```haskell
--
-- Address management for Addressbook application
--
-- Author: Gabriele Keller

module AddressDB (
  Address(..),
  AddressBook,
  newAddressBook, -- :: AddressBook
  insertAddress,  -- :: AddressBook -> Address -> AddressBook
  deleteAddress,  -- :: AddressBook -> Address -> AddressBook
                  -- delete address in adr. book, returns
                  -- org. book if adr. not in book
  changeAddress,  -- :: AddressBook -> Address -> Address -> AddressBook
  findMatches,    -- :: AddressBook -> String -> [Address]
                  -- given a search string, find all entries
                  -- where str is a prefix of first or last name
) where

import Database


type FirstName = String
type LastName  = String
type PhoneNr   = Int

data Address = Address  FirstName LastName PhoneNr
               deriving (Ord, Eq)


type AddressBook = Database Address

-- create new address book
newAddressBook:: AddressBook
newAddressBook = newDB

-- insert address
insertAddress:: AddressBook -> Address -> AddressBook
insertAddress addrBook addr =
  insertItemDB addrBook addr

-- delete address in adr. book, returns
-- org. book if adr. not in book
deleteAddress:: AddressBook -> Address -> AddressBook
deleteAddress = deleteItemDB

-- find all entries where search string is a
-- prefix of first or last name
findMatches:: AddressBook -> String -> [Address]
findMatches adrDB str =
  searchDB adrDB preFixAddr
  where prefix [] s = True
        prefix (x:xs) (y:ys)  = (x == y) && (prefix xs ys)
        preFixAddr (Address fn ln _) =
           (prefix str fn) || (prefix str ln)


-- update an exisiting address
changeAddress:: AddressBook -> Address -> Address -> AddressBook
changeAddress = changeItemDB
```