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 nombre_de_lettres n =
  let rec aux k vspk somme = (* vspk : _v_ingt-_s_ix _p_uissance _k_ *)
    if somme+vspk >= n then (k, somme)
    else aux (k+1) (26*vspk) (somme + vspk)
  in aux 1 26 0;;

let clastre_nom n =
  let k, superieurs = nombre_de_lettres n in
  let reste = ref (n - superieurs - 1) in
  let nom_nombres_tab = Array.make k 0 in
    for i = k-1 downto 0 do
      nom_nombres_tab.(i) <- !reste mod 26;
      reste := !reste / 26
    done;
  String.init k (fun i -> char_of_int (nom_nombres_tab.(i) + 65));;

let sommepuiss vs k = (* somme des vs puissance i pour i de 1 à k inclus *)
  let rec aux i vspk somme =
    if i = k then somme
    else aux (i+1) (vspk*vs) (somme + vspk)
  in aux 0 vs 0;;

let clastre_rang nom =
  let k = String.length nom in
  let superieurs = ref (sommepuiss 26 (k-1))
  and vspi = ref 1
  and indice = ref (k-1) in
  while !indice >= 0 do
    superieurs := !superieurs + !vspi * (int_of_char nom.[!indice] - 65);
    vspi := !vspi * 26;
    decr indice
  done; !superieurs + 1;;

(* Autre version, grâce à la perspicacité de Tom : *)

let clastre_rang_facile nom =
  let rang = ref 0 and vspi = ref 1 in
    for i = String.length nom - 1 downto 0 do
      rang := !rang + (int_of_char nom.[i] - 64) !vspi;
      vspi := !vspi * 26
    done;
  !rang;;
```

```
(* Astuce : A compte pour 1 au lieu de 0, de sorte qu'il y ait une augmentation de 26 puissance i
pour chaque indice, correspondant à la somme du nombre de personnes ayant strictement moins de lettres,
dont le 26 puissance 0 car le meilleur a pour rang 1 et non 0 *)


(* Question 2.1 *)

let distance depart arrivee =
  let m = String.length depart and n = String.length arrivee in
  let matrice = Array.make_matrix (m+1) (n+1) (-1) in
    for i = 0 to n do matrice.(0).(i) <- i done;
    for i = 0 to m do matrice.(i).(0) <- i done;
    for i = 1 to m do
      for j = 1 to n do
        matrice.(i).(j) <- min (1 + min matrice.(i-1).(j) matrice.(i).(j-1))
        (matrice.(i-1).(j-1) + (if depart.[i-1] = arrivee.[j-1] then 0 else 1))
      done
    done; matrice.(m).(n);;

(* Question 2.2 *)

(* Insérer au début : *)
  let (insertion, suppression, remplacement) = couts in
(* Remplacer à l'endroit pertinent *)
    for i = 0 to n do matrice.(0).(i) <- i*suppression done;
    for i = 0 to m do matrice.(i).(0) <- i*insertion done;
(* Remplacer aussi la formule *)
        matrice.(i).(j) <- min (min (matrice.(i-1).(j)+insertion) (matrice.(i).(j-1)+suppression))
        (matrice.(i-1).(j-1) + (if depart.[i-1] = arrivee.[j-1] then 0 else remplacement))

(* Exercice 3 *)

type ('a, 'b) table_hachage = { fonction : 'a -> int; donnees : ('a * 'b) list array};;

let cree f taille = { fonction = f; donnees = Array.make taille [] };;

let ajoute (cle, valeur) table =
  let indice = table.fonction cle in
  if List.mem cle (List.map fst table.donnees.(indice)) then failwith "Clé déjà présente";
  table.donnees.(indice) <- (cle, valeur)::table.donnees.(indice);;

let modifie cle nv_valeur table =
  let indice = table.fonction cle in
  let rec remplacement l = match l with
  | [] -> failwith "Clé absente";
  | (c, v)::q -> if c=cle then (c, nv_valeur)::q else (c, v)::(remplacement q)
  in table.donnees.(indice) <- remplacement table.donnees.(indice);;

let supprime cle table =
  let indice = table.fonction cle in
  let rec suppression l = match l with
  | [] -> failwith "Clé absente";
  | (c, v)::q -> if c=cle then q else (c, v)::(suppression q)
  in table.donnees.(indice) <- suppression table.donnees.(indice);;
```

```
let consulte cle table =
  let indice = table.fonction cle in
  let rec consultation l = match l with
  | [] -> failwith "Clé absente";
  | (c, v)::q -> if c=cle then v else consultation q
  in consultation table.donnees.(indice);;
```