Université Lyon 1 - Licence STS - UE LIFAP2 – 15 mai 2018 - 1h
Aucun document autorisé – Calculatrices et téléphones interdits

**Nom :** **Prénom :**

**Numéro d'étudiant :**

**C'est cette feuille qu'il faut rendre.** Ne pas l'utiliser comme brouillon.

---

**Premier exercice (2 points) - Evaluation**

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

- `(append (cons '(a b) (list 'c '(d e))) '(f (g)))` ➔ ((a b) c (d e) f (g))
- `(cadr (car (cdr '(a (b (c (d)))))))` ➔ (c (d))

**Deuxième exercice (4 points) – Fonction modification liste plate**

Définir une fonction qui, étant donnée une liste de nombres de longueur paire, forme des triplets constitués de deux nombres consécutifs de la liste et de leur addition.

`(triplet '(1 2 3 4 5 6)) -> ((1 2 **3**) (3 4 **7**) (5 6 **11**))`

```
(define triplet ; liste de sous-liste
 (lambda (l) ; liste de nombres                          ; 0.5 pts
  (if (null? l)                                          ; 0.5 pts
     l
     (cons                                               ; 0.5 pts
         (list (car l) (cadr l) (+ (car l) (cadr l)))    ; 1.5 pts
         (triplet (cddr l))))))                          ; 1 pt
```

**Troisième exercice (5 points) – Fonction liste avec un let**

Définir une fonction qui, étant donné une liste plate de nombres, compte **en un seul passage** le nombre de valeurs strictement inférieures et le nombre de valeurs supérieures à un nombre donné en paramètre.

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

```
(define infSup ; une liste de 2 nbs
 (lambda (n l) ; une liste plate de nombre et un entier   ; 0.5 pts
  (if (null? l)                                           ; 0.5 pts
     '(0 0)
     (let ((R (infSup n (cdr l))))                        ; 1.5 pts ou version descendant
       (if (< (car l) n)                                  ; 0.5 pts
          (list (+ (car R) 1) (cadr R))                   ; 1.5 pts
          (list (car R) (+ 1 (cadr R))))))))              ; 0.5 pts pour complément
```

**Quatrième exercice (5 points) – Fonction AB**

Définir une fonction qui multiplie par 10 tous les éléments d'un arbre de nombres qui n'ont qu'un seul fils.

```
(mult10 '(5(4()(2()()))(7(1()())(8(9()())()))))
     -> '(5(40()(2()()))(7(1()())(80(9()())())))
```

```
(define mult10 ; AB
 (lambda (a) ; AB de nombres                                              ; 0.5 pts
  (cond ((vide? a) a)                                                     ; 0.5 pts
      ((and (vide? (fils-g a)) (not(vide? (fils-d a))))                   ; 1 pt
        (cons-binaire (* 10 (valeur a)) (vide) (mult10 (fils-d a))))      ; 1 pt
      ((and (vide? (fils-d a)) (not(vide? (fils-g a))))                   ; 1 pt pour complément
        (cons-binaire (* 10 (valeur a)) (mult10 (fils-g a)) (vide)))
      (else  (cons-binaire (valeur a) (mult10 (fils-g a)) (mult10 (fils-d a)))   ; 1 pt
  ))))
```

**Cinquième exercice (4 points) – Fonction mystère avec profondeur**

Que fait la fonction `mystere` ? Quelles sont ses spécifications ? Donnez un exemple montrant bien les différentes caractéristiques de la fonction.

```
(define mystere
    (lambda (x)
      (cond ((null? x) x)
            ((list? (car x)) (cons (mystere (car x)) (mystere (cdr x))))
            ((symbol? (car x)) (mystere (cdr x)))
            ((number? (car x)) (cons (- (car l)) (mystere (cdr x))))
            (else (cons (car x) (mystere (cdr x))))
      )))
```

fonction qui, étant donnée une liste quelconque, parcours la liste en profondeur pour rendre la même liste dans laquelle les atomes ont été supprimés et les nombres remplacés par leur opposé.

Exemple : (mystere '(-2 (0 a 4) "c"  (r (3 f)) 5 (b)))  ⇒ (2 (0 -4) "c" ((-3)) -5 ()))

1 pt pour entrée/sortie

2 pt pour description

1 pt pour exemple avec les 4 cas