Expression Logique et Fonctionnelle . . . Évidemment

TD n° 2 : Ordre supérieur et polymorphisme

Exercice 1 *Sommes*

Question 1 Réalisez une fonction `somme1` paramétrée par les entiers a et b qui calcule $\sum_{k=a}^{b} k$.
Question 2 Quel est le type de l'expression `somme1 a` si `a` est un nombre entier ?
Question 3 Réalisez une fonction `somme` paramétrées par une fonction f de type `int -> int` et deux entiers a et b qui calcule $\sum_{k=a}^{b} f(k)$. Quel est le type de cette fonction ?
Question 4 Comment utiliser la fonction `somme` pour définir la fonction `somme2` qui calcule la somme des carrés des entiers de a à b, a et b étant passés en paramètres ?

Exercice 2 *Formes curryfiées et décurryfiées*

Question 1 Réalisez une fonction nommée `curryfie` qui transforme une fonction s'appliquant sur un couple, c'est-à-dire une fonction de type `'a * 'b -> 'c`, en la fonction équivalente curryfiée, de type `'a -> 'b -> 'c`.
Question 2 Réalisez la fonction réciproque.

Exercice 3 *Expressions fonctionnelles*

Donner des expressions fonctionnelles ayant les types suivants :

1. `int -> int -> int`
2. `(int -> int) -> int`
3. `int -> (int -> int)`
4. `int -> int -> int -> int`
5. `int -> (int -> int) -> int`
6. `(int -> int) -> int -> int`
7. `(int -> int -> int) -> int`

Exercice 4 *Expressions fonctionnelles polymorphes*

Donner des expressions fonctionnelles ayant les types suivants :

1. `'a -> 'a`
2. `'a -> int`
3. `'a -> 'a -> bool`
4. `('a -> 'b) -> 'a -> 'b`
5. `('a -> 'b) -> ('b -> 'c) -> 'a -> 'c`
6. `('a -> 'b) -> ('c -> 'b) -> 'a -> 'c -> bool`
7. `('a -> int -> int) -> 'a -> 'a -> int`
8. `('a -> 'b -> 'c) -> ('a -> 'b) -> 'a -> 'c`
9. `'a -> 'b`

Exercice 5 *Typage d'expressions fonctionnelles polymorphes*

Expliquez pourquoi le type des expressions fonctionnelles qui suivent ne dépend pas de l'environnement dans lequel elles sont évaluées, et déterminez ce type.

1. `fun f x y -> f x y`
2. `fun f x y -> (f x) y`
3. `fun f x y -> f (x y)`
4. `fun f x y -> f (x y f)`
5. `fun f x y -> (f x) (y f)`
6. `fun f x y -> (f x) / (f y)`
7. `fun f x y -> x f (f y)`
8. `fun f x y -> f (f x y)`

Exercice 6 *Fonctionnelles polymorphes*

Pour chacune des fonctions demandées ci-dessous, indiquez son type et donnez en une réalisation

1. Application d'une fonction f à un argument x. En nommant `applique` cette fonction, voici deux exemples d'appel à cette fonction.

```
# applique sin 0. ;;
- : float = 0.
# applique String.length "ELFE" ;;
- : int = 4
```

   Quelle est la valeur d'une application partielle de cette fonction comme par exemple `applique sin` ?

2. Composition de deux fonctions f et g. En nommant `compose` cette fonction, en voici un exemple d'utilisation

```
# let id = compose sqrt (function x -> x *. x) ;;
val id : float -> float = <fun>
# id 3. ;;
- : float = 3.
# id (-3.) ;;
- : float = 3.
```

3. Itération n fois d'une fonction f sur un argument x, ie calcul de $f^n(x) = f(f(\dots f(x)\dots))$. En nommant `itere` cette fonction, voici deux exemples d'utilisation, le premier pour caluler des approximations successives de $\sqrt{2}$, le second pour calculer les nombres de Fibonacci.

```
# let racine_de_deux n  =
  itere  (function x -> x /. 2. +. 1. /. x) n 1. ;;
val racine_de_deux : int -> float = <fun>
# List.map racine_de_deux [0; 1; 2; 3; 4; 5] ;;
- : float list =
[1.; 1.5; 1.4166666666666652; 1.41421568627450966;
 1.41421356237468987; 1.41421356237309492]
```

```
# let fibo n = snd (itere (function (x,y) -> (x+y,x)) n (1,0)) ;;
val fibo : int -> int = <fun>
# List.map fibo [0;1;2;3;4;5;6;7;8] ;;
- : int list = [0; 1; 1; 2; 3; 5; 8; 13; 21]
```

4. Itération conditionnelle d'une fonction. En nommant `tantque` cette fonction, en voici un exemple d'utilisation pour calculer le nombre de chiffres d'un entier écrit en base 10.

```
# let longueur z =
  snd (tantque (function (x,y) -> (x/10,y+1))
               (function (x,_) -> x<10)
               (z,1));;
val longueur : int -> int = <fun>
# List.map longueur [1; 123; 12345; 1234567] ;;
- : int list = [1; 3; 5; 7]
```

   Montrez comment utiliser la fonction `tantque` pour réaliser la fonction `itere`.