UE - Programmation Fonctionnelle

TD 1

Exercice 1. <1.1>
```
let deuxieme l =
  match l with
  |[] -> fail ———
  |[e] -> fail ———
  |t1::t2::_ -> t2
  |_ -> failwith "liste trop courte" ;;
```
类型：'a list → 'a = <fun>

"::" 在列表前增加一个元素

"@" 把两个列表连接起来

t :: l

l1 @ l2

<1.2>
```
let n_a_zero n =
  match n with
  | 0 -> [0]                    (* [n; n-1; ...; 1; 0] *)
  | _ -> n :: n_a_zero (n-1) ;;
```
类型：int → int list = <fun>.

```
let rec zero_a_n n =
  if n<=0 then [0]
  else zero (n-1) @ [n]          (* [0; 1; ...; n-1; n] *)
```
类型：int → int list = <fun>.

List.fold_right -liste

<1.3>
```
let indice_e l e =
    let rec indice_aux liste i =
      match liste with
      |[] -> []
      |hd::tl -> if hd = e then i :: indice_aux q (i+1)
                 else indice_aux q (i+1)
```

Exercice 2 : <2.1>
```
let map f l =
    List.fold_right (fun t res_q -> (f t) :: res_q) l []
```

<2.2>
```
let rec flatten l =    (* aplatissement d'une liste de listes *)
    List.fold_right (fun t q -> t @ q) l []
```

<2.3>
```
let fst l =
    List.fold_right (fun (x,y) res_q -> x :: res_q) l []
    List.map (fun (x,y) -> x) l.
```

<2.4> let split t =
List.fold_right (fun (st, nd) (split1, split2) ->
(st :: split1, nd :: split2)) t ([], [])