```
(* Q1 *)
type catégorie_permis = A | B | C | D | E;;
(* Le type catégorie_permis est défini. *)

type ville = { nom_ville: string; code_postal: int };;
(* Le type ville est défini *)

type catégorie_voie = Rue | Avenue | Boulevard | Allée | Impasse ;;
(* Le type catégorie_voie est défini *)

type voie = { cat: catégorie_voie; nom_voie: string };;
(* Le type voie est défini *)

type adresse = { numéro_adresse: int; rue: voie; commune: ville };;
(* Le type adresse est défini *)

type identité = { nom: string; prénom: string; domicile: adresse };;
(* Le type identité est défini *)

type permis = { personne: identité; catégories: catégorie_permis list };;
(* Le type permis est défini *)

type date = { jour: int; mois: int; année: int };;
(* Le type date est défini *)

type immatriculation = { chiffres: int; lettres: string; département: int };;
(* Le type immatriculation est défini *)

type carte_grise =
{ département_carte_grise: int;
  date_première_mise_en_circulation: date;
  numéro_immatriculation: immatriculation;
  date_immatriculation: date;
  propriétaire: identité };;
(* Le type carte_grise est défini *)

type vignette =
{ département_vignette: int;
  année_vignette: int;
  immatriculation_vignette: immatriculation };;
(* Le type vignette est défini *)

(* Q2 *)
let permis_voiture p = mem B p.catégories;;
(* permis_voiture : permis -> bool = <fun> *)

let nouveau_permis p c =
  { personne = p.personne; catégories = c::p.catégories };;
(* nouveau_permis : permis -> catégorie_permis -> permis = <fun> *)

(* Q3 *)
let département_of_adresse adr = adr.commune.code_postal / 1000;;
(* département_of_adresse : adresse -> int = <fun> *)

(* Q4 *)
let achat_véhicule_occasion cg d ident =
  { département_carte_grise = département_of_adresse ident.domicile;
    date_première_mise_en_circulation = cg.date_première_mise_en_circulation;
    numéro_immatriculation = cg.numéro_immatriculation;
    date_immatriculation = d;
    propriétaire = ident };;
(* achat_véhicule_occasion : *)
```



```
(*   carte_grise -> date -> identité -> carte_grise = <fun> *)

(* Q5 *)
let changement_adresse ident nv_adr =
  { nom = ident.nom; prénom = ident.prénom; domicile = nv_adr };;
(* changement_adresse : identité -> adresse -> identité = <fun> *)

let changement_domicile cg d nv_adr nv_imm =
  { département_carte_grise = département_of_adresse nv_adr;
    date_première_mise_en_circulation = cg.date_première_mise_en_circulation;
    numéro_immatriculation = nv_imm;
    date_immatriculation = d;
    propriétaire = changement_adresse cg.propriétaire nv_adr };;
(* changement_domicile : *)
(*  carte_grise -> date -> adresse -> immatriculation -> carte_grise = <fun> *)

(* Q6 *)
let cohérence_département_carte_grise cg =
  cg.département_carte_grise = cg.numéro_immatriculation.département &&
  cg.département_carte_grise = département_of_adresse cg.propriétaire.domicile;;
(* cohérence_département_carte_grise : carte_grise -> bool = <fun> *)

(* Q7 *)
let avant d1 d2 =
  d1.année < d2.année ||
  d1.année = d2.année && d1.mois < d2.mois ||
  d1.année = d2.année && d1.mois = d2.mois && d1.jour <= d2.jour;;
(* avant : date -> date -> bool = <fun> *)

let cohérence_dates_carte_grise cg =
  avant cg.date_première_mise_en_circulation cg.date_immatriculation;;
(* cohérence_dates_carte_grise : carte_grise -> bool = <fun> *)

(* Q8 *)
let rec identification imm cgl = match cgl with
   | [] -> failwith "identification: immatriculation inconnue"
   | cg::cgl' -> if cg.numéro_immatriculation = imm then cg.propriétaire
                 else identification imm cgl';;
(* identification : immatriculation -> carte_grise list -> identité = <fun> *)

(* Q9 *)
let achat_vignette cg a =
  { département_vignette = cg.département_carte_grise;
    année_vignette = a;
    immatriculation_vignette = cg.numéro_immatriculation };;
(* achat_vignette : carte_grise -> int -> vignette = <fun> *)

let rec achat_vignettes_location cgl a = match cgl with
   | [] -> []
   | cg::cgl' -> (achat_vignette cg a)::(achat_vignettes_location cgl' a);;
(* ou *)
let achat_vignettes_location cgl a =
  map (function cg -> achat_vignette cg a) cgl;;
(* achat_vignettes_location : carte_grise list -> int -> vignette list = <fun>*)

(* Q10 *)
let rec examen_permis c reçus pl = match pl with
   | [] -> []
   | p::pl' -> (if mem p.personne reçus then (nouveau_permis p c) else p)::
               (examen_permis c reçus pl');;
(* examen_permis : *)
(*  catégorie_permis -> identité list -> permis list -> permis list = <fun> *)
```