# Encore des exercices sur la récursivité

Exercice 1 ) Ecrire la fonction `(inverser L)` qui construit une liste dont les éléments sont les mêmes que ceux de la liste L, mais à l'envers.
Indice : regardez la fonction `(append …)` dans scheme

Exercice 2 ) Concevoir la fonction `(compter s L)` qui prend un symbole s et une liste L de symboles et qui compte les occurrences de s dans L.
Attention, les symboles peuvent être quelconques sans valeurs associées et il ne faut pas les évaluer (pour cela, on a la fonction `quote`).
Par exemple, soit la liste :

```
(define L1 '(Q U E  L E  L A V A B O  E S T  B E A U X))
(compter  'E L1)
>>>> 4
```

Exercice 3) Écrire la fonction `(begaier L)` prenant une liste de symboles (une phrase constante) en argument qui retourne une phrase où tous les mots sont répétés.
Dans la phrase, les mots sont séparés par des espaces. Par exemple :

```
(define L2 '(QUE  LE  LAVABO  EST  BEAUX))
 (begaier L2)
>>>>  (QUE QUE LE LE LAVABO LAVABO EST EST BEAUX BEAUX)
```

Exercice 4) Écrire une fonction `(corriger L)` qui ôte d'une phrase tout bégaiement. Par exemple :

```
? (corriger ((begaier L2))
>>>> (QUE  LE  LAVABO  EST  BEAUX)
```

Exercice 5) Essayer et expliquer la suite de fonctions :

```
(define op '+)
(define a 12)
(define b -3)
(define ex1 (list op a b))
(eval ex1)
```

Que fait la fonction `(eval …)` ?

Exercice 6) On peut utiliser l'expérience de l'exercice 5 pour faire des calculs. Créer la fonction `(calcul)` sans argument, qui lit une expression symétrique de l'entrée et qui exécute l'expression reçue. Exemple :

```
> (calcul)
23 + 12              ← c'est la ligne tapée...
35
>
```