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

# TD n°1

## Exercice 1

Sans utiliser la boucle interactive, donner la réponse que donnerait la boucle interactive à chacune des phrases suivantes : le type et la valeur, et éventuellement le nom, ou expliquer pourquoi la boucle interactive indiquerait une erreur :

```
1 + 2 * 3;;
3 + 1 - 4 - 2;;
1 + 2.;;
1 +. 2.;;
1. +. 2.;;
'\n';;
"Hello " ^ " there!";;
let answer = 42;;
(0 <= answer) && (answer <= 10);;
if (0 <= answer) then "OK" else "KO";;
let result = -1.5;;
if result < 0.0 then -1 else 1.;;
fun x -> 2. *. x +. 1.;;
let f = fun x -> 2. *. x +. 1.;;
(f 2);;
(f 2.5);;
(f 0.5) +. (f 1.0);;
f(f 2.0);;
f(f(f 1.0));;
(f(f(f 1.0))) +. 1.0;;
(1, "One");;
(1, '1');;
(1, 1.0);;
let 0point = (0., 0.);;
let Origin = (0., 0.);;
let _person = ("Anne", "Nonyme", true);;
let to_string = fun (firstname, lastname, is_student) ->
  lastname ^ ", " ^ firstname ^ ": " ^
  if is_student then "étudiant" else "enseignant";;
to_string _person;;
to_string ("Jean", "Bon");;
```

## Exercice 2

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 x = 1;;
let x = x + 1;;
let y = x = 2;;
let cube = fun n -> n*n*n;;
let f = cube;;
cube;;
cube 2;;
f 3;;
let double_cube = fun n -> 2*(cube n);;
let f = fun n -> if n >=0 then n else -n;;
let f = fun n -> if if n>=0 then true else false then n else -n;;
```

## Exercice 3

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. Pour chaque fonction, expliquer ce qu'elle fait.

```
let add = fun (num1, num2) -> num1 + num2;;
let sub = fun (num1, num2) -> num1 - num2;;
add (1,2);;
sub (1,2);;
let succ = fun num -> add (num, 1);;
succ;;
succ 41;;
let pred = fun num -> sub (num, 1);;
pred;;
pred (succ 42);;
let choose_operation = fun increment -> if increment then succ else pred;;
choose_operation true;;
(choose_operation false) 1;;
(choose_operation true) 1;;
```

## Exercice 4

### Question 1

Écrire quatre fonctions, toutes de type **`float -> float`** :

1. `incr1` qui incrémente son argument de 1
2. `incr2` qui incrémente son argument de 2
3. `decr1` qui décrémente son argument de 1
4. `decr2` qui décrémente son argument de 2

### Question 2

Définir une valeur `operations`, un quadruplet dont les composantes sont les fonctions définies dans la question précédente, dans l'ordre indiqué dans la question précédente.

### Question 3

Écrire une fonction `choose` qui prend en argument un couple de valeurs composée :

- d'un quadruplet de fonctions de type **`float -> float`**,
- d'un entier

et qui renvoie une fonction de type **`float -> float`**.

Si l'entier est compris entre 0 et 3, `choose` doit renvoyer la fonction correspondante du quadruplet, sinon elle doit renvoyer la fonction identité sur les nombres à virgule flottante.

La session suivante de la boucle interactive qui suit est un exemple d'utilisation de la fonction `choose`.

```
# (choose (operations, 0)) 1.0;;
- : float = 2.
# (choose (operations, 1)) 1.0;;
- : float = 3.
# (choose (operations, 2)) 1.0;;
- : float = 0.
# (choose (operations, 3)) 1.0;;
- : float = -1.
# (choose (operations, 4)) 1.0;;
- : float = 1.
# (choose (operations, 42)) 1.0 ;;
- : float = 1.
```

```
# (choose( ((fun x->1.5*.x), (fun x->2.*.x),
            (fun x->2.5*.x), (fun x->3.*.x)), 2)) 2. ;;
- : float = 5.
```