Correction du DS 1

Option informatique, première année

Julien Reichert

```
(* exo1 : unit -> int array *)
let exo1 () =
  let tab = Array.make 42 0 in
    for i = 0 to 41 do (* on peut démarrer à 1 vu qu'il n'y a pas d'intérêt au premier tour *)
      tab.(i) <- i
    done; tab;;
(* plus court : let exo1 () = Array.init 42 (fun i -> i);; *)

(* echange_ref : 'a ref -> 'a ref -> unit *)
let echange_ref a b =
  let c = !a in (* ou ref !a, surtout pas a *)
  a := !b;
  b := c;; (* !c si c est ref !a *)

(* occurrences : 'a -> 'a array -> int list et de même pour occurrences_rec *)
let occurrences elt tab =
  let rep = ref [] in
    for i = Array.length tab - 1 downto 0 do
      if tab.(i) = elt then rep := i::(!rep)
    done; !rep;;

let occurrences_rec elt tab =
  let rec aux accu i =
    if i = -1 then accu (* parcours en sens inverse pour avoir une liste croissante *)
    else aux (if tab.(i) = elt then i::accu else accu) (i-1) (* façon exotique d'écrire *)
  in aux [] (Array.length tab - 1);;

let euclide a b =
  let aa = ref (max (abs a) (abs b)) and bb = ref (min (abs a) (abs b)) in
    if !aa = 0 then raise Division_by_zero; (* a et b étaient nuls *)
    (* pas besoin de else, vu qu'on a déclenché une erreur ici, cela allège *)
    while !bb <> 0 do
      let c = !aa mod !bb in (* compenser l'absence d'affectations simultanées *)
      aa := !bb; bb := c
    done; !aa;;

let rec euclide_rec a b =
  if a = 0 && b = 0 then raise Division_by_zero
  else if a < 0 then euclide_rec (-a) b
  else if b < 0 then euclide_rec a (-b)
  else if a < b then euclide_rec b a
  else if b = 0 then a
  else euclide_rec b (a mod b);;
```

```
(* croissante : 'a list -> bool *)
let rec croissante l = match l with
| [] -> true (* cas résiduel jamais atteint si l n'est pas d'emblée vide *)
| [_] -> true
| a::b::q -> if a > b then false else croissante (b::q);; (* ne pas oublier b, ni la parenthèse *)
(* Attention, le if + booléen, d'habitude honni, sert à rendre la fonction récursive terminale
sans se servir d'un buffer. Il est tout aussi acceptable d'écrire (a <= b) && croissante (b::q). *)
```