TD numéro 10
map et apply

Pour chaque fonction, écrire pour réviser une version de la fonction sans utiliser map, puis une en l'utilisant.

• Écrire une fonction qui, étant donnée une liste d'atomes, construit une liste de singletons :

```
(consing '(a b 1 3 c)) → ((a) (b) (1) (3) (c))
```

• Écrire une fonction qui utilise une fonction de tri passée en argument pour trier les sous-listes d'une liste passée en argument.

```
(trilist tri-bulles '((2 3 1 4) (3 4 2) (5 4 3 7)))
          → ((1 2 3 4) (2 3 4) (3 4 5 7))
```

• Écrire une fonction qui étant donnée une liste de listes, par exemple ((a b c) (d e) (f)), renvoie ((1 a b c) (1 d e) (1 f)).

• Écrire une fonction qui étant données deux listes de symboles construit une liste de couples de la manière suivante :

```
(conscouples '(anne sophie lisa marie) '(yann pierre olivier luc))
→ ((anne yann) (sophie pierre) (lisa olivier) (marie luc))
```

• Écrire une fonction qui calcule le maximum d'une liste de nombres.

• Écrire une fonction qui calcule la profondeur d'une liste.

• Écrire une fonction qui calcule le maximum d'une liste de listes de nombres.

```
(maxlistes  '((2 1 3) (5 1 6 4) (4 3 2 1))) → 6
```

• Écrire une fonction qui calcule le maximum d'une liste en profondeur.

```
(maxprof '(1 2 (3 1 (4) 5) (2 3))) → 5
```

• Définir la spécification de la fonction mystère ci-dessous :

```
(define mystere
  (lambda (l) ; liste d'entiers
    (if (null? l)
        '(0 0)
        (let ((r (mystere (cdr l))))
          (if (even? (car l))
              (cons (+ (car l) (car r)) (cdr r))
              (list (car r) (+ (cadr r) (car l))))))))
```