

## University of Massachusetts Lowell

---

91.301: Organization of Programming Languages  
Fall 2004

### Problem Set 6 **Airline Mergers: Using Generic Operations**

Issued: Section 201 (Dimock): Monday, 25-Oct-2004, Section 202 (Yanco): Thursday, 28-Oct-2004  
Due: Section 201 (Dimock): Wednesday, 3-Nov-2004, Section 202 (Yanco): Thursday, 4-Nov-2004  
Reading: Covers text (SICP 2nd Edition by Abelson & Sussman): Through 2.5.2

### Overview

This assignment asks you to integrate three different file systems using dispatch-by-type and data-directed programming techniques. Section 2.4, and the beginning of section 2.5 of the book should be helpful in doing this assignment. In addition, the structure of one of the file systems uses trees to represent sets. If you have trouble reading the code in the appendix, you might want to read sections 2.3.3 and 2.3.4 in the book, which cover some applications of trees.

For this problem set you will be writing a fair bit of code, and packaging it in several different ways. Luckily, if a problem has several different parts, the code can generally be written by coding the first part, and then cutting and pasting snippets from the first part and extending the functionality to get other parts. The heart of the problem set is in Problems 3 and 4 – which have less code than Problems 1 and 5. Problems 3 and 4 can be worked without the preceding problems, but can not be tested thoroughly Problem 1.

You will turn in

- `ps6-ans.ss` containing new procedures
- `ps6-type.ss` packaging code by operation for dispatch on type
- `ps6-dd.ss` packaging code for table dispatch

Note that the `submit` command requires you to submit *all* your files for the assignment in a single command.

The code for this problem set is in two files: `ps6.ss` contains the lookup, insert, and delete code, the code for data abstraction, the tagging code, and the “file systems” for testing. You need to read and understand this code to work the problem set. `ps6-table.ss` contains an implementation of tables with `put` and `get`. You do not need to understand this code to use it. `put` and `get` are described on Page 181 of the textbook.

## An adaptation of a True Story of Our Time

The high cost of service has caused declining profits, or increasing losses for most major airlines. Two trends are emerging: small no-frills airlines currently seem to be profitable. But as the old giants fail, they seem to be bought out by other giant airlines forming increasingly fewer large carriers. Among these unfortunate souls are three airlines that have decided to merge to form one single airline. They are **People-Delay Airline** (slogan - *Fly the executive style* (i.e. always late)), **New Jersey Air** (slogan - *JETS in the Meadowlands*) and **Epsilon Airline**, a Delta Airline spin-off (slogan - *We achieve perfection so long as the error lies in the neighborhood of epsilon*). These three airlines are, however, very worried about integrating their personnel files. Their file systems are structured differently, and they have so many employees that designing a completely new file system would require them to hire 10 data entry clerks for 3 years to enter all the data into the new system. The CEO at People-Delay Airline, Al Cheapo, thinks that this is too expensive. He has appointed you to think up a way of integrating the three file systems without having to modify the internals of any one system. You, of course, know a good trick for doing this, if you have gone to the recent lectures.

### Test personnel files (also in ps6.ss)

The small personnel files exist in two versions: `nj-air-database`, `people-delay-database`, and `epsilon-air-database` just contain employee records. `nj-air`, `people-delay`, and `epsilon-air` are tagged versions for use in Problem 3 and later.

```
;; individual databases, untagged.
(define nj-air-database
  (list 'moe (cons 'salary 40000) '(address 88 martin st))
  (list 'joe (cons 'salary 30000) '(address 77 salem st)))

(define people-delay-database
  (list 'jane '(address 350 woodcock st) (cons 'salary 34000))
  (list 'ruth '(address 90 sparks st)))

(define epsilon-air-database
  '(leaf (bob ((leaf (address 23 bachman st))
                 (leaf (salary . 45300))
                 (address salary))
          (address salary)))
  '(leaf (amy ()
            (leaf (address 79 emery st))
            (address)))
  '(bob amy))

;; tags for various databases
(define nj-air-tag 'nj-air)
(define people-delay-tag 'people-delay)
(define epsilon-air-tag 'epsilon-air)

;; tagged databases
(define nj-air (attach nj-air-tag nj-air-database))
(define people-delay (attach people-delay-tag people-delay-database))
(define epsilon-air (attach epsilon-air-tag epsilon-air-database))

;; combined-database for final problem:
(define personnel-file (list nj-air people-delay epsilon-air))
```

**Problem 1:**

Just when you were ready to begin work, a small fire broke out in the computer lab at People-Delay. While all the data was backed up offsite, somehow the software engineering group managed to never back up their code! As a result, the procedures for looking up, inserting and deleting employee records for People-Delay's personnel file were lost. Your first task is to reconstruct those procedures. You begin by analyzing the structure of People-Delay's personnel file.

Notice that the names of the employees are alphabetized from left to right. In fact, the personnel file is structured exactly the same as New Jersey Air's file structure, except for one difference - the employees are alphabetically arranged in People-Delay Airline's file, while the employees are randomly arranged in New Jersey Air's file structure.

Reconstruct the `lookup`, `insert` and `delete` procedures for People-Delay's personnel file `people-delay-database`. Try to make your procedures efficient by using the fact that employee names are alphabetized. Name these procedures `lookup-ordered`, `insert-ordered` and `delete-ordered`. Test each procedure using the miniaturized file system provided in the appendix.

