CSE 240 Spring 2019 Homework 11: Scheme (50 points)

Due Saturday, April 6, 2019, at 11:59PM, plus a 24-hour grace period

---

Introduction

The aim of this assignment is to make sure that you understand and are familiar with the concepts covered in the lectures. By the end of the assignment, you should have

- understood the concepts of functional programming paradigm.
- written functional programs in Dr. Racket Scheme.
- understood names, forms, and procedures in functional programming paradigm.
- understood the use of let-form and unnamed procedure.
- applied recursion to solve complex problems.

Reading: Text Chapter 4. Read course notes (slides). This is a complete new language, and you need to spend more time to read and to learn programming in this paradigm.

Preparation: Complete the multiple choice questions in the textbook exercise section. The answer keys can be found in the course Web site. These exercises can help you prepare for your weekly quiz and the exam. You are encouraged to read the other exercise questions and make sure you understand these questions in the textbook exercise section, which can help you better understand what materials are expected to understand after the lectures and homework on each chapter.

You are expected to do the majority of the assignment outside the class meetings. Should you need assistance, or have questions about the assignment, please contact the instructor or the TA during their office hours.

You are encouraged to ask and answer questions on the course discussion board. However, **do not share your answers and code** in the course discussion board.

Practice Exercises (no submission required)

1. **Tutorial: Getting Started with DrRacket**
   1.1. To complete this assignment, you will need to download and install a copy of Dr Racket (http://racket-lang.org/download/) on your local PC.
   1.2 Start the program DrRacket.
   1.3 Choose the "**R5RS**" from the language menu:

DrRacket menu: language →choose language →**Other Languages – R5RS.**

1.4. Enter your programs/forms in the upper window of DrRacket and click on the "run" button to execute your programs/forms, e.g., enter:

```
(write "hello world")

(newline)

(write (+ (* 3 8) 10))

(- 20 5)

(write (read))
```

```
(display "hello world")

 (newline)

 (display (+ (* 3 8) 10))

 (- 20 5)


(display (read)) ; input a number
from keyboard
```

Click on **run**, the following results should appear in the lower window:

```
hello world
34
15
```

2. Use DrRacket to calculate the following expressions/forms.

(1) (3 + (5 + (7 + (9 + (11 + 13))))).
(2) (((((3 + 5) + 7) + 9) + 11) + 13)
(3) ((2+4)+(3+5)+(6+8))
(4) (2 + 4 + 6 + 8 + 10 + 12)
(5) (2 + 3 * 5 + 4 * 6 + 7)
(6) 125187
(7) Input two integers: (* (read) (read))

3. Write Scheme programs/forms to

(1) find the second element of the list '(2 4 6 8 10 12). Your form should work for any list containing two or more elements.
(2) find the last element of the list '(2 4 6 8 10 12). Your form only needs to work for lists of six elements.
(3) merge the two lists '(1 2 3 4) and '(5 7 9) into a single list '(1 2 3 4 5 7 9)
(4) obtain the length of the list '(a b x y 10 12)
(5) check whether '(+ 2 4) is a symbol
(6) check whether '+ is a member of the list '(+ 3 4 6)
(7) check whether "+", '(+ 3 5), "(* 4 6)" are strings
(8) check whether (* 3 5), '(/ 3 7), (1 2 3 4), "(+ 2 8) and "(1 2 3)" are strings

4. Write a function to find the factorial of a number.

4.1 Follow a recursive procedure to obtain (fact n).

1.2 Use comments to indicate in the code the four steps of the fantastic approach: [4 points]

1) The size-n problem
2) All the stopping conditions and the return values.
3) All the size-m problems
4) The lines of code that construct the size-n solution from the size-m solutions.

Test cases:

(fact 5) →120

(fact 8) →40320

Test much bigger input number

Programming Exercise (50 points)

In this assignment, you will be learning Scheme through the use of Dr. Racket. We would like to start with some basic concepts; trying to understand prefix notation and the use procedure in Scheme. You will also implement nested procedures and recursive procedures. You may only use the procedures shown in the text and slides - not any of the additional library procedures in Scheme. Solve the following questions in the hw11.rkt template provided.

