COMP 210, Fall 2000
First Exam, September 27, 2000

This exam is conducted under the Rice Honor Code. It is a closed-notes, closed-book exam. Please

1. Print your name legibly on the outside of the test booklet. If you need more space, use the back of the page.
2. Write the honor pledge and sign it. *(5 points)*

The exam has four questions. Where a question uses the results from an earlier question, you should assume that you have the perfect solution to that earlier question. For example, if question one has you create a program **foo**, you can assume, in question two that **foo** exists and that it works as specified, *even if your answer to question one is incomplete or incorrect.*

You may use the following Scheme syntax in your programs

```
define  define-struct  cond  else
```

and the following primitive operations

```
cons, cons?, first, rest, empty?
number?, +, -, *, /, =, <, <=, >, >=, zero?,
add1, sub1, symbol=?, eq?, boolean?, and, or, not,
```

as well as any operations introduced by `define-struct`.

Of course, you may use whatever constants are necessary, including `empty`.

I have tried to give you enough information to answer each question on the exam. In the event that you need additional information, or a clarification, make a reasonable assumption and document it. (Write down, explicitly, any assumptions that you make.)

Question 1. *(20 Points)* Warming up

a. The sales tax in Harris County, Texas is 8.25%. Develop a Scheme program called **sales-tax** that consumes an amount and produces the amount of sales tax levied on that purchase.

   Show all the steps in the design methodology that relate to your program.

Question 1. Warming up

b. In a restaurant, it is customary to provide a gratuity of 15% for the waitstaff. Develop a Scheme program called **restaurant-bill** that computes the total cost of dining at a restaurant. It should consume a number that it interprets as the cost of food and drink. It should return the total cost, including sales tax and tip, for the meal.

   Show all the steps in the design methodology that relate to your program.

   (*Consultant's advice*: Sales tax does not apply to gratuities.)

Question 1. Warming up

Hand evaluate the following Scheme expressions. Show all the steps.

c. `(= (+ (* 3 3) (* 4 4)) (* 5 5))`

d. `(zero? (- (* 3 4) (+ 5 6)))`

Background for questions two, three, and four: You are responsible for creating a database to hold the grade information for COMP 210. Since this is COMP 210, the code must be written in Scheme. We have already done the data analysis for you.

For each student, you must store a name, a student identification number, the number of points on the homeworks & exams, as well as the number of extra credit points. This leads to the following compound object (in Scheme).

```
;; a student is
;; (make-student  name id  hwks  ex1 ex2 ex3  ec)
;; where name is a symbol and id, hwks, ex1, ex2, ex3, and ec are numbers
;;        hwks will hold the total score for the homework, on a scale of 1 to 100
;;        ex1 will hold the score for the first exam, on a scale of 1 to 100
;;        ex2 will hold the score for the second exam, on a scale of 1 to 100
;;        ex3 will hold the score for the third exam, on a scale of 1 to 100
;;        ec   will hold any extra credit points
(define-struct  student (name id hwks ex1 ex2 ex3 ec))
```

To make the arithmetic easier, all of the scores (hwks, ex1, ex2, and ex3) are computed on a 100-point basis. That is, a perfect set of homeworks would garner 100 points, as would a perfect exam.

Of course, COMP 210 does not have a single student; it has over 100 students. Thus, you need to organize the information on the full set of students. Your background suggests that a list would be a natural way to do this. This leads to the following data definition:

```
;; a list-of-students is either
;;  –- empty, or
;;  –- (cons f r), where f is a student and r is a list-of-students
;; [Since we are using cons, we don't have a define-struct here]
```

You may assume that the list of students always has one or more entry. A class with no students is never taught.

Each question will specify which parts of the design methodology you must show.

Example Data:

```
(define S1 (make-student 'Jane 90 80 100 80 2))
(define S2 (make-student 'Frank 82 95 88 92 1))
```

Some example students

```
(define L1
  (cons (make-student 'Jane 98 96 100 80 2)
     (cons (make-student 'Frank 82 95 88 92 1)
        (cons (make-student 'Pat 87 73 85 99 4)
           (cons (make-student 'Mary 90 91 89 95 2)
              (cons (make-student 'Bubba 50 79 68 61 0)
                    empty))))))
```

A list-of-students

Question 2. *(25 Points)* Manipulating compound objects

Develop the program **score** that consumes a student and produces the student's score, on a scale of 1 to 100. Recall that homework counts for 50% of the grade, the first exam counts for 10% of the grade, and the other two exams each count for 20% of the grade. Extra credit is added in, at face value, after the rest of the score has been computed.

No template is needed for this question. Show the final code and the results you expect from applying your code to the example student **S1** given on page 5.

Question 3. *(25 Points)* Manipulating lists of data

Develop the program **best-score**. It should consume a list-of-students and return a number. **best-score** should return the highest score achieved by any student in the class. Grading is as defined in the write-up for Question 2.

Show the template for your program. Show the final code and the result of invoking your program on the list **L1** given on page 5.

Question 4. *(25 Points)* Programs that return lists

Develop a program **above-threshold** that works on the COMP 210 student data structure used in questions two and three. It should consume a list-of-students and a number. It should produce a list containing all of the students whose score, as defined in question two, is greater than the number passed as an argument.

You can re-use the template from the previous question. Show your final code, along with the result of invoking your program on the example data **L1** shown on page 5, with a threshold value (the second argument) of 80.

Extra Credit *(10 Points)* Recursion over the natural numbers
Develop a program **natnum-exp** that consumes two natural numbers, **m** and **n**, and produces the quantity $\mathbf{m^n}$. (Recall that, for COMP 210, we have defined the natural numbers as the counting numbers, along with zero.) Your program is limited to using addition, subtraction, and multiplication of natural numbers.

Show your data definition, template, and final code.