```
(* Vahe Karamian - CS 408 *)


open List;;

let rec interval n m =
 if n > m then [] else n :: interval (n + 1) m;;

let filter_append p l l0 =
  let rec filter = function
    | [] -> l0
    | h :: t -> if p h then h :: filter t else filter t in
   filter l;;

let rec concmap f = function
  | [] -> []
  | h :: t -> f h (concmap f t);;

let rec safe x d  = function
  | [] -> true
  | h :: t ->
     x <> h && x <> h + d && x <> h - d && safe x (d + 1) t;;

let rec ok = function
  | [] -> true
  | h :: t -> safe h 1 t;;

let find_solutions size =
 let line = interval 1 size in
 let rec gen n size =
   if n = 0 then [[]] else
   concmap
    (fun b -> filter_append ok (map (fun q -> q :: b) line))
    (gen (n - 1) size) in
 gen size size;;

let print_solutions size =
 let sol_num = ref 1 in
 iter
   (fun chess ->
     Printf.printf "\nSolution number %i\n" !sol_num;
     sol_num := !sol_num + 1;
     iter
       (fun line ->
         let count = ref 1 in
         while !count <= size do
           if !count = line then print_string "Q " else print_string "- ";
           count := !count + 1
         done;
         print_newline())
       chess)
   (find_solutions size);;

let queens () =
 let size =
   print_string "Chess size ? "; flush stdout; read_int () in
 print_solutions size;;

if !Sys.interactive then () else queens ();;
```