TD-TP n°3

Listes, fonctions de listes, chaînes de caractères.

Exercice 1. Sans les tester sur ordinateur, devinez le résultat Racket des instructions suivantes:

- `(car (cdr (cdr '(a b c d))))`
- `(cdr '(a (b c d)))`
- `(cdr (car (cdr '(a (b c) (d e)))))`
- `(list 'a (cons '(b c) '(d)))`
- `(cons 2 (cons 3 (cons (list 4) '(3 2 4))))`
- `(append (list 'a #t 7) '(1 2 3))`
- `(append 'a '(b c) '(d))`
- `(list-ref (list 1 2 3 4 5 6) 4)`

Exercice 2. On définit deux listes $L1$ et $L2$ par

```
(define L1 '(a b c))
(define L2 '(d e))
```

Déterminer les instructions Racket permettant d'obtenir, à partir de $L1$ et $L2$, les résultats suivants:

1. ((b c) (d e))
2. (c e)
3. (b c d).

Exercice 3. Fonctions de listes

1. Écrire une fonction qui retourne le second élément d'une liste (qu'on pourra supposer contenant au moins deux éléments).
2. Écrire une fonction qui retourne vrai si et seulement si une liste n'a que deux éléments.
3. Écrire une fonction récursive qui calcule la longueur d'une liste (c'est-à-dire son nombre d'éléments).
4. Écrire une fonction qui renvoie le dernier élément d'une liste non vide.
5. Écrire une fonction qui renvoie la liste de tous les nombres d'une liste L.
6. Écrire une fonction qui supprime tous les éléments d'une liste qui sont égaux à un symbole e passé en paramètre.
7. Écrire une fonction (sans utiliser *list-ref*) qui prend en paramètres un entier n et une liste L, et qui renvoie le $n$ème élément de L. Si n est négatif ou va au delà de la liste, on renverra -1.
8. Écrire une fonction qui prend en paramètre une liste L, un entier i tel que $0 \leq i \leq$ longueur(L) et un symbole x, et qui insère l'élément x dans L après le $i$ème élément.

Exercice 4. On souhaite écrire de trois manières différentes une fonction *som-prod* qui calcule la somme et le produit des éléments d'une liste L. Par exemple, (som-prod '(1 4 2 3)) devrait renvoyer le couple (10 24).

1. Écrire une première fonction récursive sans mémorisation.
2. Écrire une version récursive avec mémorisation.
3. Écrire une fonction récursive terminale.

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

```
(define (mystere x L)
  (cond ((null? L) 0)
        ((eq? x (car L)) (+ 1 (mystere x (cdr L))))
        (else (mystere x (cdr L)))))
```