(Séquence 6.3

Usage de Aliste

Abbréviations

```
;;; cadr: LISTE[alpha] -> alpha
;;; ERREUR lorsque la liste donnée a moins de 2 éléments
;;; (cadr L) rend le deuxième élément de la liste donnée

;;; cddr: LISTE[alpha] -> LISTE[alpha]
;;; ERREUR lorsque la liste donnée a moins de deux éléme
;;; (cddr L) rend la liste L sans ses 2 premiers élément
```

```
(cadr L) ≡ (car (cdr L))

(cddr L) ≡ (cdr (cdr L))

(cadr '(3 5 2 5))          →          5

(cddr '(3 5 2 5))}         →          (2  5)

(cadddr '(2 3 5 6 8))      →          6

(cddddr '(2 3 5 6 8))      →          (8)
```

```
(define (mon-dico)
  '(("chat"    "tabby")
    ("souris"  "mouse")
    ("chat"    "cat")
    ("chien"   "dog")
    ("fromage" "cheese")))

(define (traduire dico)
  (define (wordfor mot)
    (let ((a (assoc mot dico)))
      (if a (cadr a) mot) ) )
  wordfor )

(map (traduire mon-dico)
     '(le chien poursuit le chat) )
   → (le dog poursuit le tabby)
```

Définition de `assoc`

`assoc` est une fonction prédéfinie en Scheme.
Elle pourrait se définir comme :

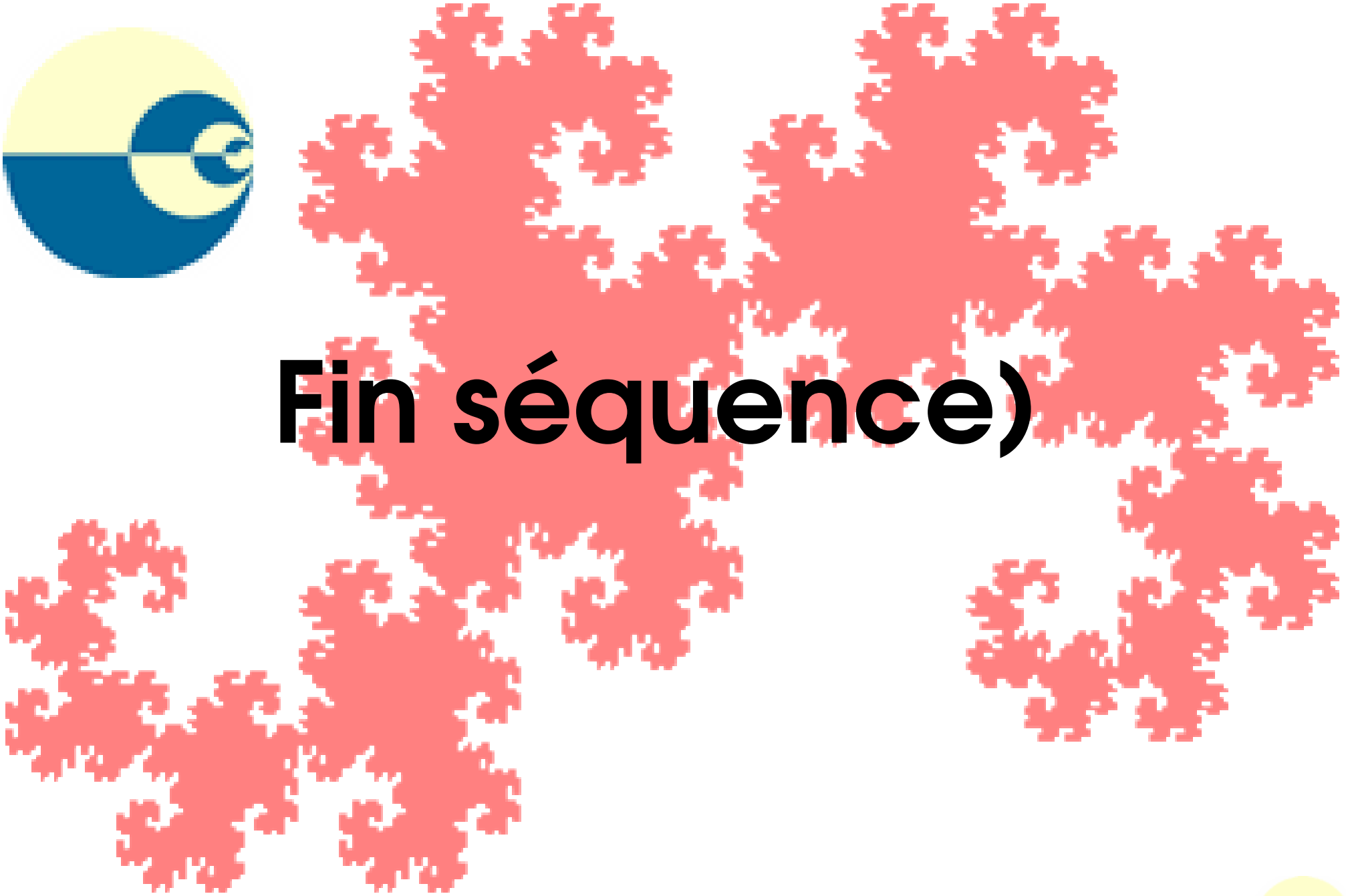
```
(define (assoc cle a-liste)
  (if (pair? a-liste)
      (if (equal? cle (caar a-liste))
          (car a-liste)
          (assoc cle (cdr a-liste)))
      #f))
```

Fonction `valeur-de`

La fonction `valeur-de` donne la valeur associée à une clé.

```
;;; valeur-de: alpha * LISTE[COUPLE[alpha beta]]
;;;                   -> beta + #f
;;; (valeur-de clef aliste) rend la valeur de la
;;; 1ère association de aliste dont le 1er élément
;;; est égal à clef. Retourne Faux en cas d'échec.
(define (valeur-de cle a-liste)
  (let ((couple (assoc cle a-liste)))
    (if couple
        (cadr couple)
        #f) ) )
```

Rappel : toute valeur différente de `#f` est équivalente à `Vrai`.



Fin séquence)