```
;===========================================================================
;                                   EXO 2
;===========================================================================
(define fact
  (lambda (n)
    (if (< n 2)
        1
        (* n (fact (- n 1))))))

(define fib
  (lambda (n)
    (if (< n 2)
        n
        (+ (fib (- n 1)) (fib (- n 2))))))

;===========================================================================
;                                   EXO 3
;===========================================================================
(define somme-carrés
  (lambda (n)
    (if (zero? n)
        0
        (+ (* n n) (somme-carrés (- n 1))))))

; (somme-carrés 2)
; (somme-carrés 3)


(define somme-cubes
  (lambda (n)
    (if (zero? n)
        0
        (+ (* n n n) (somme-carrés (- n 1))))))

; (somme-cubes 3)
; (somme-cubes 5)

;;
;; Une première version de sigma
;;
(define sigma
  (lambda (f n)
    (if (zero? n)
        0
        (+ (f n) (sigma f (- n 1))))))

(define somme-carrés
  (lambda (n)
    (sigma (lambda (n) (* n n)) n)))

(define somme-cubes
  (lambda (n)
    (sigma (lambda (n) (* n n n)) n)))

;===========================================================================
;                                   EXO 4
;===========================================================================
(define delta
  (lambda (a b c)
    (- (* b b) (* 4 a c))))

(define racines
```



```
  (lambda (a b c)
    (cond ((> (delta a b c) 0)
           (list (/ (- (- b) (sqrt (delta a b c))) (* 2 a))
                 (/ (+ (- b) (sqrt (delta a b c))) (* 2 a))))
          ((= (delta a b c) 0)
           (list (/ (- b) (* 2 a))))
          (else (list)))))

; Bien sûr, cette solution n'est pas très efficace puisqu'on calcule
; plusieurs fois le DELTA. Dans une version plus réaliste, on utiliserait
; la forme LET (voir R^4RS, page 10)

; (racines 1 2 1)
; (racines 1 -1 -6)
; (racines 1 0 -1)
; (racine 1 1 1)

;===========================================================================
;                                   EXO 5
;===========================================================================
(define dérivée
  (lambda (f)
    (lambda(x dx)
      (/ (- (f (+ x dx)) (f x))
         dx))))

#|
(define carré (lambda (n) (* n n)))
(define d-carré (dérivée carré))
|#

;;
;; Deuxième version
;;
(define dérivée
  (lambda (f dx)
    (lambda(x)
      (/ (- (f (+ x dx)) (f x))
         dx))))
#|
(define carré (lambda (n) (* n n)))
(define d-carré (dérivée carré 0.001))
|#

;===========================================================================
;                                   EXO 6
;===========================================================================
(define sigma
  (lambda (f)
    (define somme (lambda (n)
                    (if (zero? n)
                        0
                        (+ (f n) (somme (- n 1))))))
    somme))

(define somme-carrés (sigma (lambda (n) (* n n))))
(define somme-cubes  (sigma (lambda (n) (* n n n))))
```