SECOND QUIZ

You have 15 minutes from the start of class to complete this quiz. Read the questions with care; work with deliberate speed. Don’t give us more than we ask for. The usual instructions apply. Good luck!

Problem 1 (5 points)

Complete the following function definition according to the contract and purpose given.

;; course-score: number number number -> number
;; Compute a student’s overall score in a course given three inputs—a score for all assignments, a score for the midterm, and a score for the final. Assume that the inputs and the return value will all be in the range 0 to 100 and compute the overall score using the weights specified below.

(define ASST-WEIGHT 0.20) ; Assignments worth 20%
(define MIDTERM-WEIGHT 0.30) ; Midterm worth 30%
(define FINAL-WEIGHT 0.50) ; Final worth 50%

(define course-score
  (lambda (assignments midterm final)
    (+ (* assignments ASST-WEIGHT) (* midterm MIDTERM-WEIGHT) (* final FINAL-WEIGHT))))

Algebraically equivalent expressions are okay, of course. Unambiguous misspellings of names are okay.

Scoring: 1 point: Addition applied to at least one of the arguments
1 point: At least one argument multiplied by appropriate weight
1 point: Three products (correct or not) summed up
1 point: Correct grouping of each weight with appropriate argument
1 point: Everything else correct (including using defined constants instead of .2/.3/.5)

This rubric may not be perfect. If you find a situation where the rubric seems to give too high or low a score, let me know.

Problem 2 (5 points)

Evaluate each of the following expressions. The function even? is predefined in DrScheme; it returns true if its argument is evenly divisible by 2 (an even number).

(a) (define score-message
    (lambda (score)
      (cond
        ((>= score 90) 'Great)
        ((>= score 80) 'Good)
        ((>= score 70) 'Okay)
        (else 'DoSomethingDifferent)))))

(score-message 75)

(b) (and (even? 24) (= 7 (/ 14 3)))

(c) (or (even? 17) (even? (* 2 (/ 75 25))))
Problem 3 (15 points)

A quiz is a structure (make-quiz problems possible) where problems is the number of problems and possible is the total number of points possible on the quiz (for all problems together).

(a) (2 points) Write a structure definition for the quiz structure described above.

(b) (2 points) Fill in the following blank with a quiz constructor that describes the quiz you’re taking now.

(define second-quiz _____________________________________________________________)

(c) (1 point) Write a Scheme expression that returns the number of points possible on second-quiz.

(d) (5 points) Complete the definition of the following function.

;;; average-points-per-problem: quiz -> number
;;; Given a quiz as input, return the average number of points possible per problem.
(define average-points-per-problem

(e) (5 points) Complete the definition of the following function.

;;; add-a-problem: quiz number -> quiz
;;; Update a quiz by adding one new problem with a specified number of points. That is,
;;; return a quiz that matches the input quiz, but with one more problem and an updated
;;; number of possible points (based on the second input).
(define add-a-problem