Programmation fonctionnelle

Laura Kallmeyer

Solutions des TD du 4.10.00

1 Quelques expressions en LISP

1. Valeurs des expressions suivantes:

```
> (car (cdr '((a b c) d)))
D
> (cdr (car '((a b c) d)))
(B C)
> (cddr (caaddr '((a b) (c d) ((e g h) f))))
(H)

> (cons 'a nil)
(A)
> (cons 'a ())
(A)
> (cons '((a (b))) '(c))
(((A (B))) C)

> (eq 'b (car '(b)))
T
> (eq (car '(a b)) (cadar '((c a b) d)))
T
> (eq (car '((a))) '(a))
NIL
> (eq (eq 'a 'b) (null '(a b)))
T
> (eq (eq 'a 'b) '(null (a b)))
NIL
> (null (cdr '(c)))
T
> (null (eq 'a 'b))
T
```

2. Extractions de `poire`:

```
(car '(poire pomme orange))

(caadr '(orange (poire) pomme))

(caaar (cadadr '(pomme (orange (((poire)) pomme)) citron)))

(cdr '(pomme . poire))
```

3. Valeurs des expressions suivantes:

```
> (and (null nil) (eq 'a 'a))
T
> (and (car '(a)) (null ()))
```

```
T
> (and 'a 'b 'c 'd)
D
> (or (car '(a)) (null ()))
A
> (or nil t)
T
> (eq 'a (and (car '(a))))
T
> (or (and (null nil) (cadr '(b a c))) (null (and (null '(a)))))
A
> (and (and (null nil) (cadr '(b a c))) (null (and (null '(a)))))
T
```

2 Définir une fonction

1. `(defun det (mot) (or (eq mot 'un) (eq mot 'le)))`

2. ```
   (defun length3 (une-liste)
      (and (null (cdddr une-liste)) (not (null (cddr une-liste)))))
   ```

3. ```
   (defun cdr-car-list (doublet)
     (cons (cdr doublet) (cons (car doublet) ())))
   ```

4. ```
   (defun xor (p1 p2)
     (or (and p1 (null p2))
         (and (null p1) p2)))
   ```
   ```