CS1101, A04

Exam 1

Name:

Problem	Points	Score
1	15	
2	20	
3	25	
4	20	
5	20	
	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
$+ - * / = < > <= >=$ *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).

---

1. (15 points) Write a function *largest-of* that consumes three numbers and produces the largest of the three numbers. Your solution should include the contract and purpose.

(exam continues next page)

2. (20 points) Create a cleaner version of the following code by introducing helper functions and constants. The final code should have the same behavior as the original code.

```
;; pipe-surface : number number number → number
;; consumes the length, diameter (width), and hole diameter of a pipe and produces the surface area
;; of the pipe (areas of the rings on each end plus areas of the outer and inner pipe walls)
(define (pipe-surface pipe-len pipe-diam hole-diam)
  (+ (* 2 (- (* 3.142 (/ pipe-diam 2) (/ pipe-diam 2))
             (* 3.142 (/ hole-diam 2) (/ hole-diam 2))))
     (* pipe-len 3.142 pipe-diam)
     (* pipe-len 3.142 hole-diam)))
```

(exam continues next page)

3. (25 points) A bank offers two kinds of accounts. *Checking accounts* offer a certain number of free checks each month but require a minimum balance (for example: 3 free checks with no minimum balance, or 10 free checks with $500 minimum balance). *Savings accounts* have an interest rate (like 2% or 1.4%), a minimum balance (like $100), a fee (like $5) for when the balance is below the minimum, and the date of the last transaction on the account. Dates consist of a day, month, and year.

   (a) (15 points) Provide data definitions for savings accounts, checking accounts, and accounts in general. Provide examples of data for each **define-struct** that you write.

(exam continues next page)

(b) (10 points) Write a function *fee-owed* that consumes a savings account and the current account balance (a number) and produces a number for the fee currently owed. The fee should be 0 if the balance is at least as large as the minimum, otherwise the fee is as given in the account.

;; fee-owed : savings-account number → number
;; consumes savings account and balance and produces fee owed for the balance

(exam continues next page)

4. (10 points) An airline stores maintenance information on its airplanes using the following data definition:

```
;; An plane is a (make-plane number number boolean string)
(define-struct plane (miles-flown miles-since-checked problems? mechanic))
```

The information stored is how many miles the plane has flown, how many miles it has flown since its last safety check, whether anyone has reported a problem with the plane, and the name of the last mechanic to check the plane.

The airline wants to write a function *service-priority* that consumes a plane and produces a priority ("low", "medium", or "high") for servicing the plane. The priority is high if problems have been reported on the plane or if it has flown more than 100,000 miles since last servicing. Priority is medium if the plane has no problems, but has flown between 50,000 and 100,000 miles since last servicing. Otherwise, the priority is low.

Write a set of test cases for *service-priority* (including expected answers). **Do not write the function—just write the test cases.**

(exam continues next page)

5. (20 points) The votes cast in a recent election are stored as a list of names of candidates (strings). Each name appears in the list once for each vote it received (i.e, if Jessie got 2 votes, then "Jessie" is in the list twice).

   (a) (10 points) Write a function *votes-for* that consumes a name and a list of votes and produces the number of times that the name appears in the list.

   ;; votes-for : string list-of-string → number
   ;; consumes name and list of votes and produces number of times name is in list

   (b) (10 points) Write a function *more-votes?* that consumes two names and a list of votes and produces a boolean indicating whether the first name received more votes than the second.

   ;; more-votes? : string string list-of-string → boolean
   ;; consumes two names and a list of votes and determines whether the first name got more votes than the second.

(end of exam)