 'c '(d e)))          ➔ …… (a c (d e)) ……
(cons (list 'a '(b c)) (append '(d) '(e (f))))  ➔ …… ((a (b c)) d e (f)) ……
```

Deuxième exercice (4 points)

On définit les listes suivantes :

```
(define L1 '(a b c))
(define L2 '(d e f))
```

Donner les expressions Scheme utilisant L1 et L2 et permettant d'obtenir les résultats suivants :

……(list (cadr L1) (cdr L2))…… ➔ `(b (e f))`

……(append (cdr L1) (cons (car L2) (cddr L2))…… ➔ `(b c d f)`

ou (append (cdr L1) (list (car L2) (caddr L2)))

Troisième exercice (7 points)

Définir une fonction en Scheme qui, étant donnés une liste d'entiers L et un entier n, retourne la liste des éléments de L divisibles par n.

```
(divisibles '(7 3 6 8 4 12) 3) -> (3 6 12)
```

Indication : x est divisible par n si (modulo x n) vaut 0.

```
(define divisibles ; -> liste d'entiers                ; 1 pt pour l'entête
 (lambda (l n) ; liste d'entiers, entier               ; 1 pt pour les commentaires
  (if (null? l)                                        ; 0.5 pt
     '()                                               ; 0.5 pt
     (if (= 0 (modulo (car l) n))                      ; 1,5 pt
        (cons (car l) (divisibles (cdr l) n))          ; 1,5 pt
        (divisibles (cdr l) n)))))                     ; 1 pt
```

Quatrième exercice (7 points)

Définir une fonction qui, étant donnés un nombre x et une liste de nombres L, compte **en un seul passage** le nombre d'éléments de L strictement inférieurs à x et le nombre d'éléments de L supérieurs à x.

```
(NbInfSup 9 '(7 4 9 2 6 10 8)) -> (5 2)
```

```
(define NbInfSup ; -> liste de 2 nbs                          ; 0,5 pt pour l'entête
  (lambda (n l) ; un nombre et une liste de nombres           ; 0.5 pt pour les commentaires
    (if (null? l)                                             ; 0.5 pt
        '(0 0)                                                ; 0.5 pt
        (let ((R (NbInfSup n (cdr l))))                       ; 3 pts pour le let
          (if (< (car l) n)                                   ; 0,5 pt
              (list (+ (car R) 1) (cadr R))                   ; 0,75 pt
              (list (car R) (+ 1 (cadr R))))))))              ; 0,75 pt
                                                              (donc 4 pts si fonction ok sans let)
Ou

(define NbInfSup ; -> liste de 2 nbs                          ; 0,5 pt pour l'entête
  (lambda (n l) ; un nombre et une liste de nombres           ; 0.5 pt pour les commentaires
    (nis n l 0 0)))                                           ; 1 pt

(define nis ; -> liste de 2 nbs                                   ; 0.5 pt pour les commentaires
  (lambda (n l ni ns) ; un nombre, une liste de nombres, 2 entiers ; 0,5 pt pour les param
    (if (null? l)                                                       ; 0.5 pt
        (list ni ns)                                                    ; 1 pt
        (if (< (car l) n)                                               ; 0,5 pt
            (nis n (cdr l) (+ ni 1) ns)                                 ; 1 pt
            (nis n (cdr l) ni (+ ns 1))))))                             ; 1 pt
```