When you test out the `insert-ordered` procedure, use the given constructor `make-record-table` to make a new record with salary information for Alyssa, who makes 35 thousand dollars a year at People-Delay.

A copy of one of the routines has recently been found: the `lookup-ordered` procedure has turned up in the oddly-named file `ps6-ans.ss`.

Hand in a listing of your procedures and examples (as comments) of the execution of each procedure. Submit this material in `ps6-ans.ss`.

**Problem 2:**

Just out of curiosity, you want to find out what the structure of the personnel file of Epsilon Airline looks like. The personnel file presented in the code is again a minaturized version of the real file (remember that the size of the real file has caused Al Cheapo to ask you to handle this job in the first place). In 2 sentences or less, summarize how insertion of an employee record is done. Similarly, summarize how deletion of an employee record is done.

Your description should be submitted as comments in `ps6-ans.ss`.

**Problem 3:**

Now, you are ready to integrate the three file systems. Remember that what you want to achieve is to have a generic `lookup`, a generic `insert` and a generic `delete` operator, which, when given the right employee name and the right file (and the right employee record, in the case of insertion), will select the right procedure to perform the operation, depending on the requirements of the file system. You may realize that between the two integration methods - **dispatch-by-type** and **data-directed-programming**, one is more suitable for combining a small number of systems, and the other is more suitable for combining a large number of systems. Since there are only three file systems, write your code using `dispatch-by-type`.

You may find the `tagged-as?` procedure to be useful.

Hand in a listing of your new code and some examples in comments demonstrating that the generic operators work as desired on `nj-air`, `people-delay`, and `epsilon-air`. This code should be in `ps6-type.ss`.

**Problem 4:**

Al Cheapo is very pleased with your work. “If integrating these personnel files is such an easy task”, he says, “then I am going to persuade more airlines to merge with us!!” This means that you now have to use the other method to implement the integration.

Al’s business plan has the effect that you need to redesign now to use data-directed-programming.

Hand in a listing of your new code and examples demonstrating that the generic operators under this new system also work. Remember that the user’s interfaces for `lookup`, `insert`, and `delete` are unchanged from the previous version.

This code should be in `ps6-dd.ss`.

**Problem 5:**

You have probably noticed that for each company, the structure of each employee record is exactly the same as the structure of the file itself. That is, the structure of an employee record for the `unordered` type of file is again a list of pairs, where the `car` of each pair points to an identifier, and the `cdr` of each pair points to the information associated with that identifier, and the identifiers are also unordered like the names. Similarly, the employee record of the `ordered` type has the same structure as the ordered file itself. The employee record of the `tree` type file has also a tree structure. In other words, we have a recursion of structure in the files. The employee name has the same position in the record as the tag has on the tagged personnel file.

Taking advantage of this observation, write three procedures: `record-lookup`, `record-insert` and `record-delete`, which are also generic operators.

`Record-lookup` takes the name of an employee, the appropriate personnel file and an identifier, which could be either the symbol `salary` or `address`, and returns the salary or address of the employee if the employee is found in the given personnel file. For example:

```
(record-lookup 'joe nj-air 'address)

(address 77 salem st)
```

`Record-insert` takes the name of an employee, the appropriate personnel file, an identifier and the information associated with that identifier, and returns the new personnel file with the information associated with the identifier inserted in the employee’s record, if the employee is found in the given personnel file. For example:

```
> (record-insert 'jane people-delay 'id 54321)
(people-delay
 (jane (address 350 woodcock st) (id . 54321) (salary . 34000))
 (ruth (address 90 sparks st)))
```

`Record-delete` takes the name of an employee, the appropriate personnel file and an identifier, and returns the new personnel file with the information associated with the identifier deleted (together with the identifier itself), if the employee is found in the given personnel file. `Record-delete` should return the original file if the employee record exists but does not contain the identifier to delete. For example:

```
> (record-delete 'ruth people-delay 'salary)
(people-delay
 (jane (address 350 woodcock st) (salary . 34000))
 (ruth (address 90 sparks st)))
> (record-delete 'ruth people-delay 'address)
(people-delay (jane (address 350 woodcock st) (salary . 34000)) (ruth))
```

All of the above routines should return `#f` if the employee record does not exist in the appropriate personnel file.

(Hint: Make use of the already existing generic `lookup`, `insert` and `delete` operators.) Include your code in `ps6-dd.ss`.

**Problem 6:**

After the new system has been in place for a while, payroll department head Ms. I. Luva Payday complains to you that she is tired of having to remember which employee is in which personnel file. She asks you to link all the files together into a master file,

(define personnel-file (list nj-air people-delay epsilon-air)) and have an operator lookup-global, which, when given the name of the employee and the masterfile, would automatically search through each personnel file until it finds the employee, and return the employee's record. For example:

```
> (lookup-global 'joe)
(joe (salary . 30000) (address 77 salem st))
> (lookup-global 'bob)
(bob
 ((leaf (address 23 bachman st)) (leaf (salary . 45300)) (address salary))
 ()
 (address salary))
> (lookup-global 'mark)
#f
```

Hand in a listing of your lookup-global procedure, and examples of its execution. Include your code in ps6-dd.ss.

Thus the three airlines live happily ever after with their new company name: Amalgamated Airline (slogan: "The generic airline of the 21st Century").