UNIVERSITÉ D'ORLÉANS — DEUG MIAS

Informatique — Année **2002-2003**

Feuille d'exercices OCaml n° 1

Correction

Exercice 1. Evaluer la suite d' expressions suivante :

```
let x=0;;                                          # val x : int = 0
x=0;;                                              # - : bool = true
x;;                                                # - : int = 0
0;;                                                # - : int = 0
2+2;;                                              # - : int = 4
x=1;;                                              # - : bool = false
if x=1 then 0 else 1;;                             # - : int = 1
let x=1 and y=2;;                                  # val x : int = 1     val y : int = 2
let x=y;;                                          # val x : int = 2
x=y;;                                              # - : bool = true
let z=1 and y=z;;                                  # Unbound value z
let x=y+1;;                                        # val x : int = 3
let x=4 in y=4;;                                   # - : bool = false
x;;y;;                                             # - : int = 3 ( - : int = 2 non evaluée)
let y=let x=0 in 4;;                               # val y : int = 4
let y=let x=0 and y=4;;                            # Syntax error
let y=if x=0 then 4 else 5;;                       # val y : int = 5
let y= let x=1 in if x=0 then 4 else 5;;           # val y : int = 5
x;;                                                # - : int = 3
y;;                                                # - : int = 5
let x=2 and y=x+2;;                                # val x : int = 2    val y : int = 5
let x=1 in y=x+2;;                                 # - : bool = false
x;;                                                # - : int = 2
let y= let x=1 in x+2;;                            # val y : int = 3
let x=x+1 and let y=x+4;;                          # Syntax error
x;; y;;                                            # - : int = 2 (- : int = 3 non evaluée)
let x=2 in let y=2;;                               # Syntax error
let x=2 in let y=2 in x*x+y;;                      # - : int = 6
let x=3 and y=2 in x*x+y;;                         # - : int = 11
let x=2 in let y=x+1 in x*x+y;;                    # - : int = 7
let x=4 and y=x+1 in x*x+y;;                       # - : int = 19
let x=2;;                                          # val x : int = 2
let x=x*x;;                                        # val x : int = 4
let x= let x=3 in x*x;;                            # val x : int = 9
let x=1;;                                          # val x : int = 1
let x=2 in x*4;;                                   # - : int = 8
x;;                                                # - : int = 1
let x = 5 in let x = 2 and y = x*x in 2*y*x;;      # - : int = 100
let x=1 in let x=x+1 in x+3;;                      # - : int = 5
let x=1 in let x=x+1 and y=x+2 in x+y;;            # - : int = 5
let x=1 and y=2 and z=3 in y*y-4*x*z;;             # - : int = -8
let x=1 and x=2 and x=3 in y*y-4*x*z;;             # This variable is bound several times
                                                   in this matching
```

