TD numéro 4 : tris, récursivité profonde

À préparer avant la séance

1. Tri par insertion

Nous allons trier une liste en utilisant le tri par insertion. Le principe de ce tri consiste à trier récursivement le *cdr* de la liste, puis à insérer le *car* au bon endroit dans la liste.

• Écrire la fonction `insere` qui insère au bon endroit un nombre dans une liste de nombres triée.

```
(insere 4 '(1 2 5 7)) → '(1 2 4 5 7)
```

• Écrire la fonction `tri-insertion` qui trie une liste de nombres non vide.

```
(tri-insertion '(7 4 9 1)) → (1 4 7 9)
```

À faire pendant la séance

2. Récursivité profonde

• Écrire une fonction calculant le nombre total d'éléments d'une liste quelconque.

```
(compter '(1 2 (q w e) (r (e (w t)) 3 (e)) (o))) -> 12
```

• Écrire une fonction qui prend une liste et ajoute 2 à chaque nombre de cette liste en profondeur.

```
(ajoute2 '(2 b (10 a (56 3) 5) 4)) -> (4 b (12 a (58 5) 7) 6)
```

• Écrire une fonction qui, étant donnée une liste de nombres, remplace en profondeur les nombres de la liste par un booléen spécifiant si le nombre correspondant est pair ou non.

```
(pair? '(2 (3) 5 (6 (7)))) → (#t (#f) #f (#t (#f)))
```

3. Tri à bulles

Le tri à bulles est un tri par sélection : on sélectionne l'élément minimum de la liste à trier, on le met en première position, et l'on recommence avec le reste de la liste.

Pour extraire l'élément minimum de la liste, on le fait remonter (comme une bulle) vers le début de la liste par échanges successifs de deux éléments voisins.

• Écrire la fonction `bulle` qui fait remonter le minimum en début de liste.

• Écrire la fonction `tri-bulles`, qui trie une liste de nombres par appels successifs à la fonction `bulle`.

• Dérouler le tri à bulles sur la liste (7 9 1 6 2 3)

• Si l est une liste de longueur n, quel est le nombre de comparaisons effectuées par l'appel :

```
(tri-bulles l)
```