CS 2500 Exam 1 – Fall 2012

Name: _____

Student Id (if known): _____

Section (Shivers/Vona/Razzaq/Ahmed): _____

- Write down the answers in the space provided.
- You may use the usual primitives and expression forms, including those suggested in hints; for everything else, define it.
- The phrase "design this function/program" means that you should apply the design recipe. You are *not* required to provide a template unless the problem specifically asks for one. Be prepared, however, to struggle with the development of function bodies if you choose to skip the template step.
- To save time writing, you may write
  `(sqr 3)` $\rightarrow$ `9`
  instead of
  `(check-expect (sqr 3) 9)`
- Some basic test taking advice: Before you start answering any problems, read *every* problem, so your brain can be thinking about the harder problems in background while you knock off the easy ones.

Problem	Points	/out of
1		/ 6
2		/ 6
3		/ 7
4		/ 17
5		/ 11
6		/ 11
Total		/ 58

Good luck!

Problem 1 You and your homework partner have decided to start a software company writing apps for smartphones. Your first application is a dinner-check calculator. The application consumes the bill for dinner b, the number of diners d, and the tip t (expressed as a fraction, *e.g.*, 0.15, for a fifteen-percent tip). It produces the amount each diner should pay:

6 POINTS

$$\frac{b \times (1 + t)}{d}.$$

Please design this program.

Problem 2 Write the step-by-step computation that would be taken if you ran this program in the DrRacket Stepper. Besides showing the intermediate terms of the computation, label each step as either: 6 POINTS

- **arith:** Primitive "arithmetic" (of any form, not just numeric operations)
- **plug:** Function application—"plugging in"
- **conditional:** A conditional step.

```
(require 2htdp/image)

(define (f n)
  (cond [(> n 5) (sqr (- n 2))]
        [else (+ n 3)]))

(f (* 3 2))
```

Problem 3 Prof. Shivers found the following data definition scribbled on a sheet of paper shoved under his office door: 7 POINTS

```
(define-struct frobboz (a b))
(define-struct frabble (b c))

;;; A Foo is one of:
;;; - a symbol
;;; - (make-frobboz Foo Number)
;;; - (make-frabble Boolean Foo)
```

He's been struggling to write a template for a program that consumes a Foo. Please write it for him.

Problem 4 It's an election year, and politics is on everyone's mind. In the run-up to the November elections, you've been hired as a consultant for a major political candidate. They keep track of campaign donations using the following data definition:

17 POINTS

```
(define-struct donation (donor amount))

;;; A Donation is one of:
;;; - Number
;;; - (make-donation String Number)
;;; Where a simple number represents an anonymous donation.
;;;
;;; A LOD (list of donations) is one of:
;;; - empty
;;; - (cons Donation LOD)
```

The rules of campaign donation declare that a single contributor may only donate $2500 to a given campaign. Furthermore, anonymous donations are limited to $50.

The first task you are given is to design a program, `any-bad-donations?`, that will consume a list of donations and return true if any of the donations on the list are illegal ones.

To make things simpler, you should first design a helper function, `bad-donation?`, that answers this question for a single donation.

Here is a blank page for you to use, if you need it.

Problem 5 Some unscrupulous special-interest groups are attempting to get around the laws on campaign donations (see the previous problem) by making multiple donations. For example, the wealthy but unscrupulous donor `"Vona"` might try to make two donations of $2500 each: 11 POINTS

```
(cons (make-donation "Vona" 2500)
      (cons (make-donation "Vona" 2500)
            ...))
```

Design a program, `donor-total`, that will consume the name of a donor and a LOD, and produce the total amount donated by that donor. For example, it would tell us that the total amount donated by donor `"Vona"`, given the LOD above, is $5000 (which clearly violates donation limits).

Here is a blank page for you to use, if you need it.

Problem 6 Back in September, your roommate built a little web service, "HuskyBook," allowing students to post updates about their lives, link to friends, and comment on each other. Now he's decided to scale it up and offer it as a service to users outside the university. While you're convinced this is an idiotic notion—nobody could be narcissistic enough to enjoy wasting hours of their day, every day, posting the minutiae of their daily lives for the the world to see—he is, after all, your roommate, and you feel obligated to help him out. (He's offered you 15% of the company for your coding help. Whatever.)

11 POINTS

The HuskyBook site permits users to comment on each other's posts (narcissism squared!). Each comment is represented by a structure storing the name of the user who made the comment and the text of the comment. For every original post, we keep a list of these comments.

```
;;; A Comment is a (make-comment String String)
(define-struct comment (name text))

;;; A LOC (list of comments) is one of:
;;; - empty
;;; - (cons Comment LOC)
```

For example, if user Leena made a post on her HuskyBook page saying she'd made a 93 on a major test, an hour or so later, this post might have accumulated this list of comments from her friends:

```
(cons (make-comment "Olin" "Yahoo! You go, girl!")
      (cons (make-comment "Amal"
                          "I failed -- try again in the Spring. :-(")
            (cons (make-comment "Marty" "Big deal. I made a 96.")
                  empty)))
```

HuskyBook allows users to de-friend one another. For example, user Leena might want to de-friend her obnoxious "friend" Marty. It would be nice if, once a user has decided to de-friend someone, all the comments made by the former friend could be made to go away. For example, if Leena de-friended Marty, we'd want to turn the comment list above into

```
(cons (make-comment "Olin" "Yahoo! You go, girl!")
      (cons (make-comment "Amal"
                          "I failed -- try again in the Spring. :-(")
            empty))
```

Design a program, `defriend-comments`, that consumes a string naming a former friend, and a list of comments, and produces the input list without any comments made by the former friend.