Exercice 2. Même question que précédemment ...

```
let f=function x -> 1;;                                  # val f : 'a -> int = <fun>
let f=function x -> x;;                                  # val f : 'a -> 'a = <fun>
f;;                                                      # - : 'a -> 'a = <fun>
let f=function x -> x+1;;                                # val f : int -> int = <fun>
let f x= x+1;;                                           # val f : int -> int = <fun>
let f(x)=x+1;;                                           # val f : int -> int = <fun>
(f 1);;                                                  # - : int = 2
f 1;;                                                    # - : int = 2
f(1);;                                                   # - : int = 2
(f(f 1));;                                               # - : int = 3
(f f 1);;                                                # This function is applied to too many arguments
let f=function x -> function y -> x+y;;                  # val f : int -> int -> int = <fun>
(f 1);;                                                  # - : int -> int = <fun>
((f 1) 2);;                                              # - : int = 3
(f 1 2);;                                                # - : int = 3
f (1,2);;                                                # This expression has type int * int but
                                                         is here used with type int
f(1 2);;                                                 # This expression is not a function,
                                                         it cannot be applied
let f=function x -> (f x);;                              # val f : int -> int -> int = <fun>
(f 1);;                                                  # - : int -> int = <fun>
(f 1 2);;                                                # - : int = 3
let g=function x -> (f x);;                              # val g : int -> int -> int = <fun>
(g 1);;                                                  # - : int -> int = <fun>
(g 1 2);;                                                # - : int = 3
let h=function x -> (h x);;                              # Unbound value h
let x=3 in ((function x -> x+x) 4);;                     # - : int = 8
let x=3 in ((function y -> y+y) x);;                     # - : int = 6
let x=3 in ((function y -> x+y) 4);;                     # - : int = 7
let y=4 in let x=3 in ((function y -> x+y) y);;          # - : int = 7
let y=4 in let x=3 in ((function y -> x+y) x);;          # - : int = 6
let f= function (x-> let y=2 in x+y);;                   # Syntax error: ')' expected, the highlighted
                                                         '(' might be unmatched
let f= function x-> let y=2 in x+y;;                     # val f : int -> int = <fun>
let f= function x-> let y=2 and let x=x+y;;              # Syntax error
let x=7 in let f(y)=y*x                                  # - : int = 21
   in let x=true in if x then (f 3) else (f 2);;
let x=1+2 in ((function  y->y+x) x);;                    # - : int = 6
let x=1+2 in ((function  x->x+x) x);;                    # - : int = 6
let x = 5 in  let x = 4                                  # - : int = 16
   in let f = function x -> x*x in f x;;
```

Exercice 3. Soient f et g deux fonctions respectivement déclarées par

```
let f = function x ->  x ;;  et  let g= function h -> function x -> (h x) ;;
```

Quel est leur type ?

```
# val f : 'a -> 'a = <fun> et # val g : ('a -> 'b) -> 'a -> 'b = <fun>
```

Evaluer les expressions suivantes :

```
let h = function x -> x+1 ;;                  # val h : int -> int = <fun>
(f 1) ;;                                      # - : int = 1
(f h) ;;                                      # - : int -> int = <fun>
(f h 1) ;;                                    # - : int = 2
(g 1) ;;                                      This expression has type int but is here
                                                   used with type 'a -> 'b
(g h) ;;                                      # - : int -> int = <fun>
(g h 1) ;;                                    # - : int = 2
let g2 = function h ->
   function x -> (h x+1) in (g2 h 1) ;;       # - : int = 3
let f1 = function f2 -> (function x -> f2 x) in
   let g = function x -> x+1 in f1 g 2 ;;     # - : int = 3
```

Exercice 4. Ecrire une fonction *implique* qui pour deux booléens A et B calcule la valeur de vérité de $A \Rightarrow B$.

```
let implique a b  = if a then b else true;; (* implique : bool -> bool -> bool = <fun> *)
let implique = function a -> function b -> not a || b;; (* implique : bool -> bool -> bool = <fun>*)
let implique a = function b -> not a || b;;  (* implique : bool -> bool -> bool = <fun> *)
```

Évaluer les expressions suivantes :

```
(implique false);;                          #- : bool -> bool = <fun>
let f=(implique false) in (f true);;        #- : bool = true
implique (false);;                          #- : bool -> bool = <fun>
implique true false;;                       #- : bool = false
implique false false;;                      #- : bool = true
implique (true false);;                     <erreur>
```

Exercice 5. Ecrire une fonction *concat* qui prend trois chaines de caractères et les concatène.

```
let concat s1 s2 s3 = s1^s2^s3;;
let concat = function s1 -> function s2 -> function s3 -> s1^s2^s3;;
(* #concat : string -> string -> string -> string = <fun> *)
```

Évaluer les expressions suivantes :

```
let f = concat "toto" in                    #- : string = "toto est la"
            let g = (f " est ") in (g "la");;
let f=(concat "toto");;                     #f : string -> string -> string = <fun>
let f=(f " a ");;                           #f : string -> string = <fun>
let f=(f "faim");;                          #f : string = "toto a faim"
(f "?");;                                   This expression is not a function, it cannot be applied.
f;;                                         #- : string = "toto a faim"
(concat "toto" " plus " "faim");;           #- : string = "toto plus faim"
(((concat "ben")" si ")"finalement");;      #- : string = "ben si finalement"
```