MIAS 1 et MASS 1 – Programmation fonctionnelle

TP n° 2: corrigé

```
; Exercice 1
; absolue : Reel --> Reel+
;             x  |-> -x si x<0
;                    x sinon

(define absolue
  (lambda (x)
    (if (< x 0)
(- x)
x)))


; Exercice 2
; espair? : Entier --> Booleen
;             n    |-> vrai si n est pair
;                      faux sinon

(define estpair?
  (lambda (n)
    (= 0 (remainder n 2))))


; Exercice 3
; divisiblepar? : Entier, Entier --> Booleen
;                   a   ,   b    |-> vrai si a est divisible par b
;                                    faux sinon

(define divisiblepar?
  (lambda (a b)
    (= 0 (remainder a b))))


; Ecercie 4
; bissextile? : Entier --> Booleen
;                  a   |-> vrai si a est divisible par 400
;                                ou si a est divisible par 4 mais pas par 100
;                          faux sinon

(define bissextile?
  (lambda (a)
    (or (divisiblepar? a 400)
        (and (divisiblepar? a 4)
             (not (divisiblepar? a 100))))))


; Exercice 5

(define mention1
  (lambda (x)
    (if (>= x 16)
        "TB"
        (if (>= x 14)
            "B"
            (if (>= x 12)
                "AB"
                (if (>= x 10)
                    "Passable"
                    "Recale"))))))
ps
(define mention2
  (lambda (x)
    (cond ((>= x 16) "TB")
          ((>= x 14) "AB")
          ((>= x 12) "B")
          ((>= x 10) "Passable")
          (else "Recale"))))


; Exercice 6
; attention a l'ordre des tests !

(define queltype?
  (lambda (o)
    (cond ((integer? o) "entier")
          ((real? o) "reel")
          ((boolean? o) "booleen")
          ((string? o) "chaine")
          (else "autre"))))


; Exercice 7

; a) numjours : Entier -> Entier
(define numjour
  (lambda (annee)
    (let* ((a (modulo annee 19))
           (b (modulo annee 4))
           (c (modulo annee 7))
           (d (modulo (+ (* 19 a) 24) 30))
           (e (modulo (+ (* 2 b) (* 4 c) (* 6 d) 5) 7)))
      (+ 22 d e))))

; b) traduit : Entier -> Chaine
(define traduit
  (lambda (nbj)
    (if (> nbj 31)
        (string-append (number->string (- nbj 31)) " avril ")
        (string-append (number->string nbj) " mars "))))

; c) paques : Entier --> Chaine
(define paques
  (lambda (annee)
    (traduit (numjour annee))))
```

```
; Exercice 8

(define paques2
  (lambda (annee)
    ; definition de la fonction locale numj
    (define numj
      (lambda (annee)
         (let* ((a (modulo annee 19))
                (b (modulo annee 4))
                (c (modulo annee 7))
                (d (modulo (+ (* 19 a) 24) 30))
                (e (modulo (+ (* 2 b) (* 4 c) (* 6 d) 5) 7)))
          (+ 22 d e))))

    ; definition de la fonction locale trad
    (define trad
      (lambda (nbj)
        (if (> nbj 31)
            (string-append (number->string (- nbj 31)) " avril ")
            (string-append (number->string nbj) " mars "))))

    ; corps de la fonction paques2
    (traduit (numjour annee))))


; Exercice 9

(define second-degre
  (lambda (a b c)

    ; on definit une fonction locale pour resoudre le cas
    ; ou a = 0
    (define premier-degre
      (lambda (b c)
        (if (= 0 b)
            (if (= 0 c)
                "infini"
                "aucune")
            "une")))
    ; corps de la fonction second-degree
    (if (= a 0)
        (premier-degre b c)
        ; on definit une variable locale : delta
        (let ((delta (- (* b b) (* 4 a c))))
          (cond ((< delta 0) "aucune")
                ((= 0 delta) "double")
                (else "deux"))))))
```