```
(* question 2 *)

let somme a b =
  let n = vect_length a in
  let c = make_matrix n n true in
  for i = 0 to n-1 do
    for j = 0 to n-1 do
      c.(i).(j) <- a.(i).(j) || b.(i).(j)
    done
  done ;
  c ;;


let produit a b =
  let n = vect_length a in
  let d = make_matrix n n true in
  let rec aux i j = function
    | k when k = n -> false
    | k            -> a.(i).(k) && b.(k).(j) || aux i j (k+1)
  in
  for i = 0 to n-1 do
    for j = 0 to n-1 do
      d.(i).(j) <- aux i j 0
    done
  done ;
  d ;;


(* question 3 *)

(* on calcule (I+A)^n par exponentiation rapide *)

let accessible a =
  let n = vect_length a in
  let id = make_matrix n n false in
  for i = 0 to n-1 do
    id.(i).(i) <- true
  done ;
  let b = somme a id in
  let rec aux = function
    | 1 -> b
    | k when k mod 2 = 0 -> let d = aux (k/2) in produit d d
    | k -> let d = aux (k/2) in produit b (produit d d)
  in aux n ;;


(* question 4 *)

let rec affiche_chemin = function
  | []   -> ()
  | [t]  -> print_int t ; print_newline ()
  | t::q -> print_int t ; print_string " -> " ; affiche_chemin q ;;

let chemins a i j =
  let n = vect_length a in
  let rec aux chemins exclus i = function
    | _ when i = j                          -> [[j]]
    | k when k = n                          -> chemins
    | k when a.(i).(k) && not mem k exclus -> let lst = map (function l ->
i::l) (aux [] (i::exclus) k 0)
```

```
                                                                    in aux (lst
@ chemins) exclus i (k+1)
    | k                                       -> aux chemins exclus i (k+1)
  in do_list affiche_chemin (aux [] [] i 0) ;;


(* tests *)

let a = [| [| false; true; false; true; false; true |] ;
           [| false; false; true; true; true; false |] ;
           [| false; true; false; true; false; false |] ;
           [| false; false; false; false; true; false |] ;
           [| false; false; false; false; false; false |] ;
           [| false; true; false; false; false; false |] |] ;;

accessible a ;;

chemins a 0 4 ;;
```