Lab 8 – Data Abstraction

Suppose that we want to keep a database with information about our friends. In it we'll store their names, birth dates, and the cities and states in which they live. This database will be organized as a list of *records*, where each record contains all of the information about a particular friend. We can represent this abstractly as follows:

`<database> = (<record1> <record2> <record3> . . . <recordN>)`

Each record will be represented as a list of "facts" about a person. Each fact is a two-element list containing (1) a symbolic name for the *type* of information being represented, and (2) the piece of information itself.

For example, one possible record is shown below:

```
((name "Jane Doe") (city "New York") (state "NY")
 (birth-month 11) (birth-day 25) (birth-year 2002))
```

The constructor function `make-record` is used to create new records:

```
(define make-record
  (lambda (name city state month day year)
    (list
      (list 'name name)
      (list 'city city)
      (list 'state state)
      (list 'birth-month month)
      (list 'birth-day day)
      (list 'birth-year year))))
```

The selector function for records is `retrieve-info`, which takes a type symbol and a record and retrieves the appropriate information from the record. If the record contains no information of the given type, then the symbol `unknown` is returned. For example:

```
(define rec1 (make-record "Jane Doe" "New York" "NY" 11 25 2002))

(retrieve-info 'name rec1) → "Jane Doe"
(retrieve-info 'state rec1) → "NY"
(retrieve-info 'birth-year rec1) → 2002
(retrieve-info 'shoe-size rec1) → unknown
```

1. Complete the definition of the **retrieve-info** function, and test it as shown in the examples above.

```
(define retrieve-info
  (lambda (type record)
    ...))
```

2. A database called `mydb` is already provided for you for testing purposes, containing records for several people, including Jane Doe. Add yourself and a few of your friends to this database. Then, using your `retrieve-info` function as a helper, define the functions **find-record** and **find-all**. These functions each take as arguments a *type symbol*, some information of that type, and a database.

```
(define find-record
  (lambda (type value database)
    ...))

(define find-all
  (lambda (type value database)
    ...))
```

The function `find-record` returns the first record in the database that contains the given information, or the symbol `unknown` if no such record exists. For example:

```
(find-record 'city "New York" mydb)
```

would return Jane Doe's record. The function `find-all` returns a list of all of the records that contain the given information. For example:

```
(find-all 'city "New York" mydb)
```

would return a list of records for all people located in New York. IMPORTANT: Both `find-record` and `find-all` are allowed to access information in records ONLY through the `retrieve-info` function.

3. Next, write the functions **retrieve-age**, **where-is**, and **send-cards** by using the functions `find-record` and `find-all` as helpers. These functions should not depend on the assumption that the database is represented as a list of records.

```
(define retrieve-age
  (lambda (name database)
    ...))

(define where-is
  (lambda (name database)
    ...))

(define send-cards
  (lambda (month database)
    ...))
```

`Retrieve-age` takes the name of a person, and a database, and returns the person's age (as of 2021), or the symbol `unknown` if no such person exists in the database. For example:

```
(retrieve-age "Jane Doe" mydb)   → 19
(retrieve-age "Nobody" mydb) → unknown
```

`Where-is` takes a person's name, and a database, and returns a list containing the person's city and state, or the symbol `unknown` if no such person exists in the database. For example:

```
(where-is "Jane Doe" mydb)   →   ("New York" "NY")
(where-is "Nobody" mydb) → unknown
```

`Send-cards` takes a month number, and a database, and returns a list of the names of all people whose birthday occurs during the given month. If no one has a birthday in the specified month, the empty list is returned. Hint: use `map`. This could be useful for remembering to send birthday cards. For example:

```
(send-cards 11 mydb) → ("Jane Doe")
```

IMPORTANT: These functions should access information in the database ONLY through the functions `retrieve-info`, `find-record`, and `find-all`.

4. Suppose we wish to change the representation of records as follows. Instead of each record being a list of symbol-information pairs, a record will now contain a list of all of the type symbols, followed by a list of all of the information for that record, in the same order as the type symbols. For example, Jane Doe's record will now look like this:

```
((name city state birth-month birth-day birth-year)
 ("Jane Doe" "New York" "NY" 11 25 2002))
```

Rewrite the definitions of **make-record** and **retrieve-info** to reflect this new representation. These functions should still take exactly the same arguments as before. Do you need to rewrite any of the functions from Parts 2 and 3 above? Verify your answer by re-testing all of your higher-level functions using the new record format.