Exercise 2.74.

Insatiable Enterprises, Inc., is a highly decentralized conglomerate company consisting of a large number of independent divisions located all over the world. The company's computer facilities have just been interconnected by means of a clever network-interfacing scheme that makes the entire network appear to any user to be a single computer. Insatiable's president, in her first attempt to exploit the ability of the network to extract administrative information from division files, is dismayed to discover that, although all the division files have been implemented as data structures in Scheme, the particular data structure used varies from division to division. A meeting of division managers is hastily called to search for a strategy to integrate the files that will satisfy headquarters' needs while preserving the existing autonomy of the divisions.

Show how such a strategy can be implemented with data-directed programming. As an example, suppose that each division's personnel records consist of a single file, which contains a set of records keyed on employees' names. The structure of the set varies from division to division. Furthermore, each employee's record is itself a set (structured differently from division to division) that contains information keyed under identifiers such as `address` and `salary`. In particular:

a. Implement for headquarters a `get-record` procedure that retrieves a specified employee's record from a specified personnel file. The procedure should be applicable to any division's file. Explain how the individual divisions' files should be structured. In particular, what type information must be supplied?

b. Implement for headquarters a `get-salary` procedure that returns the salary information from a given employee's record from any division's personnel file. How should the record be structured in order to make this operation work?

c. Implement for headquarters a `find-employee-record` procedure. This should search all the divisions' files for the record of a given employee and return the record. Assume that this procedure takes as arguments an employee's name and a list of all the divisions' files.

d. When Insatiable takes over a new company, what changes must be made in order to incorporate the new personnel information into the central system?

Answer.

a. Figure 1 shows us the operations for headquarter in the perspective of data-directed style. Hence,

		TYPES		
		`division1`	`division2`	⋯
OPERATIONS	`get-record`	`get-record-division1`	`get-record-division2`	
	`get-salary`	`get-salary-division1`	`get-salary-division2`	⋮
	⋮	⋮	⋮	

Figure 1. Table of operations for headquarters.

the `get-record` procedure required by headquarter can be implemented as:

```
(define (get-record employee-name personnel-file)
  (let ((division-tag (type-tag personner-file)))
    ((get 'get-record division-tag) employee-name (contents personnel-file))))
```

In order to integrate the files as required, all the divisions' file must be attached tags (such as `division1`) that specify their type one another.

b. Given an employee's record, we can implement for headquarter a `get-salary` procedure as:

```
(define (get-salary employee-record)
  (let ((division-tag (type-tag employee-record)))
    ((get 'get-salary division-tag) (contents employee-record))))
```

This requires all the records to attach the tag of their division whenever adjoined into the system.

c. The `find-employee-record` procedure required by headquarter can be implemented as:

```
(define (find-employee-record employee-name personnel-file-set)
  (if (null? personnel-file-set)
      '()
      (map (lambda (personnel-file)
             (get-record employee-name personnel-file))
           personnel-file-set)))
```

d. When a new personnel file is added into the system, all the changes required is to attach corresponding tags on to the division files as well as every employee's record it contains.