CS 326 – Project #11

Purpose: Become familiar with the functional programming language, Scheme/Racket.
Points: 150

Assignment:
Write the following Scheme/Racket functions. *Note, you should not use any built-in functions.*

Part A: (5 pts each)

iseven? = return a true if the argument, an atom, is even. Return false otherwise.
[i.e., (iseven? 21) = false]
Note, the modulo function, (modulo int1int2), may be useful.

isodd? = return a true if the argument, an atom, is odd. Return false otherwise.
[i.e., (isodd? 21) = true]

Part B: (10 pts)

prod = multiply all elements in a list, including all sublists.
[i.e., (product '(1 2 3) = 6]
[i.e., (product '(2 2 (2 2 (2 2 2) 2))) = 256]

lstmul = return list by multiplying each element of a list, including sublists, by n.
[i.e., (lstmul 2 '(2 3 4)) = (list 4 6 8)]
[i.e., (lstmul 2 '(1 2 (3 4))) = (list 2 4 (list 6 8))]

sumlist = sum all atoms in a list, including all sublists.
[i.e., (sumlist '(2 3 4) = 9]
[i.e., (sumlist '(1 2 (2 3) 4)) = 12]

len = returns the length of a list, including all sublists
[i.e., (len '(1 2 3 4 5 6 7 8)) = 8]
[i.e., (len '(7 9 (1 4) 2)) = 5]

average = returns average of a list, including sublists (use the previous functions).
[i.e., (average '(4 5 6 7 8)) = 6]
[i.e., (average '(1 (2) 3 (4 5 (6 7) 8 (9) 10))) = 5.5]

flatten = flatten a list.
[i.e., (flatten '(1 2 (3 4 (5 6)))) = (list 1 2 3 4 5 6)]

rvlst = reverse all atoms in a list, including sublists
[i.e., (rvlst '(2 3 4 5)) = (list 5 4 3 2)]
[i.e., (rvlst '(1 2 (3 4) (5 6) 7 8)) = (list 8 7 (list 6 5) (list 4 3) 2 1)]

rmFirstOcc = remove first occurrence of an item from a list, including sublists
[i.e., (rmFirstOcc 3 '(2 3 4 3)) = (list 2 4 3)]
[i.e., (rmFirstOcc 5 '(1 2 (3 4 (5) 6))) = (list 1 2 (list 3 4 empty 6))]

minimum = find the smallest item in a list, including sublists
[i.e., (least '(5 2 7)) = 2]
[i.e., (minimum '(7 5 (6 1))) = 1]

Part C:
Write the following Scheme/Racket function (15 points):

insertion-sort = sort a list using the insertion sort.
[i.e., (insertion-sort '(9 1 8 2 7 3 6 4 5)) = (1 2 3 4 5 6 7 8 9)]

Note, need only handle flat lists.

Write the following Scheme/Racket program (15 points):

sqr-and-cube = Read a number and display the square and cube of that number. You do not need to handle invalid/incorrect input. The program should display appropriate headers and output as follows:

```
--------------------------------------------------
Square and Cube Program.
Give me a number, and I'll compute its square and cube.

Number: 10

The square of 10 is 100.
The cube of 10 is 1000.
```

Write the following Scheme/Racket program (20 points):

liststats = Read a list from the user and display the length, average, minimum, sum, product, unsorted list, and sorted list. You do not need to handle invalid/incorrect input. Should use many of the previously defined functions (from above as needed). You may write additional functions if desired. Program should display appropriate headers and output as follows:

```
List: (9 8 1 2 6 7 5 3 4)

------------------------
List Stats Program.

length: 9
average: 5
minimum: 1
sum: 45
product: 362880

Unsorted list:
(9 8 1 2 6 7 5 3 4)
Sorted list:
(1 2 3 4 5 6 7 8 9)
```

Note, need only handle flat lists.

Note, must use recursive solutions (not iterative). All functions should be in a single source file. Refer to the class web page for information of test data to use when testing and submitting.

Submission:

1) Submit a copy of the Scheme/Racket program (definitions window).
2) Submit a copy of the results (interactions window).
   Note, a set of test calls for the functions will be provided.