```
(****
 *
 * Module Rolodex defines the main objects of the Rolodex system and the basic
 * rolodex editing operations.
 *
 *)

module Rolodex;

  export Rolodex, Card, Name;

  object Rolodex is
    components: n:Name and cs:Card*;
    description: (*
        A Roldex is a named collection of cards.
    *);
  end Rolodex;

  object Card is
    components: n:Name and id:Id and age:Age and s:Sex and addr:Address;
    description: (*
        A Card contains the basic information for a person stored in a Rolodex:
        name, SSSN id, age, sex, and street address.
    *);
  end Card;

  object Name is string;
  object Id is integer;
  object Age is integer;
  object Sex is Male or Female;
  object Male;
  object Female;
  object Address is string;

  operation Add
    inputs: r:Rolodex, c:Card;
    outputs: r’:Rolodex;
    description: (*
        Add the given card to the given rolodex if a card of the same id is not
        already there.
    *);
    precondition:
        (* There is no card in the input Rolodex with the same id as the given
         * input card. *)
        not (exists (c’ in r.cs) c’.id = c.id)

            and

        (* The values of the given card fields are valid *)
        CardIsValid(c);

    postcondition:
        (* A card is in the output rolodex iff it is in the input rolodex or
         * it’s the new card to be added. *)
        forall (c’:Card)
            (c’ in r’.cs) iff ((c’ in r.cs) or (c’ = c));
end Add;

operation Delete
  inputs: r:Rolodex, id:Id;
  outputs: r’:Rolodex;
  description: (*
      Delete the card of the given id from the given rolodex.
  *);
  precondition:
      (* There is a card of the given id in the input Rolodex *)
      exists (c’ in r.cs) c’.id = id;

  postcondition:
      (* A card is in the output rolodex iff it is in the input rolodex and
       * its id is different than the given id. *)
      forall (c’:Card)
          (c’ in r’.cs) iff ((c’ in r.cs) and (c’.id != id));
end Delete;

operation Change
  inputs: r:Rolodex, id:Id, c:Card;
  outputs: r’:Rolodex;
  description: (*
      Change the card of the given id in the given rolodex to the given new
      card.
  *);
  precondition:
      (* There is a card of the given id in the input Rolodex and that card
       * is not identical with the given input card. *)
      exists (c’ in r.cs) (c’.id = id) and (c’ != c);

  postcondition:
      (* A card is in the output rolodex iff it is in the input rolodex and
       * is not the card with the given id in the input rolodex, or it’s the
       * given input card. *)
      forall (c’:Card)
          (c’ in r’.cs) iff (((c’ in r.cs) and
              (not exists (c’’ in r.cs) (c’’.id = id) and (c’’ = c’))) or
              (c’ = c));
end Change;

operation Find
  inputs: r:Rolodex, n:Name;
  outputs: cl:Card*;
  description: (*
      Find all of the cards of the given name in the given rolodex.
  *);
  postcondition:

      (* Cards in the output list consist of those in the input Rolodex with
       * the given name *)
      forall (c:Card)
          (c in cl) iff ((c in r.cs) and (c.n = n))

          and

```

```
        (* The output card list is sorted by ascending
         * order of card id. *)
        forall (i,j:integer |
           (i in [1..#cl]) and (j in [1..#cl]))
                if i<j then cl[i].id < cl[j].id;
  end Find;

  function CardIsValid(c: Card)->boolean =
    (* The length of the name is <= 30 characters *)
    (#(c.n) <= 30)

        and

    (* The length (i.e, numer of digits) of the id is 9 *)
    (#(c.id) = 9)

        and

    (* Age is a reasonable range *)
    ((c.age >= 0) and (c.age <= 200))

        and

    (* The length of the address is <= 40 chars *)
    (#(c.addr) < 40);

end Rolodex;
```