CS2135, C03

Final Exam

Name:

Problem	Points	Score
1	25	
2	10	
3	30	
4	35	
	Total	

You have 50 minutes to complete the problems on the following pages. There should be sufficient space provided for your answers. You do not need to show test cases or examples of data models, but you may develop them if they will help you write the programs.

Your programs may contain only the following Scheme syntax:

define define-struct define-syntax cond else lambda let letrec begin set!

and the following primitive operations:

empty? cons? cons first rest car cdr list map filter length
member append
number? $+ - * / = < > <= >=$ *zero? min max*
symbol? symbol=? equal? eq?
boolean? **and or** *not*
printf read

and the functions introduced by **define-struct**.

You may, of course, use whatever constants are necessary.

---

1. (25 points) We have been using **cond** all semester to write conditionals. Most languages include an if-statement for conditionals with only two cases. The expression

   (**if** *test then-stmt else-stmt*)

   should run the test, performing the then-stmt if the test returns true, otherwise performing the else-stmt. In this question, we want to implement such a construct.

   (a) (10 points) Should **if** be implemented as a function, as a macro, or as either? Justify your answer.

   (b) (10 points) Implement **if**. Your answer should be consistent with your answer to part(a).

   (c) (5 points) Assume you had augmented your implementation of **if** with a check that the test expression evaluates to a boolean before you execute the then or else part (assume that Scheme **cond** doesn't perform this check for you). Assume you typed the following code into the definitions window:

   ```
   (define (myfunc x)
     (if 3 x (+ x 5)))
   ```

   Would the error (that the test 3 isn't a boolean) get detected when you hit execute, when you enter a call like (*myfunc* 7) from the prompt, neither, or both? (Answer this based on how you implemented **if** in part (b).)

(exam continues next page)

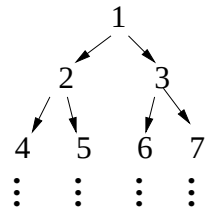
2. (10 points) Rewrite the following code so that all calls to scripts are in script position.

```
(define (small-program)
  (cond [(symbol=? 'yes (prompt-script "Last final?")) "Happy break!"]
        [else  "Hang in there!"]
```

(Assume that *prompt-script* is already defined (to print its argument string and return the result of reading input from the user) and that *prompt-script/web* already exists and takes both a prompt string and a destination/action argument.)

(exam continues next page)

3. (30 points) In class, we wrote programs to generate infinite lists (also called *streams*). We could also write programs to generate infinite trees. Consider the following infinite tree of numbers:



   (a) (20 points) Write a program *make-stream-tree* that takes a number (the root of the tree) and returns an infinite tree with the structure shown above. Be sure to include the definitions of any **define-struct**s that you create or use for this problem. (Notice the relationship between a node and the number in its left and right children – the left child doubles the parent node value, which the right child doubles and adds one to the parent value.)

(exam continues next page)

(b) (10 points) Assume we made the following definition using your *make-stream-tree* program:

(**define** *T* (*make-stream-tree* 1))

Write an expression using *T* that would evaluate to 5 (i.e., that would extract the 5 from *T*).

(exam continues next page)

4. (35 points) A company wants to design software to help teachers create and administer quizzes. The software should have the following capabilities:

   - Questions can be multiple choice or free-response, covering any academic subject.
   - Quizzes can have any number of questions.
   - The quiz software does not need to determine correct answers to questions – the teachers will provide the correct answers.
   - The questions posed to a student can depend on the student's answers to the previous question: for example, if a student misses a question, the teacher may wish to ask additional questions on the same topic, but if the student answers correctly, the teacher may want to ask questions on a new topic.
   - After giving a quiz, the software will report how many questions the student got correct.

   The following figure shows two examples of mathematics quizzes that could be generated from the same quiz program. The example on the left might represent a quiz where the student got all questions right, while the example on the right shows a case where another student taking the same quiz missed the second question and got additional questions on fractions.

1. What is 40% of 10? (a) 4 (b) 400 (c) 5 2. Compute 4/5 * 6/3. 3. Which is larger: 2/3 or 1/4? (a) 2/3 (b) 1/4 You got 3 right!

1. What is 40% of 10? (a) 4 (b) 400 (c) 5 2. Compute 4/5 * 6/3. 3. Compute 1/2 - 1/4 4. Compute 3/8 + 2/3 You got 2 right!

   (a) (20 points) Develop data definitions for quiz programs. (Do **not** include the other data definitions that the interpreter needs to execute quiz programs).

(exam continues next page)

(b) (10 points) Write the example for the sample quiz program shown above in your definitions (you may abbreviate the question strings as long as we can tell which one is which in your example).

(c) (5 points) Does your language expect the teacher (programmer) to specify the question numbers? Why or why not?

(end of exam)