CS2135, A03

Midterm Exam

Name:

Problem	Points	Score
1	20	
2	24	
3	24	
4	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 models, but you may develop them if they will help you write the programs.

Your programs may contain only the following Scheme syntax:

define define-struct cond else lambda local let

and the following primitive operations:

empty? cons? cons first rest list map filter length
number? $+ - * / = < > <= >=$ *zero?*
symbol? symbol=? eq?
boolean? **and or** *not*

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

You may, of course, use whatever constants are necessary.

---

1. (20 points) An on-line discussion board like myWPI provides multiple forums (topics) for discussion. A user's view of a single forum has a title, a list of messages, and a number indicating how many messages are unread.

```
;; A forum is a (make-forum symbol list[string] number)
(define-struct forum (title messages unread))

;; Example: (make-forum 'Announcements (list "Homework posted" "Exam today") 1)
```

   (a) (10 points) Write a program *all-read?* that takes a list of forums and returns a boolean indicating whether all messages in all the forums have been read.

```
;; all-read? : list[forum] → boolean
;; determine whether all messages in all forums have been read
```

   (b) (10 points) Write a program *add-message* that takes the title of a forum, a message and a list of forums and returns a list of forums. In the returned list, the named forum includes the new message in the list of messages and has its unread count incremented by 1. All other forums in the returned list are the same.

```
;; add-message : symbol string list[forum] → list[forum]
;; returns list of forums with message added as unread to the named forum
```

(exam continues next page)

2. (24 points) Consider the following data defintion:

```
;; A city is a (make-city symbol symbol boolean number list[symbol])
(define-struct city (name state airport? hotel-room-count features))

;; Example: (make-city 'Bloomington 'Indiana true 1250 (list 'basketball 'hiking 'restaurants))
```

Assume that *cities* is a list of city. For each of the following expressions, give an English description of what the expression produces (i.e., a purpose statement like "produces a list of planes with high mileage" or "increases the mileage of all planes"). Do not simply rephrase the code into English–your answer should describe the result, not how the expression works.

Recall that *length* counts how many items are in a list and *sum* adds up the numbers in a list of numbers. *member* determines whether a symbol is in a list of symbols.

(a) (8 points)

```
(filter (lambda (acity) (and (not (city-airport? acity))
                             (member 'skydiving (city-features acity))))
        cities)
```

(b) (8 points)

```
(sum (map city-hotel-room-count
          (filter (lambda (acity) (symbol=? (city-state acity) 'Wyoming))
                  cities)))
```

(c) (8 points)

```
(length (filter (lambda (n) (> n 100))
                (map city-hotel-room-count
                     (filter city-airport? cities))))
```

(exam continues next page)

3. (24 points) You want to augment a discussion board system like myWPI with a message notification feature that periodically sends email to subscribers informing them of unread messages. Each subscriber supplies their name, email address, and a function from a list of forums (same definition as in question 1) to boolean that indicates whether to send a notification.

```
;; A forum is a (make-forum symbol list[string] number)
(define-struct forum (title messages unread))

;; A subscriber is a (make-subsc symbol symbol (list[forum] → boolean))
(define-struct subsc (name email update?))
```

   (a) (12 points) Define two subscribers: one that wants to be informed if there is an unread message on at least one forum, and another who wants to be informed only if there are at least 2 unread messages in the forum named 'homework. Make up names and email addresses for these subscribers. You may use the functions from question 1 as helper functions without copying them. Define all other helper functions that you use as part of your answer.

(exam continues next page)

(b) (12 points) Write a program *notify* that consumes a list of subscribers and returns a list of email addresses (symbols, according to the data definition). The returned list should consist of the email addresses of all subscribers who want to be notified about the current forum status. Assume you have a helper function *get-forums* that takes a subscriber name and returns a list of forums (this function determines the number of unread messages based on the name of the subscriber). **Do not try to write get-forums – simply assume it is provided and use it in your solution.**

;; notify : list[subsc] $\rightarrow$ list[symbol]
;; returns list of email addresses of subscribers to notify about unread messages
;; uses get-forums helper function to get the list of forums per user

(exam continues next page)

4. (32 points) Decision trees capture a hierarchy of questions with yes/no answers. Each internal node of the tree is a question, and the leaves contain answers. To use a decision tree, you ask the question at the top of the tree, then repeat on the yes or no branch depending on the answer to the question.

```
;; A decision-tree is either
;; - symbol, or
;; - (make-branch string decision-tree decision-tree)
(define-struct branch (ques yes no))

;; Example of decision tree to recommend activities to people
(define DT (make-branch "Do you like outdoors activities?"
                        (make-branch "Do you like covering long distances?"
                                     (make-branch "Like having your feet near the ground?"
                                                  'bicycling
                                                  'hang-gliding)
                                     'hiking)
                        'ping-pong))
```

(a) (16 points) Write a program *get-solution* that takes a decision tree and a list of symbols ('y or 'n) and returns a symbol. The returned symbol is the answer at the leaf found by following the yes and no options as indicated in the list. For example (*get-solution DT* (*list* 'y 'y 'y)) would return 'bicycling, while (*get-solution DT* (*list* 'n)) would return 'ping-pong.

Assume that the list of symbols has enough symbols to reach a leaf.

```
;; get-solution : decision-tree list[symbol] → symbol
;; return the result for the path through the tree given in the list
```

(exam continues next page)

(b) (16 points) Write a program *replace-result* that takes a result in the tree (a symbol), a tree to insert in place of the result, and the decision tree to make the replacement in. The function returns a tree with the same contents as the third argument except the result is replaced with the second argument. For example, (*replace-result* 'ping-pong (*make-branch* "Are you lazy?" 'tv 'ping-pong) *DT*) would return the tree

```
(define DT (make-branch "Do you like outdoors activities?"
                        (make-branch "Do you like covering long distances?"
                                     (make-branch "Like having your feet near the ground?"
                                                  'bicycling
                                                  'hang-gliding)
                                     'hiking)
                        (make-branch  "Are you lazy?" 'tv 'ping-pong)))
```

```
;; replace-result : symbol decision-tree decision-tree → decision-tree
;; replace symbol (result) with first tree in second tree
```