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 sommeettaille l accu n = match l with
    | [] -> accu, n
    | a::q -> sommeettaille q (a +. accu) (n + 1)
  in let s, t = sommeettaille l 0. 0 in s /. (float_of_int t);;

(* Exercice 2 *)

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

(* Exercice 3 *)

let rec forall p l = match l with
  | [] -> true
  | a::q -> if p a then forall p q else false;;

(* Exercice 4 *)

let lignes n =
  if n = 0 then 1 else if n = 1 then 0 else if n = 2 then 1 else
  let reponse = Array.make (n+1) (-1) in
    reponse.(0) <- 1; reponse.(1) <- 0; reponse.(2) <- 1;
    let rec remplit indice = (* variante de mémoïzation avec effet de bord uniquement *)
      if reponse.(indice) = -1 then (* donc indice >= 3 à ce stade *)
        ( remplit (indice-2); remplit (indice-3);
        reponse.(indice) <- reponse.(indice-2) + reponse.(indice-3) )
      in remplit n; reponse.(n);;

(* Exercice 5 *)

let min_acceptable_le_retour lind t =
  if lind = [] then failwith "Aucun indice acceptable";
  let rec mini accu l = match l with
  | [] -> accu
  | a::q -> mini (min t.(a) accu) q
  in mini t.(List.hd lind) (List.tl lind);;
(* On ne vérifiera pas que les éléments de lind sont des indices existants. *)
```

```
(* Exercice 6 *)

type 'a arbrebin = Vide | Noeud of 'a arbrebin * 'a * 'a arbrebin;;
type 'a arbre = V | R of 'a * ('a arbre list);;

let rec taille_arbre_bin a = match a with
  | Vide -> 0
  | Noeud(g,_,d) -> 1 + taille_arbre_bin g + taille_arbre_bin d;;

(* Exercice 7 *)

let rec somme_arbre_bin a = match a with
  | Vide -> 0
  | Noeud(g,n,d) -> somme_arbre_bin g + n + somme_arbre_bin d;;

(* Exercice 8 *)

let rec taille_arbre a = match a with
  | V -> 0
  | R(_, fils) -> let rec somme_tailles l = match l with
      | [] -> 1
      | a::q -> taille_arbre a + somme_tailles q
      in somme_tailles fils;;

(* Exercice 9 *)

type fraction = {num : int ; den : int};;

let rec pgcd a b =
  if a = 0 && b = 0 then failwith "PGCD de 0 et 0";
  if b = 0 then a else pgcd b (a mod b);;

let simplifie fraction =
  if fraction.den = 0 then raise Division_by_zero;
  let p = pgcd fraction.num fraction.den in
    let signe = (if fraction.den / p > 0 then 1 else -1) in
      { num = fraction.num * signe / p ; den = fraction.den * signe / p};;

let addition f1 f2 =
  let n1 = f1.num and n2 = f2.num and d1 = f1.den and d2 = f2.den in
    simplifie { num = n1 * d2 + d1 * n2; den = d1 * d2};;

let soustraction f1 f2 =
  let n1 = f1.num and n2 = f2.num and d1 = f1.den and d2 = f2.den in
    simplifie { num = n1 * d2 - d1 * n2; den = d1 * d2};;

let multiplication f1 f2 =
  let n1 = f1.num and n2 = f2.num and d1 = f1.den and d2 = f2.den in
    simplifie { num = n1 * n2; den = d1 * d2};;

let division f1 f2 =
  let n1 = f1.num and n2 = f2.num and d1 = f1.den and d2 = f2.den in
    if d2 = 0 then raise Division_by_zero (* si n2 = 0, ce sera fait par simplifie *)
    else simplifie { num = n1 * d2; den = d1 * n2};;
```

```
(* Exercice 10 *)

let echange_lignes m i j =
  for k = 0 to Array.length m.(i) - 1 do
    let buff = m.(i).(k) in m.(i).(k) <- m.(j).(k); m.(j).(k) <- buff
  done;;

let divise_ligne m i lambda =
  for k = 0 to Array.length m.(i) - 1 do
    m.(i).(k) <- division m.(i).(k) lambda
  done;;

let soustrait_ligne m i j lambda =
  for k = 0 to Array.length m.(i) - 1 do
    m.(j).(k) <- soustraction m.(j).(k) (multiplication lambda m.(i).(k))
  done;;

(* Pas besoin du prétraitement du TP 8 d'IPT SUP en raison du typage fort de Caml *)

let pivot mat =
  let n = Array.length mat in
  for i = 0 to n-1 do
    if mat.(i).(i).num = 0 then
      (
      let j = ref (i+1) in
        while !j < n && mat.(!j).(i).num == 0 do incr j done;
        if !j = n then failwith "Matrice non inversible";
        echange_lignes mat i !j
      );
    divise_ligne mat i mat.(i).(i);
    for j = 0 to n-1 do
      if j <> i then soustrait_ligne mat i j mat.(j).(i)
    done
  done;;
```