CSc 135 – Spring 2017 - Programming Languages
Homework 2

due Sunday April 9 at 11:59:00 pm

Write the following Scheme functions:

1. **combine3** takes three positive integers and return an integer which is made of the last two digits of the first integer, the first and last digits of the second integer and the first two digits of the last integer. If any of the integers has less than two digits return 0. For example:

   (combine 4 124 80534) will return 0
   (combine3 247939 968634 40348726) will return the integer 399440 and
   (combine3 5678 456 9264) will return 784692.

   If any of your numbers has less than two digits, return 0. Note that you probably need to define several auxiliary functions.

2. **inc-by-n** takes an integer x and a digit d and returns an integer where each digit in x has been incremented by n. If the result of the incrementation would exceed 9, the result mod 10 will be used. For example:

   (inc-by-n 53462 3) will return 86795 and
   (inc-by-n 682901 4) will return 026345

   Note that if the first digit of the answer is "0" it will not print.

3. **countnumber** takes a list of integers L (with possible nested sublists) and an integer n and count the number of instances of n in the list. For example:

   (countnumber '(5 7 34 28 5 72 (5)) 5) returns 3 and
   (countnumber '(3 (67 34 (29 56) 23) 56 (5 (45 34 56) 0)) 34) returns 2

4. **mapeven** takes a function and a list and return a list similar to the list argument where the function has been applied to every other element starting with the second one. For example:

   (mapeven square '(3 6 22 9 5 7 10 15)) will return (3 36 22 81 5 49 10 225)
   (mapeven even? '(3 6 22 9 5 7 10 15)) will return (3 #t 22 #f 5 #f 10 #f)
   (mapeven list? '(2 (3 4) 6 10 (88 23) (5))) will return (2 #t 6 #f (88 23) #t)

   Note that the function applied only to the top elements of the list. Make sure that your function works with any relevant function, and that it works with lists of even and odd length.

5. **addTheSquares** takes a list of integer and returns the sum of all the squares. For example:
   (addTheSquares '(6 3 9 2 6)) will return 166 (i.e. 36 + 9 + 81 + 4 + 36)

   Then write a tail recursive version of that same function.

6. **makeInsertInList** takes an argument x and returns a function which will have a list argument and will insert x after each element of the list.

   For example, if "makeInsertInList" was called as follows:
   (makeInsertInList 5)

   a function would be produced that takes as input a list and returns a list where 5 would have been inserted after each element of the list.

   For example, if "makeInsertInList" was used as follows:

   (define F (makeInsertInList 5)

   then the produced function F would behave as follows:

   (F '(2 3)) would return (2 5 3 5)
   (F '((4 8) 2 9)) would return ((4 8) 5 2 5 9 5)

   Your task is just to write makeInsertInList, not "F".
   Of course, makeInsertInList should work for ANY type of input, including lists. Hence if "makeInsertInList" was used as follows:

   (define G (makeInsertInList '(a b))

   (G '(x y z)) would return (x (a b) y (a b) z (a b))

   Note that the insertion only takes place at the top level.

Generally speaking, you can only use the functions defined in the notes or in exercises. In particular, you cannot use built-in functions which would make the problems trivial. However, modulo and quotient may be used as needed. If you have any doubt, ask.

==========

Submission instructions:

- Place all of your Scheme functions in ONE file.
- Make sure your functions work in Dr-Racket "Advanced Student" before submitting the file.
- Include comments in your code clearly delineating each of the problems.