

Programmation Fonctionnelle et Symbolique : feuille 3

Listes

Exercice 3.0

Tester les fonctions de base sur les listes: `cons`, `car`, `cdr`, `list`, `append`, `reverse`,

Exercice 3.1

1. Écrire une version simplifiée de `mapcar (f l)` dont l'action est d'appliquer `f` sur chacun des éléments de `l`.
2. La tester avec $f(x) = x + 10$.

Exercice 3.2

1. Écrire une version simplifiée de `remove-if (pred l)`. Cette fonction reconstruit la liste `l` sans les éléments qui vérifient `pred`.
2. La tester avec $pred(x) \equiv (x = 0)$.

Exercice 3.3

1. Écrire une version de `make-list (n e)`. Cette fonction construit la liste qui contient `n` fois l'élément `e`.
2. Obtenir la liste `(glou glou glou glou glou)`.

Exercice 3.4

1. Écrire une version de `assoc (key l)` où `l` est une liste d'association (une liste de paires pointées). Cette fonction donne en résultat, soit la première paire dont la clef est `key`, soit ```faux` s'il n'y en a pas.
2. La tester sur la liste d'association dont chaque paire est constituée d'une chaîne de caractères et d'un symbole de fonction arithmétique associé (e.g. `(`add ' . +)`).

N.B. Vous pouvez également tenter d'écrire des versions complètes des fonctions ci-dessus en vérifiant leurs spécifications sur la documentation Lisp que vous avez en ligne.

Les exercices suivants sont tirés du livre de David S. Touretzky ```COMMON LISP: A gentle introduction to Symbolic Computation`".

Exercice 3.5

1. Écrire une fonction `swap-first-last (l)` qui "échange" le premier et le dernier élément d'une liste.
2. La tester avec la liste `' (YOU CANT BUY LOVE)`.

Exercice 3.6

1. Écrire une fonction `rotate-left(l)` qui fait une rotation circulaire vers la gauche des éléments d'une liste.
2. La tester.

Exercice 3.7

1. Écrire une fonction `rotate-right(l)` qui fait une rotation circulaire vers la droite des éléments d'une liste.
2. La tester.

Exercice 3.8

Un objet est décrit par une liste de propriétés. Les propriétés d'un couple d'objets sont regroupées dans une seule liste séparée en son milieu par le symbols `-vs-`.

```
(defparameter *prop*  
  '(large red shiny cube -vs- small shiny red four-sided pyramid))
```

1. Écrire une fonction `right-side` qui donne la liste des propriétés du deuxième objet (à droite de `-vs-`).
2. Écrire une fonction `left-side(l)` qui donne la liste des propriétés du premier objet (à gauche de `-vs-`).
3. Écrire une fonction `compare` qui retourne le nombre de propriétés communes aux deux objets sous forme d'une liste (`x PROPRIETES COMMUNES`).
Exemple: `(compare *prop*)` retourne `(2 PROPRIETES COMMUNES)`.