Corrigé Fiche 1 (25/09/2024)

Colonne 1

Exercice 1

```
let f x = x + x

val f : int -> int = <fun>

let g t = t +. (f t) (* Ne type pas *)

Line 1, characters 18-19:
Error: This expression has type float but an expression was expected
of type
         int

let h x y z =
    let aux x = x *. x in ( aux y , x + z)

val h : int -> float -> int -> float * int = <fun>
```

Exercice 2

```
A(1,2) = A(0, A(1,1))
    A(1,1) = A(0, A(1,0))
        A(1,0) = A(0,1) = 2
           = A(0, 2) = 3
       = A(0, 3) = 4
```

Pour calculer A(1,2), on appelle 6 fois la fonction d'ackerman

```
let rec ackerman m n = match m,n with
|0,n -> n+1
|m,0 -> ackerman (m-1) n
|m,n -> ackerman (m-1) (ackerman m (n-1))

val ackerman : int -> int -> int = <fun>
```

$A(1,n)=A(0,A(1,n-1))=A(0,A(0,A(1,n-2)))....=n+2$

Par récurrence :

- Inititalisation : $A(1,0)=A(0,1)=2$
- Hérédité : Soit $A(1,n)=n+2$, montrons que $A(1,n+1)=n+3$

  $A(1,n+1)=A(0,A(1,n))=A(0,n+2)=n+3$

  Donc $A(1,n)=n+2$

Pour A(2,n), par récurrence :

- Inititalisation : $A(2,0)=A(1,1)=3$
- Hérédité : Soit $A(2,n)=2n+3$, montrons que $A(1,n+1)=2(n+1)+3$

  $$A(2,n+1)=A(1,A(2,n))=A(1,2n+3)=(2n+3)+2=2(n+1)+3$$

  Donc $A(2,n)=2n+3$

Exercice 3

```
let rec first_in l n = match l with
|[] -> false
|(h1,_)::t -> n=h1 || first_in t n

val first_in : ('a * 'b) list -> 'a -> bool = <fun>

let maxi l =
    let rec aux l m = match l with
    |[] -> m
    |h::t -> aux t (max m h)
    in match l with
    |[] -> failwith "aucun element"
    |h::t -> aux t h

val maxi : 'a list -> 'a = <fun>

let mini l =
    let rec aux l m = match l with
    |[] -> m
    |h::t -> aux t (min m h)
    in match l with
    |[] -> failwith "aucun element"
    |h::t -> aux t h

val mini : 'a list -> 'a = <fun>

let count l n =
    let rec aux l cnt = match l with
    |[] -> cnt
    |h::t -> if h= n then aux t (cnt+1) else aux t cnt
    in
    aux l 0

val count : 'a list -> 'a -> int = <fun>

let count_list l =
    let rec aux l acc = match l with
    |[] -> acc
    |h::t -> if first_in acc h then aux t acc (* element deja compté*)
             else aux t ((h, count l h)::acc)
    in aux l []
```

```
val count_list : 'a list -> ('a * int) list = <fun>
```

Colonne 2

Exercice 1

- Initialisation : 0=0
- Hérédité : $\sum_{k=0}^{n+1} k^2 = \sum_{k=0}^{n} k^2 + (n+1)^2 = \frac{n(n+1)(2n+1)}{6} + (n+1)^2$

  ¿ $\frac{(n+1)(n(2n+1)+6(n+1))}{6} = \frac{(n+1)(2n^2+7n+6)}{6}$ ¿ $\frac{(n+1)(n+2)(2n+3)}{6}$

Exercice 2

```
type expr1 =
| Entier of int
| Addition of int * int

let troisplusquatre = Addition(3,4)

type expr1 = Entier of int | Addition of int * int
val troisplusquatre : expr1 = Addition (3, 4)

type expr2 =
| Entier of int
| Addition of expr2 * expr2

type expr2 = Entier of int | Addition of expr2 * expr2

let rec eval e = match e with
|Entier(n) -> n
|Addition(e1,e2) -> (eval e1) + (eval e2)

val eval : expr2 -> int = <fun>

type expr3 =
| Entier of int
| Addition of expr3 * expr3
| Multiplication of expr3 * expr3

type expr3 =
    Entier of int
  | Addition of expr3 * expr3
  | Multiplication of expr3 * expr3

let rec eval e = match e with
|Entier(n) -> n
|Addition(e1,e2) -> (eval e1) + (eval e2)
|Multiplication(e1,e2) -> (eval e1) * (eval e2)
```

```
val eval : expr3 -> int = <fun>
```

Exercice 3

```
let pair n = (n mod 2) = 0

val pair : int -> bool = <fun>

let rec entier n = match n with
|0 -> []
|n -> n::(entier (n-1))

val entier : int -> int list = <fun>

let rec entier_pair n = match n with
|0 -> []
|n -> if pair n then n::(entier (n-1)) else entier (n-1)

val entier_pair : int -> int list = <fun>

let rec filter p l = match l with
|[] -> []
|h::t -> if p h then h::(filter p t) else filter p t

val filter : ('a -> bool) -> 'a list -> 'a list = <fun>

let entier_pair_2 n = filter pair (entier n)

val entier_pair_2 : int -> int list = <fun>
```

Colonne 3

Exercice 1

```
let f x y = x::y

val f : 'a -> 'a list -> 'a list = <fun>

let g x = x x (* Ne type pas *)

Line 1, characters 12-13:
Error: This expression has type 'a -> 'b
       but an expression was expected of type 'a
       The type variable 'a occurs inside 'a -> 'b

let h x y t = [x;y] :: t

val h : 'a -> 'a -> 'a list list -> 'a list list = <fun>
```

Exercice 2

```
let rec fact n = match n with
|0 -> 1
|n -> n * (fact (n-1))

val fact : int -> int = <fun>

let binom n k = (fact n)/((fact k)*(fact (n-k)))

val binom : int -> int -> int = <fun>
```

$\binom{n}{0} = 1$

```
let rec binom2 n k = match n,k with
|n,0 -> 1
|n,k when k>=n -> 1
|n,k -> (binom2 (n-1) (k-1)) + (binom2 (n-1) k)

val binom2 : int -> int -> int = <fun>
```

- Initialisation : $\binom{0}{0} = \binom{0}{0}$
- Hérédité : $\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k} = \binom{n}{n-(k-1)} + \binom{n}{n-k}$ ¿ $\binom{n}{n-k} + \binom{n}{n+1-k} = \binom{n+1}{n+1-k}$

Exercice 3

```
let double n = n+n

val double : int -> int = <fun>

let rec map f l = match l with
|[] -> []
|h::t -> (f h)::(map f t)

val map : ('a -> 'b) -> 'a list -> 'b list = <fun>

let doublel = map double

val doublel : int list -> int list = <fun>
```