MIAS 1 et MASS 1 – Pogrammation fonctionnelle
TP n° 1 : corrigé

; Exercice 1 :

```
(+ 1 2)
(+ 5 (* 6 (+ 2 3)) 5)
(+ (* 2 3 5) (- 6 8))
(+ (* (- 7 3) 8) (- 12) (* 5 3))
```

; Exercice 2 :

```
(define x 2)
(define y 3)
(* x y (+ x y))
(sqrt (+ (* 4 x) (* 3 y) (/ 16 x)))
```

; Exercice 3 :

```
(define a #t)
(define b #f)
(define c #t)
(and a (or c b))
(or a (and (not c) b a))
(or (and a c) (not (or b a)))
```

; Exercice 4 :

```
; entier-suivant : Entier --> Entier
;                    n   |--> n+1

(define entier-suivant
   (lambda (n) (+ 1 n)))
```

; Exercice 5 :

```
; carre : Reel --> Reel
;           x  |-> x*x

(define carre
   (lambda (x) (*  x x)))
```

; Exercice 6 :

```
; hypo : Reel, Reel --> Reel
;          a ,   b  |-> racine carree de a^2+b^2

(define hypo
   (lambda (a b) (sqrt (+ (carre a) (carre b)))))
```

; Exercice 7 :

```
; connaissant le diametre de la roue et la distance parcourue
; le nombre de tours est egal a la distance divisee par le perimetre
; de la roue

; nbtours : Reel, Reel --> Reel
;          dist, diam |-> dist/(3.1415927 * diam)

(define nbtours
   (lambda (dist diam) (/ dist (* 3.1415927 diam))))
```

; Exercice 8 :

```
; aire-carre : Reel --> Reel
;                c  -> carre(c)

(define aire-carre
   (lambda (c) (carre c)))
```

```
; definition de la constante pi

        (define pi 3.1416)

; aire-disque : Reel --> Reel
;                r  |-> carre(r)*pi

        (define aire-disque
           (lambda (r) (* pi (carre r))))

; volume-sphere : Reel --> Reel
;                 r   |-> pi*r*r*r*4/3

        (define volume-sphere
           (lambda (r) (* (/ 4 3) pi (* r (carre r)))))
```

; Exercice 9 :

```
; Far->Celsius : Reel --> Reel
;                 f  |-> 5/9(f-32)

        (define Far->Celsius
           (lambda (f) (* 5/9 (- f 32))))

; remarque : les nombres rationnels existent en scheme
; et s'ecrivent sous forme de fraction

; Celsius->Far : Reel --> Reel
;                 c  |-> 9/5 * c + 32

        (define Celsius->Far
            (lambda (c) (+ (* 9/5 c) 32)))
```

; Exercice 10 :

```
; min->sec : Entier --> Entier
;             m    |-> 60 * m

        (define min->sec
           (lambda (m) (* 60 m)))

; h->min : cette fonction est identique a min->sec
; on peut donc ecrire :

        (define h->min min->sec)

; duree->sec : Entier, Entier, Entier --> Entier
;                h  ,   m   ,   s    |-> min->sec( h->min(h) ) + min->sec(m) + sec

        (define duree->sec
           (lambda (h m s)
              (+ (min->sec (h->min h))
                 (min->sec m)
                 sec)))

; la fonction prend 6 parametres correspondant aux deux horaires
; diff-hor : Entier, ... , Entier  --> Entier
;             h1, m1,s1,h2,m2,s2  |-> duree->sec(h2,m2,s2) - duree->sec(h1,m1,s1

        (define diff-hor
            (lambda (h1 m1 s1 h2 m2 s2)
                (- (duree->sec h2 m2 s2)
                   (duree->sec h1 m1 s1))))
```