

# Travaux Dirigés de Programmation Fonctionnelle

## «Une véritable théorie des types»

### I Règles de typage

1. Parenthéser autant que possible ces expressions de type :
  - (a) `int * float -> float`
  - (b) `int -> float * float`
  - (c) `int -> float -> float`
  - (d) `int * float * float`
2. Enlever les parenthèses inutiles dans ces expressions de type :
  - (a) `(int * float) -> float`
  - (b) `int * (float -> float)`
  - (c) `(int -> float) * float`
  - (d) `int -> (float * float)`
  - (e) `(int -> float) -> float`
  - (f) `int -> (float -> float)`
  - (g) `int * (float * float)`
  - (h) `(int * float) * float`
3. Associer correctement ces valeurs (a)-(d) à ces types (1)-(4) :
  - (a) `(0,function x -> x+.0.) ; ;`
  - (b) `function (x,y) -> if x=0 then y else 0. ; ;`
  - (c) `function x -> function y -> if x=0 then y else 0. ; ;`
  - (d) `function x -> (x 0)+.0. ; ;`
  - (1) `int * float -> float`
  - (2) `(int -> float) -> float`
  - (3) `int * (float -> float)`
  - (4) `int -> float -> float`
4. Dérouler l'algorithme d'inférence de type vu en cours (construction et résolution du système d'équations) pour trouver le type de ces fonctions :
  - (a) `let fa = function x -> function y -> x+y ; ;`
  - (b) `let fb = function x -> (x 0)+0 ; ;`
  - (c) `let fc = function x -> function y -> (y x)+x ; ;`
  - (d) `let rec fd = function x -> if x=0 then 0 else (fd (x-1)) ; ;`
5. Même question que 4. pour les fonctions suivantes :

