

INFO0054 - Programmation fonctionnelle

Répétition 1: l'interpréteur Scheme

Jean-Michel BEGON

16 Février 2017

Évaluations simples

Exercice 1.

Donner le résultat de l'évaluation des expressions suivantes :

3	c
#t	(define d #t)
(+ 1 2)	(define Robert 'Bob)
(/ 2 3)	Robert
(+ (* 3 4) 10)	(null? 'a)
(* 3 (- 12 5))	(number? 1)
(+ (+ 2 3) (+ 4 5))	(number? '1)
(- (+ 5 8) (+ 2 4))	(number? #t)
(define a 4)	(number? 'a)
a	(number? a)
(quote a)	(boolean? 3)
'a	(boolean? #t)
(define b a)	(boolean? #f)
b	(boolean? '#t)
(define a 6)	(boolean? 'd)
a	(boolean? d)
b	(symbol? 'b)
(define c (quote a))	(symbol? #f)

Les primitives cons, list, append

Exercice 2.

Donner le résultat de l'évaluation des expressions suivantes :

(car '(a b))	(list '(a) '(b))
(car (quote (a b)))	(append '(a) '(b))
(cdr '(a b))	(cons 'a (cons 'b (cons 'c '())))
(car (cdr '(a b)))	(car (cons 'a '()))
(cdr (cdr '(a b)))	(cdr (cons 'a '()))
(cons 'a '())	(cons a '(a b))
(cons 'a '(b))	(cons x '(a b))
(cons '() '())	(cadr '(a b c d))
(cons '(a) '(b))	(cadar '((a b) (c d) (e f)))

```

(list? (cons 'a 'b))
(list? (cons 'a (cons 'b '())))
(list? (cons (cons 'b '()) 'a))
(list? (cons (cons 'b '()) (cons 'b '())))
(null? (car '(a)))
(null? (cdr '(a)))
(null? '())
(null? '(a b))
(null? (car '(())))
(null? (car '((())))
(symbol? (car '(a b c)))

(symbol? (cons '() '()))
(equal? 'a (car '(a b)))
(equal? '(a b c) '(a b c))
(equal? '(a (b c)) '(a b c))
(equal? (cdr '(a c d)) (cdr '(b c d)))
(equal? '(car '((b) c)) (cdr '(a b)))
(cons (car '(a b c))
      (cons
        (car (cdr '(a b c)))
        (cons (car (cdr (cdr '(a b c)))) '())))

```

Premières formes et fonctions

Exercice 3.

Calculer en une seule forme SCHEME le nombre de secondes dans une année (non bissextile).

Exercice 4.

Écrire une forme qui rend un si x est égal à 1, ..., cinq si x est égal à 5 et inconnu sinon.

De là, définir une fonction *sayit* qui rend un si l'argument est 1, ..., cinq si l'argument est égal à 5 et inconnu sinon.

Exercice 5.

Donner le résultat de l'évaluation des expressions suivantes :

```

(lambda (y x) (cons x y))
((lambda (y x) (cons x y)) '() 'a)
(define id (lambda (x) x))
(id 1)

(id '(1 2 3))
(id id)
((id id) (id id))
(((id id) (id id)) 3)

```

Remarque : La valeur renvoyée par (define ...) n'est pas spécifiée.

Exercice 6.

La variable `phrase` est définie comme suit :

```

(define phrase
  '(((e) x (e r (c (i c e) c (o m (p l (i q))) u e))))))

```

Extraire le `m` au moyen de `car`, `cdr`, `cadr`, ...

Exercice 7.

Construire la liste (S C H E M E) au seul moyen de la liste `ls` définie comme suit :

```

(define ls '(C E H M S))

```

et des fonctions `cons`, `car`, `cdr`, `cadr`, ...

Lambdas imbriqués

Exercice 8.

Donner le résultat de l'évaluation des expressions suivantes :

```
((lambda (x) (* x x)) ((lambda (x) (* 2 x)) 6))
((lambda (x) (* x x)) (lambda (x) (* 2 x)))
((lambda (x y) ((lambda (z) (+ 12 y)) (* x x x x))) 5 2)
((lambda (x y) ((lambda (z) (* 3 z y)) (* x x))) 5 2)
((lambda (f x y) (f y x)) (lambda (u w) (+ 2 (* (+ 8 u) w))) 6 7)
((lambda (x) (lambda (y) (+ y x))) 2)
(((lambda (f) (lambda (x) (f (+ x 3)))) (lambda (y) (* 4 y))) 10)
```