1. Using Dr. Racket to compute the following expressions. [5 points]

   1.1 9 - 2 + 5
   1.2 5 * (6 + 12 + 5) – 25
   1.3 7 * ((5 - (1 * 3)) + (2 * 4))
   1.4 3 * (4 + (((6 * 6) + (6 * 6)) / (10 + 2)))
   1.5 ((((((4 + 6) * (6 + 4)) / 2) / 2) – 5) / 2) + ((((4 * 5) + (5 * 4)) / 2) + (4 * 5))

2. Define a procedure "Add" that takes parameters and returns the sum of them.

   [5 points]

```
> (Add  20  30)
50
```

3. Define a recursive procedure called "Square" that will compute the square amount of a value by calling the "Add" procedure defined in the previous question. [Total 10 points]

   3.1 You must use the Add procedure defined above. [5 points]
   3.2 You will need to account for negative values as well. [3 points]
   3.3 Use comments to indicate which code represents the four steps of the fantastic abstract approach, respectively: [2 points]
   ; The code for size-n problem is:
   ; The code for stopping condition and its return value are:
   ; The code for size-m problem is:
   ; The code for constructing size-n problem is:

Hint: This will require a conditional and possibly the (abs x) procedure. You may not use multiplication in this procedure definition. Note, you can use this formula to identify your size-(n-1) problem: $n^2 = 1+3+5+ \dots + (n+n-1)$.

```
 > (Square  5)
    25
 > (Square  -7)
    49
```

4. Define a procedure "ReadForSquare" to read a value for the Square procedure defined above. This procedure takes no parameters and will take an input value from keyboard and pass to the Square procedure defined above. [5 points]

```
> (ReadForSquare)  ; enter 5 from keyboard
25
```

5. Define a procedure "DiffSquares" that will compute the difference between the square values of 2 inputs. You must implement the ReadForSquare procedure defined above. This procedure should never return a negative value, as it should return the absolute value. [5 points]

```
> (DiffSquares)
5       ; input
10      ; input
75      ; output
```

6. Implement a procedure called (AddLet x y z), which will return x+y+z. In this procedure, you must implement let-form to bind values. You may name the parameters whatever you like. [5 points]

```
> (AddLet 60 40 5)
105
```

7. In this question, you must use an unnamed procedure to add three numbers. Your code should look like the following: [5 points]
((lambda (parameter list) (*add your code here*) argument list)

   Test case:
   arguments 60 40 5
   return 105

8. Create a recursive procedure called (sumOdds lst), where lst is a list of numbers. The procedure should return the sum of all odd numbers on the list. When the list is empty, the result should be 0.

   8.1) Use a <u>named</u> procedure called (sumOdds lst) to implement task. [5 points]
   Test case: (sumOdds '(1 3 4 5 6 -7)) → 2

   8.2) Use an <u>unnamed</u> procedure to implement task.
   ((lambda (lst) (*add your code here*) '(1 3 4 5 6 -7)) where the list '(1 3 4 5 6 -7) is the argument. [5 points]

Grading of Programming Assignment

The grader will grade your program following these steps:

(1) Compile the code. If it does not compile, 20% of the points given will be deducted. For example, if there are 20 points possible, you will earn 16 points if the program fails to compile.

(2) The grader will read your program and give points based on the points allocated to each component, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

(3) Enter comments for important steps of procedures.

What to Submit?

You are required to submit your solutions in a compressed format (.zip). Zip all files into a single zip file. Make sure your compressed file is labeled correctly - lastname_firstname11.zip.

For this home assignment, the compressed file MUST contain the following:

hw11.rkt	(completed Scheme program)

No other files should be in the compressed folder.
If multiple submissions are made, the most recent submission will be graded, even if the assignment is submitted late.

Where to Submit?

All submissions must be electronically submitted to the respected homework link in the course web page where you downloaded the assignment.

Late submission deduction policy

- No penalty for late submissions that are received within 24 hours after the deadline
- 10% grade deduction for every day it is late after the grace period;
- No late submission after Tuesday at 11:59PM.

Academic Integrity and Honor Code.

You are encouraged to cooperate in study group on learning the course materials. However, you may not cooperate on preparing the individual assignments. Anything that you turn in must be your own work: You must write up your own solution with your own understanding. If you use an idea that is found in a book or from other sources, or that was developed by someone else or jointly with some group, make sure you acknowledge the source and/or the names of the persons in the write-up for each problem. When you help your peers, you should never show your work to them. All assignment questions must be asked in the course discussion board. Asking assignment questions or making your assignment available in the public websites before the assignment due will be considered cheating.

The instructor and the TA will **CAREFULLY** check any possible proliferation or plagiarism. We will use the document/program comparison tools like MOSS (Measure Of Software Similarity: http://moss.stanford.edu/) to check any assignment that you submitted for grading. The Ira A. Fulton Schools of Engineering expect all students to adhere to ASU's policy on Academic Dishonesty. These policies can be found in the Code of Student Conduct:

http://www.asu.edu/studentaffairs/studentlife/judicial/academic_integrity.htm

ALL cases of cheating or plagiarism will be handed to the Dean's office. Penalties include a failing grade in the class, a note on your official transcript that shows you were punished for cheating, suspension, expulsion and revocation of already awarded degrees.