Université Lyon 1 - Licence STS - UE LIFAP2 – 16 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 :

`(cdr (append '(a (b)) (list 'c 'd)))` ➔ …… ((b) c d)……

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

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

……(list (list (car L2) (car L1)) (cdr L1) (cdr L2))…… ➔ `((d a) (b c) (e f))`

Troisième exercice (7 points)

Définir une fonction en Scheme qui, étant donnés une liste de nombres L et deux nombres Inf et Sup, retourne la liste des éléments de L compris entre Inf et Sup inclus.

```
(filtre '(2 1 6 4 8 3 9 7) 2 7) -> (2 6 4 3 7)
```

```
(define filtre ; -> liste de nb                          ; 1 pt pour l'entête
  (lambda (l inf sup) ; liste de nb et 2 nb              ; 1 pt pour les commentaires
    (if (null? l)                                        ; 0.5 pt
        '()                                              ; 0.5 pt
        (if (and (>= (car l) inf) (<= (car l) sup))      ; 1,5 pt
            (cons (car l) (filtre (cdr l) inf sup))      ; 1,5 pt
            (filtre (cdr l) inf sup)))))                 ; 1 pt
```

Quatrième exercice (7 points)

Définir une fonction qui, étant donnés une liste d'entiers L et un entier n, construit **en un seul passage** une liste de 2 sous-listes : celle contenant les éléments de L divisibles par n et celle contenant les éléments de L non divisibles par n.

```
(trieDiv '(5 3 6 8 9 10) 3) -> ((3 6 9) (5 8 10))
```

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

```
(define trieDiv ; -> liste de 2 listes d'entiers                    ; 0,5 pt pour l'entête
  (lambda (l n) ; l liste d'entier, n entier                        ; 0.5 pt pour les commentaires
    (if (null? l)                                                   ; 0.5 pt
        '(() ())                                                    ; 0.5 pt
        (let ((R (trieDiv (cdr l) n)))                              ; 3 pts pour le let
          (if (= 0 (modulo (car l) n))                              ; 0,5 pt
              (list (cons (car l) (car R)) (cadr R))                ; 0,75 pt
              (list (car R) (cons (car l) (cadr R))))))))           ; 0,75 pt
                                                                    (donc 4 pts si fonction ok sans let)
Ou

(define trieDivDesc ; -> liste de 2 listes d'entiers                ; 0,5 pt pour l'entête
  (lambda (l n)  ; l liste d'entier, n entier                       ; 0.5 pt pour les commentaires
    (trie l n '() '())))                                            ; 1 pt

(define trie ; -> liste de 2 listes d'entiers                       ; 0.5 pt pour les commentaires
  (lambda (l n ld lnd) ; l, ld et lnd listes d'entier, n entier     ; 0,5 pt pour les param
    (if (null? l)                                                   ; 0.5 pt
        (list (reverse ld) (reverse lnd))                           ; 1 pt
                                                                    ; (ne pas pénaliser si listes à l'envers)
        (if (= 0 (modulo (car l) n))                                ; 0,5 pt
            (trie (cdr l) n (cons (car l) ld) lnd)                  ; 1 pt
                                                                    ; on accepte aussi (append ld (list (car l))
            (trie (cdr l) n ld (cons (car l) lnd))))))              ; 1 pt
                                                                    ; on accepte aussi (append lnd (list (car l))
```