

# INFO0054 - Programmation fonctionnelle

## Répétition 1: l'interpréteur Scheme

Jean-Michel BEGON

05 février 2019

### Évaluations simples

---

#### Exercice 1.

Donner le résultat de l'évaluation des expressions suivantes :

3	(number? #t)
#t	(number? 'a)
(+ 1 2)	(number? a)
(/ 2 3)	(boolean? 3)
(+ (* 3 4) 10)	(boolean? #t)
(* 3 (- 12 5))	(boolean? #f)
(+ (+ 2 3) (+ 4 5))	(boolean? '#t)
(- (+ 5 8) (+ 2 4))	(boolean? 'd)
(define a 4)	(boolean? d)
a	(symbol? 'b)
'a	(symbol? #f)
(define b a)	(cons 'a '())
b	(cons 'a '(b))
(define a 6)	(car '(a b))
a	(cdr '(a b))
b	(car (cdr '(a b)))
(define d #t)	(cdr (cdr '(a b)))
(define Robert 'Bob)	(cadr '(a b c d))
Robert	(cadar '((a b) (c d) (e f)))
(null? 'a)	(null? (car '(())))
(number? 1)	(equal? '(car '((b) c)) (cdr '(a b)))
(number? '1)	

### Premières formes et fonctions

---

#### Exercice 2.

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

---

#### Exercice 3.

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 4.**

Donner le résultat de l'évaluation des expressions suivantes :

```
(lambda (y x) (cons x y))           (id 1)
((lambda (y x) (cons x y)) '() 'a)  (id '(1 2 3))
(define id (lambda (x) x))          (((id id) (id id)) 3)
```

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

---

**Exercice 5.**

Écrire une fonction `sum-int` prenant en argument un naturel  $n$  et calculant la somme des naturels inférieurs ou égaux à  $n$ .

---

**Exercice 6.**

Définir la fonction `sum-list` qui calcule la somme des éléments d'une liste de nombres.

---

**Exercice 7.**

Définir la fonction `mod` qui renvoie le modulo de deux nombres.

Remarque : la fonction *modulo* est prédéfinie, nous la redéfinissons sous un autre nom à titre d'exercice.

---

**Exercice 8.**

Écrire une fonction `big` qui prend comme arguments un nombre entier  $n$  et une liste `l` de nombres entiers, telle que `(big n l)` est la liste des éléments de `l` plus grands que  $n$ .

```
(big 5 '(7 -3 6 1 -2 5 6)) ⇒ (7 6 6)
```