CS1101, A04

Exam 2

Name:

Problem	Points	Score
1	36	
2	32	
3	32	
	Total	

You have 50 minutes to complete the problems on the following pages. There should be sufficient space provided for your answers. You do not need to show templates, but you may receive partial credit if you do. You also do not need to show test cases or examples of data definitions (unless a problem states otherwise), but you may develop them if they will help you write the programs.

Your programs may contain only the following Scheme constructs:

define define-struct cond else

and the following primitive operators:

empty? cons? cons first rest list
$+ - * / = < > <= >=$ *zero?*
string=? string-length symbol=?
and or *not*

and the operators introduced by **define-struct**.

You may, of course, use whatever constants are necessary (*empty*, *true*, *false*, 0, etc).

(If you did the advanced lab 4 and want to use *map* or *filter*, go ahead, but make sure you know what you are doing first.)

---

1. (36 points) An airline maintains information on its flights using the following data definition:

```
;; A flight is a (make-flight string string symbol)
(define-struct flight (from to frequency))

;; Example: (make-flight "Boston" "Chicago" 'daily))
```

The frequency is one of 'daily, 'weekday, or 'weekend.

(a) (18 points) Write a function *service-between?* that consumes two city names and a list of flights and determines whether there is a flight from the first city to the second.

```
;; service-between? : string string list-of-flight → boolean
;; consumes two city names and list of flights and determines whether there is a flight from first city to second
```

(exam continues next page)

(b) (18 points) Write a function *daily-to* that consumes a city name and a list of flights and returns a list of cities (strings) to which the airline has a daily flight from the given city.

```
;; daily-to : string list-of-flight → list-of-string
;; consumes city and list of flights and produces list of cities with daily service from named city
```

(exam continues next page)

2. (32 points) Consider the following definition of family trees used in a medical database:

An *ftree* is one of
- 'unknown,
- (*make-person string string ftree free*)

(**define-struct** *person* (*name disease father mother*))

(a) (16 points) Write a function *count-disease* that consumes a disease and a family tree and produces the number of people in the family tree who have the given disease.

;; count-disease : string ftree → number
;; consumes disease and family tree and produces number of people with that disease

(exam continues next page)

(b) (16 points) Write a program *update-disease* that consumes a person's name, a disease, and a family tree and produces a family tree in which the named person now has the named disease. All other information in the tree should remain the same. You may assume that the given name appears only once in the tree.

;; update-disease : string string ftree → ftree
;; consumes person's name, a disease and a family tree and produces family tree in which
;; named person has given disease and all other info in tree remains the same

(exam continues next page)

3. (32 points) An emergency response team maintains a hierarchy of who calls who (and at what phone number) in the event of an emergency. For example, Ann calls Bill and Mark, Bill calls Susan, Mark calls David and Omar, etc. One person starts the calls (in this example, Ann). Note that each person may be responsible for calling different numbers of people. Every person should have a phone number, even if nobody else calls them.

   (a) (16 points) Provide a data definition that captures people and who they call as described above. **Include the arrows on your data definition.** You may use a single number for the phone numbers (ie, no need to separate out area codes, etc). (You can check your work by trying to write the above example in your data definition, but **an example is not required**.)

(exam continues next page)

(b) (16 points) The team wants to evaluate how long it takes to notify everyone of an emergency. Write a function *max-time-to-notify* that consumes a person and produces a number (the most time it takes before everyone starting from that person has gotten a phone call). Assume that each person needs 1 minute to call each other person they must notify. On the example on the previous page, the time to notify is 4 minutes (2 for Ann's calls, then another 2 for Mark's; Bill's calls take less time than Mark's). Ignore the order in which people get called (ie, the result should be the same whether Ann called Mark or Bill first). You may use Scheme's built in operators *max* (returns the largest of its inputs) and *length* (returns the length of a list).

;; max-time-to-notify : person → number
;; consumes person and produces time needed to call everyone in the hierarchy

(end of exam)