Correction TP de Logique n°4

Licence d'informatique

Exemple de fonctions

▸ **Exercice 1.** Écrire une fonction `sum` qui calcule la somme de deux entiers

```
# let sum a b = a+b;;
val sum : int -> int -> int = <fun>
```

▸ **Exercice 2.** Écrire une fonction `incr` qui incrémente de 1 un entier en utilisant la fonction `sum`. Comparer avec `let g = sum 1`.

```
# let incr = sum 1;;
val incr : int -> int = <fun>
```

▸ **Exercice 3.** Écrire une fonction `xor` qui calcule le ou exclusif de deux booléens.

```
# let xor a b = (a || b) && not(a && b);;
 val xor : bool -> bool -> bool = <fun>
```

▸ **Exercice 4.** Écrire une fonction `fact` qui calcule la factorielle d'un entier.

```
# let rec fact = function
    0 -> 1
  | n -> n*fact(n-1);;
val fact : int -> int = <fun>
```

▸ **Exercice 5.** Écrire une fonction `comp` telle que `comp f g` soit la composée $f \circ g$ des fonctions f et g. Observer le type donné.

✂ -

```
# let comp f g x = f (g x) ;;
val comp : ('a -> 'b) -> ('c -> 'a) -> 'c -> 'b = <fun>

# let factincr = comp fact incr;;
val factincr : int -> int = <fun>

# factincr 4;;
- : int = 120
```

- ✂

▸ **Exercice 6.** Écrire la fonction `iter` qui calcule l'itérée d'une fonction. On devra avoir `iter f n x` vaut `f(f ... (f x)...)` (n fois).

✂ -

```
# let rec iter f n x =
  match n with
    0 -> x
  | _ -> iter f (n-1) (f x);;
val iter : ('a -> 'a) -> int -> 'a -> 'a = <fun>

# let mul x=2*x;;
val mul : int -> int = <fun>

#  iter mul 8 1;;
- : int = 256
```

- ✂

Système de typage

▸ **Exercice 7.** Donner des fonctions Caml qui produisent les types suivants :

1.
```
# let f1 = sum;;
```
`int -> int -> int`

2.
```
# let f2 f = incr (f 0);;
```
`(int -> int) -> int`

3.
```
# let f3 a = comp sum (sum a);;
```
`int -> int -> int -> int`

4.
```
# let f4 x f = incr (f (incr x));;
```
`int -> (int -> int) -> int`

5.
```
# let f5 f = incr (f 1 1);;
```
`(int -> int -> int) -> int`

6.
```
# let f6 f = comp incr (comp f incr);;
```
`(int -> int) -> int -> int`

▸ **Exercice 8.** Déterminer les types des fonctions suivantes, sans les évaluer sur machine :

1. `let h (f,g) = function x -> f(g x);;`

```
val h : ('a -> 'b) * ('c -> 'a) -> 'c -> 'b
```

2. `let i (a,b,c) = function x -> (a b (c * x));;`

```
val i : ('a -> int -> 'b) * 'a * int -> int -> 'b
```

3. `let j a b c = if (a>1 || c) then (b c a) else (a,c);;`

```
val j : int -> (bool -> int -> int * bool) -> bool -> int * bool
```

▸ **Exercice 9.** Ecrire une fonction `pgcd` qui calcule le plus grand diviseur commun de deux entiers. Définir ensuite une fonction `ppcm` qui calcule le plus petit commun diviseur de deux entiers. On pourra utiliser la fonction `pgcd`.

```
let rec pgcd n = function
    0 -> n
  | m when m>n -> pgcd m n
  | m -> pgcd m (n mod m);;
val pgcd : int -> int -> int = <fun>

let ppcm n m = (n*m)/(pgcd n m);;
val ppcm : int -> int -> int = <fun>
```