UNIVERSITÉ D'ORLÉANS **DEUG MIAS**

Informatique Année **2001-2002**

Feuille d'exercices OCaml n° 1bis

Exercice 1. Typer les fonctions suivantes et donner la réponse de l'interprète pour les évaluations demandées :

1.
```
# let g = function (x, y, z) ->
      ((if x then y else 1), if y > 10 then "exemple" else z^"a") ;;
```
2.
```
# let f = function x -> (function y -> x + y) ;;
# let g = f 2 ;;
# g 5 ;;
# let g1 = function f -> 2*(f (function x -> x+1));;
# let h = function g -> (g 7)/3 ;;
# g1 h ;;
# let g = function f -> (function x -> f (function y -> x+y) + x) ;;
# let g = function f -> function x -> f (function y -> x+y) + x ;;
# g h 6 ;;
```

Exercice 2. Soit la définition suivante :

```
# let rec f n = if n = 0 then true else g (n-1)
   and g n = if n = 0 then false else f (n-1) ;;
```

que renvoient les interprétations suivantes :

```
# f 4 ;;
# f 5 ;;
# g 4 ;;
# g 5 ;;
```

Que font les fonctions f et g ?

Exercice 3. On souhaite écrire la fonction exponentielle de base 2 ($e : N \to N$ telle que $e(n) = 2^n$) en utilisant l'addition :

$$e(0) = 1, e(n+1) = e(n) + e(n).$$

La traduction directe en Caml donne :

```
# let rec exp n = if n = 0 then 1 else exp (n-1) + exp (n-1) ;;
# exp : int -> int = <fun>
```

Quel est le défaut majeur de cette version ? Proposer une autre version de la fonction exp pour le corriger.

Exercice 4. Typer les fonctions suivantes :

1.
```
# let d = function f -> (f 0) ;;
# d (function x -> x+1) ;;
# d (function x -> "argument"^(if x=0 then "" else "non")^" nul") ;;
```
2.
```
# let id x = x ;;
# ((id true), (id 2)) ;;
# let id x = x in (id true), (id 2) ;;
# let g f = (f true), (f 2) ;;
```

Exercice 5. Soit la déclaration suivante :

```
# let entier_it f a =
   let rec g = function 0 -> a
                  |   n -> f (g (n-1))
  in g ;;
```

Que calcule la fonction entier_it ?

Exercice 6. Donner le type des expressions suivantes :

1. `# fun f x y -> (f x) y ;;`
2. `# fun f x y -> (f x) / (f y) ;;`
3. `# fun f x y -> f x, y ;;`

4. `# fun f x y -> f (x, y) ;;`
5. `# fun f x y -> f (x y f) ;;`
6. `# fun f x y -> (f x) (y f) ;;`

Exercice 7. Quelle est la réponse de Ocaml lorsqu'on saisit l'expression

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