CS2135, A02

Midterm Exam

Name:

Problem	Points	Score
1	24	
2	36	
3	40	
	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. Unless a problem states otherwise, you are not required to use map and filter in your answers.

Your programs may contain only the following Scheme syntax:

define define-struct cond else lambda let

and the following primitive operations:

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

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

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

---

1. (24 points) Consider the following data defintion for toys.

An *toy* is one of
- (*make-robot number number boolean?*)
- (*make-crayons number list*[*symbol*])
- (*make-bicycle number number symbol*

(**define-struct** *robot* (*price height programmable?*))
(**define-struct** *crayons* (*price colors*))
(**define-struct** *bicycle* (*price num-wheels color*))

Assume that *toylist* is a list of toys. 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 the red bicycles")

(a) (8 points)

```
(map crayons-colors (filter crayons? toylist))
```

(b) (8 points)

```
(empty? (filter (lambda (a-robot) (> (robot-height a-robot) 10))
                (filter robot-programmable?
                        (filter robot? toylist))))
```

(c) (8 points)

```
(map (lambda (toy)
       (cond [(bicycle? toy) (make-bicycle (* price .75)
                                           (bicycle-num-wheels toy)
                                           (bicycle-color toy))]
             [else toy]))
     toylist)
```

(exam continues next page)

2. (36 points) An on-line auction site (like EBay) wants to add an automatic bidding feature. For a given item, customers can provide a function that decides whether to bid on the item based on the current top bid and the number of minutes left in the auction. The site will periodically run these functions and register any generated bids.

   A monitor consists of the name of the item that a customer wants to bid on (e.g. 'dvd-player) and a function that takes two numbers (the top bid and the minutes remaining) and returns either a bid or false (if the customer doesn't want to bid on the item yet). The following data definitions capture both monitors and bids:

   A monitor is a
   (*make-monitor symbol* (*number number -> bid* **or** *false*))
   (**define-struct** *monitor* (*item-name try-bid*))

   A bid contains the customer's name and the amount they want to bid:

   A bid is a
   (*make-bid symbol number*)
   (**define-struct** *bid* (*cust-name amount*))

   (a) (10 points) Write a program *bidder-name* that consumes a bid amount and a list of bids and returns the name of the first customer in the list who bid the given amount.(or false if no customer bid the given amount).

   ;; bidder-name : number list[bid] → symbol or false
   ;; return name of first customer who bid the given amount

(exam continues next page)

(b) (10 points) A customer Eddie wants to bid $200 on a stereo if the top bid is less than $200 and the amount of time left in the auction falls below 5 minutes. Write the *make-monitor* expression that captures this automatic bid request.

(c) (16 points) Write a program *new-bids* that consumes the current top bid, the number of minutes remaining in the auction, and a list of monitors and returns a list of bids. The program runs each monitor in the input list. If the monitor returns a bid (instead of false), put the returned bid in the output list.

;; new-bids : number number list[monitor] → list[bid]
;; generates customer bids based on current bid and remaining time

(exam continues next page)

3. (40 points) Consider the following definition of family trees:

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

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

A medical researcher wants to augment and use this model to study health patterns in families.

(a) (5 points) Edit the data definition to store two additional pieces of information per person: pulse rate (a number) and whether the person smokes. Make sure you also add or edit the **define-struct**s as necessary (you may edit the existing data definition text; you do not need to recopy it). You do **not** need to create a separate struct for a person's health information.

(b) (10 points) Write a program *count-smokers* that consumes a ftree and counts the number of smokers in the tree.

;; count-smokers : ftree → number
;; count how many people in tree smoke

(exam continues next page)

(c) (15 points) Our researcher needs to count how many people satisfy various traits. All of these programs check a condition on each person in the tree and include that person in the count if they satisfy that condition. Write a program *count-trait* that consumes a function from person to boolean (the condition) and a ftree and returns the number of people in the tree that satisfy the condition.

;; count-trait : (person → bool) ftree → number
;; count how many people in tree satisfy given condition

(exam continues next page)

(d) (5 points) Rewrite your *count-smokers* function using *count-trait*.

(e) (5 points) Use *count-trait* to implement a function *percent-trait* that takes a condition (a function from person to boolean) and computes the percentage of people in the tree that satisfies the condition.