```
(* TD N°1 : Tables de hachage *)

(************************* Question N°2 *************************)

(* en itératif, le plus simple avec les chaînes de caractères et les tableaux *)
let hash m s =
  let r = ref 0 in
  for i = 0 to string_length s - 1 do
    r:= (128 * !r + int_of_char s.[i]) mod m
  done;
  !r
;;

(* en récursif *)
let hash m s =
  let rec aux res = function
    | -1 -> res
    | i ->
        let c = int_of_char (s.[i]) in
        aux ((c + 128 * res) mod m) (pred i)
  in aux 0 (string_length s - 1)
;;
(* note : pred i -> i - 1 *)

(************************* Question N°3 *************************)

let nouvelle m = make_vect m [];;

(************************* Question N°4 *************************)
(* la taille de la table de hachage est la taille du tableau *)
let ajoute t s =
  let i = hash (vect_length t) s in
  if not (mem s t.(i)) then
    t.(i) <- s :: t.(i)
;;

(************************* Question N°5 *************************)

let trouve t s =
  let i = hash (vect_length t) s in
  mem s t.(i)
;;

(* mem e l renvoie true si e est un élément de la liste l *)

(************************* Question N°6 *************************)

type 'a tableh = { max              : int ;
                   mutable donnees : 'a list vect } ;;

let nouvelle m t =
  { max = m;
    donnees = make_vect t [] }
;;

let redim t =
  let m = vect_length t.donnees in
  let new_t = make_vect (2 * m + 1) [] in
  let ajoute t s =
    let i = hash (vect_length t) s in
    if not (mem s t.(i)) then
      t.(i) <- s :: t.(i)
```



```
  in
  for i = 0 to m - 1 do
    do_list (ajoute new_t) (rev t.donnees.(i))
  done;
  t.donnees <- new_t
;;

let ajoute t s =
  let i = hash (vect_length t.donnees) s in
  if not (mem s t.donnees.(i)) then
    t.donnees.(i) <- s :: t.donnees.(i);
  if list_length t.donnees.(i) > t.max then
    redim t
;;
```