Scheme Language Assignment

Purpose
To attain familiarity with the functional programming language Scheme; to analyze its runtime efficiency; and to research its history, design goals, and design decisions.

Note
Use the DrRacket IDE (https://racket-lang.org/download/) for this assignment. You can also use the Racket documentation (https://docs.racket-lang.org/guide/). Racket is a language extremely close to Scheme. We are going to be using Scheme but the Racket documentation should still be helpful due to the major overlap between languages. At the top of your code, make sure to use #lang scheme

Directions

1. Week 1 – Get familiar with Scheme (30 pts)
    a. Design a simple struct, a variable, and a set of functions to operate on these in Scheme.
    b. Make sure to set the language to scheme ("#lang scheme") at the top
    c. Struct
        i. Create a structure using the define-struct function that contains an id, description and price for a single item that could be sold at a grocery store.
    d. Variable
        i. Define a list of struct objects, each with their own id (must start at 1 and count up consecutively), description (simple string stating what it is, e.g. "milk") and price (a positive integer or floating number).
        ii. You must have at least 7 objects in the list.
        iii. Give a name for this list as you will need to refer to it.
            1. I refer to it as the master list in the text below.
    e. Functions
        i. lookup
            1. Takes an id and a list (which initially should be the list you defined above) as parameters and returns the object from the list that matches the id (hint: use car and cdr functions)
        ii. getdesc
            1. Takes an id as parameter
            2. Uses the lookup function to find the object with that id and returns the description of the object
        iii. getprice
            1. Takes an id as parameter
            2. Uses the lookup function to find the object with that id and returns the price of the object

iv. subtotal
    1. Takes a list (of id's) as a parameter
    2. Goes through the parameter list and sums up the prices of all objects matching the id's. Uses getprice on each id in the list
v. total
    1. Takes a list (of id's) as a parameter
    2. Use the format function to display the subtotal (found by calling the subtotal function you defined) as well as the total (found by using the subtotal and multiplying it by 11%).
    3. Use Scheme's let expression to avoid evaluating the subtotal twice.
        a. "Let" is a common tool used in functional languages to define a new scope and any local variables for use inside that scope.
vi. getlist
    1. Takes no parameters
    2. Returns a list of lists where each inner list is the id, description and price of each object in the master list
    3. Since this doesn't take any parameters, you may want to define a helper function that takes a list as a parameter and have getlist call that helper function passing it the master list. This way you can use recursion with the helper function.

f. Test code
    i. Write test code for each function
    ii. This lets you make sure everything is working correctly.
g. Put comments to your code properly to describe the functionality of each function

2. <u>Week 2 – Study its runtime efficiency (30 pts)</u>
    a. Fortunately Scheme is really easy to time.
        i. Just use the time function
    b. However the hard part is finding enough work so the time isn't 0 and we can actually analyze it
    c. Therefore you need to create a decently complicated function:
    d. Create a function called gettiminglist
        i. This function takes the desired length of the list as a parameter
        ii. Its job is to create a list of random whole number values (with the amount of values specified by the parameter) that are between 1 and the length of your master list (inclusively).
            1. For example, if there are 7 items in your master list with id's from 1 to 7 inclusively and you call this function with a parameter of 10, one possible outcome could be: (1 2 5 3 3 4 6 1 4 7)
        iii. Use the length function to get the length of your list, do not hard-coding it
        iv. You will have to add one to all the id's in your list as without this, they will go from 0 up to but not including the length of your master list.
    e. Now create a helper function called runtime
        i. This function will merely call the gettiminglist function passing it an argument of 20000 (that's 20 thousand) and simply passing the returned list to the subtotal function. The subtotal function can then lookup each id in the list and therefore we can time the multiple calls to the lookup function
        ii. Call Scheme's time function on the subtotal call as part of the definition of this runtime function

f. Add code line (runtime) five times. It should give you different times Scheme calculates.

g. Run the code and create a table with all the timings including the averages. Put this in a separate document called "timings" and include it with your submission. Your table should resemble the following (with actual times for the # placeholders):

Run	CPU TIME	REAL TIME	GC TIME
1	#	#	#
2	#	#	#
3	#	#	#
4	#	#	#
5	#	#	#
Average	#	#	#

3. Week 3 – Study its history and design (30 pts)
    a. Look up the history of the Scheme programming language
    b. Write a report including the following:
        i. Who invented Scheme
        ii. What was it invented for
        iii. What were the design goals of the language
        iv. Is it considered to be a success
    c. After this, do some further research into the Scheme language and add to your report some well-informed, well researched opinions of the language with respect to the following language attributes:
        i. Programmer efficiency
        ii. Writability
        iii. Regularity
            1. with regards to any of the 3 forms we've learned about
        iv. Security
        v. Extensibility
    d. All opinions should be backed by references. You can also state your opinions based on your own experience using the language during weeks 1 and 2, but you need to state from which functions or tasks you concluded your opinions
    e. At the end of your report, write a separate session and place all your references. (Please refer to wiki page https://en.wikipedia.org/wiki/Scheme_(programming_language) if you don't know how to write references)
    f. Either Times New Roman, Arial, or Calibri fonts allowed
    g. 10, 11 or 12 point font
    h. Single or double column
    i. Min of 1 full page
    j. Max of 3 pages
    k. Clearly indicate which aspect of language design you are talking about each time you move on to a new design aspect
    l. i.e. If you are talking about programmer efficiency, please make it clear that you are talking about programmer efficiency. When you then start talking about writability, again make it clear that you are now talking about writability, and so on.

Submission (10 pts)

- Put your original program (the entire project) from week 1 in a folder called "week 1".
- Put your altered timing program (the entire project) from week 2 in a folder called "week 2".
- Put your language design report (in pdf format) from week 3 into a folder called "week 3".
- Now put the "week 1", "week 2" and "week 3" folders into another folder with your first and last name as the name of that folder
- Finally, zip up the folder with your name on it and submit it on Moodle.
- Points will be taken off if these submission directions aren't followed