INFO0054 - Programmation fonctionnelle

Répétition 1: l'interpréteur Scheme

Jean-Michel Begon

11 Février 2015

Évaluations simples

Exercice 1.
Donner le résultat de l'évaluation des expressions suivantes :

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

Les primitives `cons`, `list`, `append`

Exercice 2.
Représenter le résultat en mémoire des opérations suivantes :

```
'(2)
(cons 1 2)
(list 1 2)
'(3 4 5)
'((3 4 5))
(list '(1 2) '(3 4 5))
(cons '(1 2) '(3 4 5))
(append '(1 2) '(3 4 5))
```

Exercice 3.
Donner le résultat de l'évaluation des expressions suivantes :

```
(car '(a b))
(car (quote (a b)))
(cdr '(a b))
(car (cdr '(a b)))
(cdr (cdr '(a b)))
(cons 'a '())
(cons 'a '(b))
(cons '() '())
(cons '(a) '(b))
(list '(a) '(b))
(append '(a) '(b))
(cons 'a (cons 'b (cons 'c '())))
(car (cons 'a '()))
(cdr (cons 'a '()))
(cons a '(a b))
(cons x '(a b))
(cadr '(a b c d))
(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 4.
Calculer en une seule forme SCHEME le nombre de secondes dans une année (non bissextile).

Exercice 5.
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 6.
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 7.
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 8.
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 9.
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)
```