NAME:

CS 1101
Exam 3
A-Term 2010

Question 1:	_____	(20)
Question 2:	_____	(20)
Question 3:	_____	(40)
Question 4:	_____	(20)
TOTAL:	_____	(100)

You have 50 minutes to complete this exam. 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 DrRacket constructs:

define define-struct cond else local begin

and the following primitive operators:

empty? cons? cons first rest list append

`+ - * / = < > <= >=`

string=? string-length symbol=? error format
filter map
and or *not*
set!

and the operators introduced by **define-struct** (including the set operators on structures).

You may, of course, use whatever constants are necessary (*empty, true, false,* 0, etc.)

1. (20 points) A trucking company keeps information on each truck in its fleet. The information stored for each truck is the fuel consumption (in miles per gallon), the number of miles on the odometer, and the license plate number of the truck. Here are the data definitions:

```
;; a truck is a (make-truck number number string)
(define-struct truck (mpg odometer license))

;; list-of-truck is either
;; empty, or
;; (cons truck list-of-truck)
```

Using map and/or filter, as appropriate, write a program that satisfies the following contract and purpose:

```
;; higher-mileage-trucks:  list-of-truck number -> list-of-string
;; consumes a list of trucks and a number that represents an odometer reading.
;; The function produces a list of the license plate numbers of those
;; trucks in the list that have an odometer reading greater than the
;; given value.
```

2. (20 points) Here are the data definitions from Problem 1 again:

```
;; a truck is a (make-truck number number string)
(define-struct truck (mpg odometer license))

;; list-of-truck is either
;; empty, or
;; (cons truck list-of-truck)
```

It is desired to write a program that produces the truck in the list with the lowest mpg. We'll use accumulator-style programming to accomplish this.

```
;; least-efficient-truck:  list-of-truck[non-empty] -> truck
;; consumes a non-empty list of trucks and returns the truck from the list that
;; has the lowest mpg.
(define (least-efficient-truck alot)
  (least-efficient-accum (rest alot) (first alot)))
```

Write the function `least-efficient-accum`. Don't forget to provide a contract and purpose. (Hint: The list consumed by `least-efficient-accum` can be empty or non-empty.)

3. (40 points) Faculty in the CS department at a local college are required to serve on department committees.Two faculty members in the department have the following responsibilities:

    Bill is a member of the Undergrad Committee and the Policy Committee.
    Anita is a member of the Facilities Committee and the Policy Committee.

   Here are some data definitions and some examples of data that a DrRacket programmer has written to represent relationships between faculty members and committees:

```
;; a faculty-member (fm) is a struct
;; (make-fm string list-of-committee)
(define-struct fm (name committees))

;; a committee is a struct
;; (make-committee string list-of-fm)
(define-struct committee (name members))

;; a list-of-fm is either
;; empty, or
;; (cons fm list-of-fm)

;; a list-of-committee is either
;; empty, or
;; (cons committee list-of-committee)

(define Bill (make-fm "Bill" empty))
(define Anita (make-fm "Anita" empty))
(define Policy (make-committee "Policy" empty))
(define Undergrad (make-committee "Undergrad" empty))
(define Facilities (make-committee "Facilities" empty))
```

   (a) (20 points) Provide the additional DrRacket statements that will represent the memberships of Bill and Anita on the three department committees.

(b) (20 points) Write a function called `fm-on-committee` that consumes an fm and a committee and produces void. The function adds the given fm as a member of the given committee, and adds the given committee to the list of committees that the fm serves on. Provide a contract, purpose, and EFFECT for your function.

4. (20 points)

A credit card company accepts electronic payments against a person's credit card balance. The balance is kept in a variable:

```
;; balance:  number
;; remembers a credit card balance
(define balance ...)
```

When someone makes an electronic payment, a message is generated that displays the amount paid, the previous balance on the credit card, and the new balance. For example, if someone pays $200 on a $500 balance, the following message is generated:

```
"You paid $200 on a $500 balance.  Your new balance is $300."
```

Write the function `make-payment`:

```
;; make-payment:  number -> string
;; consumes the dollar amount to pay against the credit card balance, and
;; produces a string as described above
;; EFFECT:  the amount of the given payment is subtracted from balance
```