CS2135, C03

Midterm Exam

Name:

Problem	Points	Score
1	20	
2	24	
3	20	
4	36	
	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

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) In a game involving robots, a robot can do one of two things at each round of the game: move forward a certain number of steps, or rotate position (left or right). The following data definition captures robot moves:

   A robot-move-lst is either
   - *empty*
   - (*cons number robot-move-lst*)
   - (*cons symbol robot-move-lst*)

   For example, the list (*list* 4 'left 2 'right 'left) says that the robot moves 4 steps on the first round, rotates left on the second, moves 2 steps on the third, and so on.

   Write a program *total-steps* that takes a robot-move-lst and returns the total number of steps that the robot took during the game. **Your answer must follow the template for this data definition** (ie, you **may not** filter out the numbers and then add them up).

(exam continues next page)

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

A *plane* is a (*make-plane number symbol number boolean*)
(**define-struct** *plane* (*mileage mechanic months-since-service problems?*))

Assume that *planelist* is a list of planes. For each of the following expressions, give an English description of what the expression does (i.e., a purpose statement like "produces a list of planes with high mileage" or "increases the mileage of all planes")

(a) (8 points)

```
(map months-since-service
     (filter (lambda (aplane) (symbol=? (plane-mechanic aplane) 'Herbie))
          planelist)
```

(b) (8 points)

```
(map (lambda (aplane)
       (cond [(symbol=? (plane-mechanic aplane) 'Julie)
              (make-plane (plane-mileage aplane) 'Julie 0 false)]
             [else aplane]))
     planelist)
```

(c) (8 points)

```
(empty? (filter (lambda (num) (< num 10000))
                (map plane-mileage
                     (filter plane-problems? planelist))))
```

(exam continues next page)

3. (20 points) Consider the following two programs:

```
;; list-nums : number → list[number]
;; return list of numbers from given number down to 1
(define (list-nums n)
  (cond [(zero? n) empty]
        [else (cons n (list-nums (- n 1)))]))
;; list-squares : number → list[number]
;; return list of squares of numbers from given number down to 1
(define (list-squares n)
  (cond [(zero? n) empty]
        [else (cons (* n n) (list-squares (- n 1)))]))
```

   (a) (10 points) Write a function *build-num-list* that captures the common computation of these two functions (in similar spirit to how filter captures the common code of several functions).

(exam continues next page)

(b) (5 points) Rewrite *list-nums* and *list-squares* using your new *build-num-list*.

(c) (5 points) Can *build-num-list* be used to build output lists containing data other than numbers? Either explain why not or provide a revised contract for a function *build-list* that behaves similarly but returns output lists of a more general type.

(exam continues next page)

4. (36 points) A company maintains information on its employees and who each one supervises. The following data definition captures employees:

   ;; An employee is a (make-emp symbol symbol number list[emp])
   (**define-struct** *emp* (*name project salary supervises*))

   (a) (18 points) Write a program *all-on-project* that takes an employee (tree) and a project (symbol) and returns a list of the names of all employees who work on that project.

   ;; all-on-project : emp symbol $\rightarrow$ list[symbol]
   ;; return list of names of all employees who work on given project

(exam continues next page)

(b) (18 points) Write a program *give-raises* that takes an employee (tree) and gives all employees who earn less than 75,000 a 10% raise.

;; give-raises : emp $\rightarrow$ emp
;; returns tree where all employees who earned less than 75,000 got a 10% raise