| Université d'Orléans | L2 Informatique |
|---|---|
| Programmation fonctionnelle | 2022-2023 |

# TD n°2

## Exercice 1

Sans l'utiliser, donner la réponse que donnerait la boucle interactive pour chacune des phrases suivantes, en supposant que celles-ci sont évaluées à la suite. Préciser à chaque fois s'il s'agit d'une expression ou d'une définition.

```
let answer = 42;;
let give = fun x -> answer;;
let answer = "Quarante-deux";;
give 10;;
give 12;;
let f = fun msg -> let answer = "Forty-Two" in msg^": "^answer;;
f "The answer is ";;
let g = fun x -> x + let x = 1 in x + let x = 2 in x;;
g(g 1);;
let swap = fun (x,y) -> (y,x);;
let l1 = [1,2,3,4];;
UOList.hd l1;;
UOList.tl l1;;
let l2 = [1;2;3;4];;
UOList.hd l2;;
UOList.tl l2;;
let k = fun lst ->
  let len = UOList.length lst in
  UOList.map((fun i -> len - i), lst);;
k l2;;
```

## Exercice 2

Donner une définition de fonction pour chacun des types suivants :

- `(int * float) -> float`
- `float -> (int * float)`
- `float -> string * int`
- `string -> char`
- `((int->int) * 'a * int) -> (int * 'a)`
- `('a * 'b * int) -> 'b * 'a`
- `'a list -> (int * 'a list)`
- `('a * 'b) list -> ('a list * 'b list)`

## Exercice 3

### Question 1

Donner le type d'une fonction `shift_left` qui prend en argument une liste et qui décale tous ses éléments vers la gauche, de façon circulaire.

Par exemple :

- `shift_left [1;2;3]` renvoie `[2;3;1]`
- `shift_left ["Bonjour";"tout";"le";"monde"]` renvoie `["tout";"le";"monde";"Bonjour"]`
- `shift_left []` renvoie `[]`

### Question 2

Sans écrire de fonction récursive, uniquement en utilisant les opérations de base sur les listes et les fonctions du module `UOList` vues dans le deuxième cours, implanter la fonction `shift_left`.

## Exercice 4

### Question 1

Donner le type, puis écrire en utilisant uniquement les fonctions du module `UOList`, une fonction `tester` qui applique toutes les fonctions d'une liste de fonctions à toutes les valeurs d'une liste de valeurs.

Par exemple, `tester([abs;fun x->x+1],[-1;0;1])` renvoie `[ [1;0;1]; [0;1;2] ]`.

### Question 2

La fonction `UOList.flatten` a le type `'a list list -> 'a list`. Cette fonction transforme une liste de listes en liste. Utiliser-la pour écrire une nouvelle version de `tester` qui renverra une liste de même type que celle de la seconde composante du couple argument de la version de `tester` de la première question.

### Question 3

Écrire une fonction `run_all_tests` qui prend en argument un couple composé d'une liste de prédicats, qui sont des fonctions de test, et d'une liste de valeurs. Si toutes les applications sur toutes les valeurs renvoient `true`, alors la fonction `run_all_tests` renvoie *"OK"*, sinon `run_all_tests` renvoie *"KO"*.