Programmation fonctionnelle : TD4

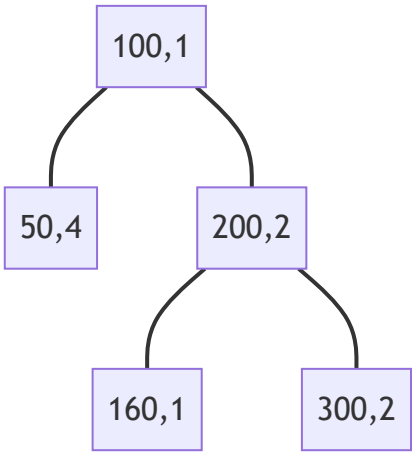
Exercice 1

Sujet

Faire les fonctions de base pour une structure de données qui prend en compte le nombre d'occurences d'un élément.

Exemple :

$$(100\ \ 50\ \ 50\ \ 50\ \ 200\ \ 200\ \ 160\ \ 300\ \ 300\ \ 50)$$



Résolution

- Représentation sous la forme `(V G D NB)`

```
#lang racket

(define estVide?
  (lambda (ND)
    (or (null? ND) zero? (cadddr ND))))

(define valeur
  (lambda (ND)
    (car ND)))

(define occurences
  (lambda (ND)
    (cadddr ND)))

(define egaux?
  (lambda (ND1 ND2)
    (if (estVide? ND1)
        (estVide? ND2)
```

```
        (if (estVide? ND2)
            #f
            (and (= (valeur ND1) (valeur ND2))
                 (= (occurences ND1) (occurences ND2))
                 (egaux? (gauche ND1)(gauche ND2))
                 (egaux? (droite ND1)(droite ND2)))))))

(define gauche
  (lambda (ND)
    (cadr ND)))

(define droite
  (lambda (ND)
    (caddr ND)))

(define developpe
    (lambda(V NB)
        (if (estVide? NB)
            '()
            (cons V (developpe V (- NB 1))))))

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

(define infixe
  (lambda (ND)
    (if (estVide? ND)
        '()
        (concatenation (infixe (gauche ND))
                       (concatenation (developpe (valeur ND) (occurence ND)
                                      (infixe (droite ND)))))))

(define prefixe
  (lambda (ND)
    (if (estVide? ND)
      '()
      (concatenation (concatenation (developpe (valeur ND) (occurences NB))
                                    (prefixe (gauche ND)))
                     (prefixe (droite ND))))))

(define postfixe
  (lambda (ND)
    (if (estVide? ND)
```

```
      '()
      (concatenation (concatenation (postfixe (gauche ND))
                                    (postfixe (droite ND))
                     (developpe (valeur ND) (occurences NB)))))))

(define nb-elements
    (lambda(ND)
        (if (estVide? ND)
            0
            (+ (nb-elements (gauche ND))
               (nb-elements (droite ND))
               (occurences ND)))))

(define construction
  (lambda (V G D NB)
    (list V G D NB)))

(define copie
  (lambda (ND)
    (if (estVide? ND)
        '()
        (construction (valeur ND)
                      (copie (gauche ND))
                      (copie (droite ND))
                      (occurences ND)))))

(define ajout
  (lambda(ND V N)
    (if (estVide? ND)
      (construction V '() '() N)
      (if (= V (valeur ND))
          (construction V (gauche ND) (droite ND) (+ N (occurences ND))))
      (if (> V (valeur ND))
          (construction (valeur ND)
                        (copie (gauche ND))
                        (ajout (droite ND) V)
                        (occurrences ND))
          (construction (valeur ND)
                        (ajout (gauche ND) V)
                        (copie (droite ND)
                        (occurences ND))))))
```