# Programmation fonctionnelle : TD2

## Exercice 1

### Sujet

On représente les ensembles d'entiers par des listes.

Définir les fonctions :

1. Ensemble vide
2. Cardinal
3. Appartenance
4. Inclusion
5. Intersection
6. Union
7. Egalité
8. Produit cartésien
9. Tous les sous-ensembles d'un ensemble

### Résolution

#### Question 1

```
(define ensemble_vide?
    (lambda(L)
        (null? L)))

(ensemble_vide? '())         ; #t
(ensemble_vide? '(2 3))      ; #f
```

#### Question 2

```
(define cardinal
    (lambda(L)
        (if (ensemble_vide? L)
        0
        (+ (cardinal (cdr L)) 1))))
```

```
(cardinal '(1 2 3))         ; 3
(cardinal '())              ; 0
```

## Question 3

```
(define appartenance?
    (lambda(x L)
        (if (ensemble_vide? L)
            #f
            (if (= (car L) x)
                #t
                (appartenance? x (cdr L))))))

(appartenance? 4 '(1 2 3))     ; #t
(appartenance? 2 '(1 2 3))     ; #t
```

## Question 4

```
(define inclusion?
    (lambda(L1 L2)
        (if (ensemble_vide? L1)
            #t
            (if (appartenance? (car L1) L2)
                (inclusion? (cdr L1) L2)
                #f))))

(inclusion? '(1 4) '(1 2 3))         ; #f
(inclusion? '(1 2) '(1 2 3))         ; #t
```

## Question 5

```
(define intersection
    (lambda(L1 L2)
        (if (ensemble_vide? L1)
            '()
            (if (appartenance? (car L1) L2)
                (cons (car L1) (intersection (cdr L1) L2))
                (intersection (cdr L1) L2)))))
```

```
(intersection '(1 4) '(1 2 3))         ; '(1)
(intersection '(1 2) '(1 2 3))         ; '(1 2)
```

## Question 6

```
(define union
    (lambda(L1 L2)
        (if (ensemble_vide? L1)
            L2
            (if (appartenance? (car L1) L2)
                (union (cdr L1) L2)
                (cons (car L1) (union (cdr L1) L2))))))

(union '(1 4) '(1 2 3))         ; '(4 1 2 3)
(union '(1 2) '(1 2 3))         ; '(1 2 3)
```

## Question 7

```
(define egalite?
    (lambda(L1 L2)
        (and (inclusion? L1 L2)
             (inclusion? L2 L1))))

(egalite? '(1 2 3) '(1 2 3))       ; #t
(egalite? '(1 2) '(1 2 3))         ; #f
```

## Question 8

### V1

```
(define produit_cartesien_helper
  (lambda(x L)
      (if (ensemble_vide? L)
        '()
        (cons (list x (car L)) (produit_cartesien_helper x (cdr L))))))

(define produit_cartesien
  (lambda(L1 L2)
    (if (ensemble_vide? L1)
      '()
      (cons (produit_cartesien_helper (car L1) L2)
```

```
            (produit_cartesien (cdr L1) L2)))))

(produit_cartesien '(1 2) '(4 5))       ; '(((1 4) (1 5)) ((2 4) (2 5)))
```

### V2

```
(define concatenation
  (lambda (L1 L2)
    (if (ensemble_vide? L1)
      L2
      (cons (car L1) (concatenation (cdr L1) L2)))))

(define f
  (lambda (T L)
    (if (ensemble_vide? L)
      '()
      (cons (list T (car L))
            (f T (cdr L))))))

(define produit_cartesien
    (lambda(L1 L2)
        (if (ensemble_vide? L1)
            '()
            (concatenation (f (car L1) L2)
                           (produit_cartesien (cdr L1) L2)))))

(produit_cartesien '(1 2) '(4 5))       ; '(((1 4) (1 5)) ((2 4) (2 5)))
```

### V3

```
(define concatenation
  (lambda (L1 L2)
    (if (ensemble_vide? L1)
      L2
      (cons (car L1) (concatenation (cdr L1) L2)))))

(define produit_cartesien
    (lambda(L1 L2)
        (if (ensemble_vide? L1)
            '()
            (concatenation (map (lambda(x) (list (car L1) x)) L2)
                           (produit_cartesien (cdr L1) L2)))))
```

```
(produit_cartesien '(1 2) '(4 5))       ; '(((1 4) (1 5)) ((2 4) (2 5)))
```

## Question 9

```
(define sous_ensemble
    (lambda(L)
        (if (null? L)
            '(())
            (let((x (sous_ensemble (cdr L))))
                (concatenation x (map (lambda(y)(cons (car L) y)) x))))))

(sous_ensemble '(1 2 3))    ; '(() (3) (2) (2 3) (1) (1 3) (1 2) (1 2 3))
```