TD-TP n°4

Récursivité en profondeur, algorithmes de tri.

Exercice 1: Fonctions récursives en profondeur.

1. Écrire une fonction qui calcule le nombre d'occurences d'un élément dans une liste, incluant les occurrences dans des sous-listes.
2. Écrire une fonction qui prend une liste et ajoute 2 à chaque nombre de cette liste en profondeur.
3. Écrire une fonction calculant le nombre total d'éléments (incluant les éléments contenus dans des sous-listes) d'une liste.
4. Écrire une fonction qui, étant donnée une liste ne contenant que des nombres ou des listes de nombres, remplace en profondeur les nombres de la liste par un booléen spécifiant si le nombre correspondant est pair ou non.
5. Écrire une fonction qui calcule la profondeur d'une liste.

Exercice 2: Tri par sélection du minimum.

1. Reprendre les fonctions *minimum* et *enleve* du cours, et s'en servir pour écrire une fonction permettant de trier une liste par sélection du minimum.
2. Écrire les fonctions permettant de trier une liste par sélection du maximum.

Exercice 3: Tri à bulles.

1. Écrire la fonction *bulle* mentionnée en cours, qui sélectionne le minimum d'une liste L et vient le placer en début de liste.
2. Écrire une fonction permettant de trier une liste par tri à bulle.

Exercice 4: Tri par insertion.

1. Écrire la fonction *insere* mentionnée en cours, qui prend en paramètres un entier x et une liste L déjà triée, et insère x à la bonne place dans L.
2. Écrire une fonction permettant de trier une liste par tri par insertion.

Exercice 5. Que fait-la fonction *mystere* ci-dessous ?

```
#lang racket
(define mystere
  (lambda (L)
    (cond ((null? L) 0)
          ((and (number? (car L)) (> (car L) 0)
                (+ 1 (mystere (cdr L)))))
          ((list? (car L))
                (+ (mystere (car L)) (mystere (cdr L))))
          (else (mystere (cdr L))))))
```