Programmation fonctionnelle - TD1

Licence 3

Stefano Guerrini

29 janvier 2010

1. Prévoir le résultat fourni par l'interpréteur OCaml après chacune des commandes suivantes, entrées dans l'ordre

```
# let x = 2;;
# let y = x + 1
  in let x = 3
     in x + y;;
# let x = 3
  in let y = x + 1
     in x + y;;
```

   Pourquoi les deux dernières commandes ne fournissent-elles pas le même résultat ?

2. Expliquer le comportement suivant :

```
# let x = 3;;
val x : int = 3
# let f y = y + x;;
val f : int -> int = <fun>
# f 2;;
- : int = 5
# let x = 0;;
val x : int = 0
# f 2 ;;
- : int = 5
```

3. Donner le type des expressions suivantes :
   (a) `(fun x -> x + 1) 6`
   (b) `fun x -> x + 1`
   (c) `fun x y -> x + y`
   (d) `let s = "a" in fun t -> t ^ s`
4. Donner des exemples de fonctions avec les types suivants :
   (a) `(int -> int) -> int`
   (b) `int -> (int -> int)`
   (c) `(int -> int) -> (int -> int)`
   (d) `int * int -> int`

(e) `int -> int * int`

5. Écrire la fonction `max` prenant deux argument entiers et renvoyant le maximum plus 1 et la fonction `max3` prenant trois argument et renvoyant le maximum plus 1. Écrire les mêmes fonctions pour les couples et les triples.
6. Écrire les projections pour les quadruples. Écrire une fonction qui prend une quadruple et renvoie la quadruple ordonnée de l'élément plus petit au plus grand. Écrire une fonction de quadruple que vérifie si la quadruple contient 0 et si les élément de la quadruple son du type $n, n+1, n+2, n+3$.
7. Écrire une fonction qui compose les 2 fonctions passées en argument, l'applique à `0` et ajoute `2` au résultat. Quelle est le type de la fonction ? Appliquer cette fonction aux fonctions `double` et `carre`Appliquer cette fonction aux fonctions `double` et `carre` qui retournent respectivement le double et le carré de leur argument.
8. Écrire une fonction `f` qui, à son argument `x`, renvoie la fonction qui multiplie son propre paramètre par `x`. Voici après une exemple d'utilisation et du résultat retourné :

```
#(f(3))(4);;
- : int = 12
```

9. Évaluer les expressions suivants (réécrire eux jusqu'au ce n'est plus possible de les transformer)
   (a) `let x=1+2 in ((function y -> y+x) x);;`
   (b) `let x=1+2 in ((function x -> x+x) x);;`
   (c)
```
let f1 = function f2 -> (function x -> f2 x)
in let g = function x -> x+1
   in f1 g 2;;
```

10. Écrire la fonction `fac` que calcul le factoriel d'un entier.