Lab 6

Programs

1. AS&S 2.2 makes good use of data abstraction. To show that your version is correctly implemented, show your code running, then load a file that redefines cons, car, and cdr as on page 91. Now show your code running correctly again.

   Now load a file that redefines cons, car, and cdr as in exercise 2.5 and repeat the demonstration.

2. AS&S 2.17

3. AS&S 2.55

4. AS&S 2.65

5. This section of the lab is not due until Wednesday night, April 8th.

   Read the book's section on dispatching-on-type and review the associated code before attempting this.

   Three hospitals store patient data as sets of patients. Some patient operators for McGeary's Holistic Medicine Tent are as follows:

```
;constructor
(define (make-patient name id symptoms-list)
        (cons name (cons id symptoms-list)))

;accessor
(define (get-patient-name patient)
        (car patient))
```

   Define get-patient-id and get-patient-symptoms.

   Also define these four operators for Harvey's House o' Healin, where the same patient info is stored in a list `(name id symptoms)`, and McCoy's Medical Miracle and Pancake House, where a patient record is a list `(id name symptoms)`.

   Now implement each hospital as a set of patient records. Define operators to retrieve a record based on name and id for each hosptital. McGeary uses an unordered list of patients, McCoy uses a list ordered by ID number, and Harvey stores patient records in a binary search tree by name. You will need to write a comparison function that orders two patient records; use string operators where necessary, as in

```
(string<? (symbol->string 'aardvark) "bongo")
```

   No two patients in one hospital will have the same ID, though two patients might have the same name. Patients in different hospitals may have the same name and ID.

   Implement a dispatching-on-type system that will find a patient by name or id (use the predicate `number?`) in any hospital and display the patient record. If more than one patient in the combined system has the same name, print a warning and show all records with that name.

   Use proper data abstraction, as explained in class. The algorithms required here are fairly trivial (though I'll be happy to go over them with you during office hours) so abstraction and style will be counted heavily.

   Remember that you can put software in different files and then use it by including in another file (load "binary-tree.scm").

   I am providing defined lists of 2000 patient names and 2000 sets of symptoms (see the file in this directory). Match them in order, and give each

the appropriate patient number. For example, patient 1 would be Jacob, with symptoms of ("a sore throat"). Write simple code to do this using the defined lists and your sequence toolkit.

However, you cannot just make a list of 2000 patients, because they belong in different hospitals (with some overlap) and of course, will be represented differently in each hospital.

Patients numbered 1-950 are in McGeary's; 700-1650 are in McCoy's; 1400-2000 are in Harvey's. Thus some patients have records in more than one hospital. For example, Braylon is patient 1399 and so is in McCoy, but patient 1400, Alisa, has records in both Harvey and McCoy. Write a simple function that extracts what you want from the lists before making the separate records for each hospital.

Use what you know about good coding practice to make this task easy. Don't use magic numbers, work with layers of code and good data abstraction. Remember, days of coding can save hours of planning.

Do not "hard code" lists of patients for hospitals, or whole patient records. Your code must create the patient lists dynamically (why?) to receive credit. You may name the different lists using define. I did so, and as a result I can call

```
> (find-id 1399)
((mccoy 1399 "Braylon" ("scurvy" "below normal temperature")))
> (find-id 1401)
((harvey "Karl" 1401 ("scurvy" "bent bones"))
 (mccoy 1401 "Karl" ("scurvy" "bent bones")))
> (find-name "Karl")
((harvey "Karl" 1401 ("scurvy" "bent bones"))
 (mccoy 1401 "Karl" ("scurvy" "bent bones")))
> (find-name "Braylon")
((mccoy 1399 "Braylon" ("scurvy" "below normal temperature")))
```

Write both of these functions and demonstrate them working on the following patients:

```
 "Gavin" "Amy" "Rodney" "Alyson" "Violet"
"Remington" "Turner" "Yareli"
```