CS1102, A05

Midterm Exam

Name:

Problem	Points	Score
1	35	
2	35	
3	30	
	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, but you may develop them if they will help you write the programs. You do not need to use map/filter unless a problem states otherwise (you may use them if you wish).

Your programs may contain only the following Scheme syntax:

define define-struct cond else lambda local

and the following primitive operations:

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

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

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

---

1. (35 points) The registrar maintains information about each course as a structure containing the course title, professors' names, maximum enrollment, and a list of names of students in the course. The following data definition captures course info:

```
;; A course is a (make-course string list[string] number list[string])
(define-struct course (name profs max-seats students))

;; Example
(make-course "CS1102" (list "Fisler" "Gennert") 84 (list "Paolo" "Jennie" "Wilson"))
```

In the following problems, you may use Scheme's built-in function *length* that consumes a list and produces the number of items in that list.

(a) (11 points) Write a program *num-enrolled* that consumes the name of a course and a list of courses and produces the number of students enrolled in the named course. **You do not need to use map or filter for this function** (but you may if you wish).

```
;; num-enrolled : string list[course] → number
;; produces number of students enrolled in named course
```

(exam continues next page)

(b) (12 points) Write a program *closed-courses* that consumes a list of courses and produces a list of names of courses that have the maximum number of students already enrolled. **Use map and/or filter if possible.**

;; closed-courses : list[course] → list[string]
;; produces list of names of courses that are already closed

(c) (12 points) Write a program *enroll* that consumes the name of a course, the name of a student and a list of courses and produces a list of courses. The named student should be enrolled in the named course in the produced list if there is room in the course. All other courses should be in the returned list unchanged. Assume that the named student is not already in the course. **Use map and/or filter if possible**.

;; enroll : string string list[course] → list[course]
;; adds named student to named course if there is room in the course

(exam continues next page)

2. (35 points) Banks allow customers to specify payments that should happen automatically once a month (such as paying a bill or donating to a charity). Customers can make the amount of the payment conditional on the balance in their account. Each auto-payment specifies the date of the month (number) to perform the transaction and a function that consumes their balance and produces a money transfer structure (data defn below) indicating how much to pay.

   (a) (10 points) Assuming the following data definition for transfers, propose a data definition for auto-payments.

   ```
   ;; A transfer is a (make-transfer number string)
   (define-struct transfer (amount to))
   Example: (make-transfer 450 "Citizens Bank")
   ```

   (b) (10 points) Using your data definition, write the example of an automatic payment that sends $25 to the Red Cross on the first of every month in which the customer's balance is at least $500. (If the balance is below $500, the auto-payment function should produce a transfer of zero dollars to the Red Cross.)

(exam continues next page)

(c) (15 points) Write a function *daily-transfers* that consumes the current date (of the month), a customer's balance and a list of the customer's auto-payments and produces the list of transfers for payments to make on the given date (assuming the given balance). **You do not need to use map/filter**, but may if you wish.

(exam continues next page)

3. (30 points) Many sports tournaments are organized as rounds in which each team plays one other team in the round and the winning team advances to the next round. At the end of the tournament, the games that occured in each round can be summarized using the following data definition:

```
;; A tournament is either
;; - a string (the name of a team), or
;; - (make-game string game-tree game-tree)
(define-struct game (winner prev-round1 prev-round2))
```

The following example shows the hierarchy of rounds on the left written using this data definition on the right:

```
      WPI
     /   \
    BU   WPI
   /  \   | \
Yale BU  WPI MIT
```

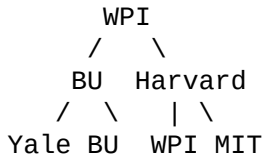
```
;; Example of a game tree
(define tourney1 (make-game "WPI"
                            (make-game "BU" "Yale" "BU")
                            (make-game "WPI" "WPI" "MIT")))
```

(a) (15 points) Write a program *num-wins* that consumes the name of a team and a tournament and produces the number of games that the team won during the tournament. For the example above, WPI has 2 wins while MIT has no wins.

```
;; num-wins : string tournament → number
;; produce number of times team won in tournament
```

(exam continues next page)

(b) (15 points) A tournament is valid if the winner of every game was actually a player in that game. The example tournament shown on the previous page is valid while the next one is not because Harvard did not play in the game between WPI and MIT.



Write a program *valid?* that consumes a tournament and produces a boolean indicating whether the tournament is valid.

;; valid? : tournament → boolean
;; determine whether winner of all games was a player in that game

(end of exam)