TDs de Logique. Feuille 5. Corrigé.

Exercice 1.—

```
#let rec inverse = function
     [] -> []
   | (x,y)::m -> (y,x)::inverse m;;
inverse : ('a * 'b) list -> ('b * 'a) list = <fun>
```

Essayer aussi :

```
#let rec map f = function
     [] -> []
  |  z::l ->(f z)::map f l;;
map : ('a -> 'b) -> 'a list -> 'b list = <fun>
#let inv = map (function (x,y) -> (y,x));;
inv : ('_a * '_b) list -> ('_b * '_a) list = <fun>
```

Exercice 2.—

```
#let rec mitose = function
   [] -> ([],[])
 | (x,y)::m ->  let (l1,l2) = mitose m in
               (x::l1,y::l2);;
mitose : ('a * 'b) list -> 'a list * 'b list = <fun>
```

ou

```
#type base = A | T | C | G ;;
Type base defined.
#let rec mitose = function
   [] -> ([],[])
 | ((x:base),(y:base))::m -> let (l1,l2) = mitose m in
                             (x::l1,y::l2);;
mitose : (base * base) list -> base list * base list = <fun>
```

Exercice 3.—

```
#let f = function
   [] -> []
 | l -> l;;
f : 'a list -> 'a list = <fun>
```

Exercice 4.—

```
#let rec f = function
   [] -> []
 | (x,y)::l -> x::y::f l;;
f : ('a * 'a) list -> 'a list = <fun>
```

Exercice 5.—

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

Exercice 6.—

```
#fun f y x -> f x y;;
- : ('a -> 'b -> 'c) -> 'b -> 'a -> 'c = <fun>
```

ou

```
#let f1 f y x = f x y;;
f1 : ('a -> 'b -> 'c) -> 'b -> 'a -> 'c = <fun>
```