Correction du DS 1

Option informatique, première année

Julien Reichert

Question 1 : Pour un tableau, `t.(0)` ; pour une chaîne, `s.[0]` ; pour une liste, `List.hd l` ou par un filtrage.

Question 2 : On utilise `&&`.

Question 3 : Pour un tableau, `t.(i) <- x` si `i` est entre `0` (inclus) et la taille de `t` (exclue) et si `x` est du type des éléments de `t` ; pour une chaîne de caractère, c'est désormais impossible, mais on peut admettre l'ancienne version `s.[i] <- x` aux mêmes conditions (donc `x` doit être un caractère) ; pour une liste c'est impossible, et pas seulement parce que l'accès direct à un élément est interdit ; pour une référence, on écrit `r := x` sous réserve là aussi que le type soit compatible.

Question 4 : `print_string "Bonjour"; print_int n;;` (sans les deux points-virgules finaux quand le code est à insérer dans une expression)

Question 5 : Le mot-clé `let` introduit une ou plusieurs (re)création(s) de valeur, qui est(sont) fermée(s) par deux points-virgules si la création est sur toute l'expression, sinon par `in`, ce qui est le cas dans les boucles.

Question 6 : Le balisage des boucles est `do ... done`.

```
(* exo1 : 'a -> 'a -> 'a -> 'a *)
let exo1 a b c =
  if a > b then if a > c then a else c
  else if b > c then b else c;;
(* Autorisé aussi : max a (max b c) et variantes *)

(* occurrences : char -> string -> int list et de même pour occurrences_rec *)
let occurrences car s =
  let rep = ref [] in
    for i = String.length s - 1 downto 0 do
      if s.[i] = car then rep := i::(!rep)
    done; !rep;;

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

(* compte_tab : 'a array -> 'a -> int *)
let compte_tab tab elt = let rep = ref 0 in
  for i = 0 to Array.length tab - 1 do
    if tab.(i) = elt then incr rep
  done; !rep;;
```

```
(* compte_liste : 'a list -> 'a -> int *)
let rec compte_liste l elt = match l with
| [] -> 0
| a::q -> if a = elt then 1 + compte_liste q else compte_liste q;;
```