CS 2500 Exam 1 – Fall 2014

Name: _____

Husky email id: _____

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

- 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		/ 5
2		/ 5
3		/ 16
4		/ 18
Total		/ 44

Good luck!

Problem 1 You are spending your Summer vacation working at a high-frequency trading firm in Manhattan. Welcome to the one-percenters! 5 POINTS

The firm's partners—when not otherwise occupied lighting cigars with flaming $100 bills and snickering at the "Days of Rage" protesters camped out at the base of your skyscraper—ask you to design a small utility function to compute the capital-gains tax for stock-market transactions.

The "capital-gains tax" is the tax you pay on the profit you make when you sell a stock. Suppose you buy some stock, hold onto it for a while, then sell it at a higher price, for a total profit of p dollars.

- If you held the stock for less than 365 days before selling the stock, the tax is 25% of your profit p. This is "short-term capital gains."
- However, if you held onto the stock for a year or more, you get to pay a lower tax rate: 15% of your profit p. This is "long-term capital gains." (The cheaper tax rate is intended to encourage investors to invest for the long-term in the stocks.)

Please design a function, `cap-gains`, that takes two inputs, `d`, the number of days you held the shares, and `p`, the amount of money you made on the shares, and produces the capital-gains tax you owe on the sale.

Problem 2 If you were to type the following code into the top/definitions window of DrRacket and click the "run" button, what would be output, in the bottom window? 5 POINTS

```
(define-struct quaggle (x y frotz))

(define q1 (make-quaggle  8  100 false))

(define q2 (make-quaggle -5   27 (= 4 (- 10 6))))

(define (fritz q) (not (quaggle-frotz q)))

(define (do-it q)
  (cond [(fritz q) (sqr (quaggle-x q))]
        [else (quaggle-y q)]))

(do-it q2)
```

Problem 3 Your instructors have been working on software to help us automate the management of course grades. We've come up with the following data definition to keep track of students and their homework scores:

16 POINTS

```
(define-struct student (name grades))

;;; A Student is a (make-student String LOG)
;;;
;;; An LOG (list of grades) is one of:
;;; - empty
;;; - (cons Number LOG)
;;;
;;; Interpretation: The grades field records the homework scores --
;;; the first homework assignment is the first element of the list,
;;; and the most recent homework assignment is the last element of the
;;; list.

;;; An LOS (list of students) is one of:
;;; - empty
;;; - (cons Student LOS)
```

Write the template for the Student data definition:

Freshman student Alfred E. Neumann got off to a rocky start, making a 62 on the first homework, but then buckled down and rallied, making an 85 on the second assignment, and has been doing fantastically well since, scoring 96, 94 and 92 on the third, fourth and fifth assignments, respectively. Please give an expression that would produce a `Student` representing Alfred and his homework scores.

We also need a function, `add-grade`, that will take a list of grades (an LOG) and a new homework score (a number), and return a new list of grades—that is, the original list of grades with the new score at the end. Please design this function.

Example:

```
(add-grade (list 62 85 96 94 92) 95)
⟶ (list 62 85 96 94 92 95)
```

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

Problem 4 A train is composed of two kinds of cars: single-decker and double-decker cars (plus one engine, at the front). Single-decker cars can hold up to 40 passengers, while double-decker cars can hold twice as many (that is, 80 passengers). 18 POINTS

Here's a data definition for trains:

```
(define-struct single (passengers front))
(define-struct double (passengers front))

;;; A Train is one of:
;;; - 'engine
;;; - (make-single Number Train)
;;; - (make-double Number Train)
;;;
;;; Interpretation:
;;; - The passengers field records how many passengers have tickets
;;;   on the given car;
;;; - the front field gives the data for the all of the train
;;;   in front of the given car.
```

Here is a simple two-car train:

```
(make-single 22 (make-double 57 'engine))
```

This train begins with the engine at the very front, followed by a double-decker car, with 57 seats sold (and 23 seats free), followed by a single-decker car at the end of the train, with 22 seats sold (and 18 seats free).

As you might expect, double-decker cars are too tall to fit under low bridges. For routing and safety purposes, the transit authority needs a function, `low-train?`, that will consume a `Train`, and return true if all its cars are single-decker, and false if the train has one or more double-decker cars. Please design this function.

The Transit Authority ticket-reservation system also needs a function, `free-seats`, that will consume a `Train` and produce the total number of free seats remaining on the train. (So, for the two-car train given above, it should produce 41). Design this function.