Correction du DS 2

Option informatique, première année

Julien Reichert

Pour une version utilisable sur l'ordinateur, voir le fichier ML associé.

```
(* Exercice 1 *)

let moyl l =
  let rec produitettaille l accu n = match l with
    | [] -> accu, n
    | a::q -> produitettaille q (a *. accu) (n + 1)
  in let p, t = produitettaille l 1. 0 in p ** (1. /. (float_of_int t));;
(* Erreur de division par zéro si liste vide. *)

(* Exercice 2 *)

let moyt t =
  let p = ref 1. and n = Array.length t in
    for i = 0 to n-1 do p := !p *. t.(i) done; !p ** (1. /. (float_of_int n));;
(* Idem *)

(* Exercice 3 *)

let rec map f l = match l with
  | [] -> []
  | a::q -> (f a)::(map f q);;

(* Exercice 4 *)

let multiplication a b =
  let rec multaux accu x y =
    if y = 0 then accu
    else multaux (if y mod 2 = 0 then accu else accu + x) (2*x) (y/2)
  in multaux 0 a b;;
```

```
(* Exercice 5 *)

(* On commence par une fonction auxiliaire.
Dans les deux fonctions, on considère que les tailles des n matrices sont données
dans un tableau de taille n+1 sans information redondante,
donc pour l'exemple de l'énoncé on aurait le tableau [|4; 6; 2; 10; 3|]. *)

let maj tab_tailles mat_nombres debut separation fin =
  let nb_mult = tab_tailles.(debut) * tab_tailles.(separation+1) * tab_tailles.(fin+1) in
    let total_mult = mat_nombres.(debut).(separation) + nb_mult + mat_nombres.(separation+1).(fin) in
  if mat_nombres.(debut).(fin) = -1 || mat_nombres.(debut).(fin) > total_mult
    then mat_nombres.(debut).(fin) <- total_mult;;

let nombre_multiplications tab_tailles =
  let n = (Array.length tab_tailles) - 1 in
    let mat_nombres = Array.make_matrix n n (-1) in
      for i = 0 to n-1 do mat_nombres.(i).(i) <- 0 done;
      for taille = 1 to n-1 do
        for i = 0 to n-1-taille do
          for j = i to i+taille-1 do
            maj tab_tailles mat_nombres i j (i+taille)
          done
        done
      done; mat_nombres.(0).(n-1);;

(* Exercice 6 *)

let creer_pile_oublieuse c = Array.make (c+1) 0;; (* c est la capacité *)

let est_vide p = p.(0) = 0;;

let empiler p e =
    if p.(0) = Array.length p - 1
    then for i = 2 to p.(0) do p.(i-1) <- p.(i) done
    else p.(0) <- p.(0) + 1; p.(p.(0)) <- e;;

let depiler p =
    if p.(0) = 0 then invalid_arg "Pile vide"
    else p.(0) <- p.(0) - 1; p.(p.(0) + 1);;

(* Exercice 7 *)

let creer_pile_oublieuse2 c = let pile = Array.make (c+2) 0 in pile.(1) <- 1; pile;;
(* c est la capacité, premier élément taille, deuxième élément position de la tête,
donc même fonction est_vide *)

let empiler2 p e =
    p.(0) <- min (p.(0) + 1) (Array.length p - 2);
    p.(1) <- p.(1) + 1;
    if p.(1) = Array.length p then p.(1) <- 2; p.(p.(1)) <- e;;

let depiler p =
    if p.(0) = 0 then invalid_arg "Pile vide"
    else p.(0) <- p.(0) - 1;
    p.(1) <- p.(1) - 1;
    if p.(1) = 1 then p.(1) <- Array.length p - 1; p.(p.(1));;
```