CSE 240 Homework 8

50 points

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 and procedures in functional programming paradigm.

Reading:

Text Chapter 4 and course notes (slides). This is a complete new language, and you need to spent more time to read and to program.

Practice Exercises (*non-graded, no submission required*)

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
```

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))
```

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

Programming Exercise (Graded)

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 under 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.

A) [10% Spec] Using Dr. Racket to compute the following expressions.

```
1) 3 + 5 - 7
2) 2 * ( 8 + 5 + 4 ) - 25
3) 10 - ( ( 3 * 5 ) + ( 2 + ( 0 * 5 ) ) )
4) 5 * ( 4 + ( ( ( 10 + 10 ) + ( 5 * 8 ) ) / ( 10 + 2 ) ) )
5) ( ( ( ( ( ( 3 + 5 ) * ( 6 + 4 ) ) / 2 ) / 2 ) - 5 ) / 3) + ( ( ( ( 2 * 10 ) +
   ( 5 * 4 ) ) / 2 ) + ( 4 * 5 ) )
```

B) [10% Spec] Bind (define) each value in 1.5 above to its' English text and then change the expression using the defined names. For example, the values in 1.1 should be replaced with names three, five, and seven, and the correct corresponding expression is (eight + two - ten).

C) [10% Spec] Define a procedure "Subtract" that takes parameters and returns the difference of them You can use the built-in "-" to define your Subtract procedure.

```
> (Subtract 120 50)
70
```

D) [20% Spec] Define a **recursive** procedure called "IntDivide" that will compute the quotient of x divided by y. You must implement IntDivide procedure by a sequence of Subtract procedures.

1) You must use the Subtract procedure defined above.
2) You will need to account for negative values as well.

Hint: This will require a conditional and possibly the (abs x) procedure. You may not use the built-in division or quotient operators in this procedure definition.

```
> (IntDivide 8 3)
2
```

E) [10% Spec] Define a procedure “ReadForIntDivide” to read the two input values for the IntDivide procedure defined in the previous. This procedure takes no parameters and will pass an input value to the Square procedure.

```
> (ReadForIntDivide)
-25
4
-6
```

F) [10% Spec] Define a **recursive** procedure called “Multiply” that will compute the product of x times y. You must implement Multiply procedure by a sequence of additions.

1) You can use the built-in + operation.
2) You will need to account for negative values as well.

```
> (Multiply 8 3)
24
```

G) [10% Spec] Define a procedure (DiffDivide x y) that will compute the following expression: x - (x/y)*y. For example, if x = 8 and y = 3, then, 8 – (8/3)*3 = 2

You must use Subtract, IntDivide, and Multiply defined in the previous questions.

```
> (DiffDivide 8 3)
2
```

H) [10% Spec] Write a function/series of functions called explode that will take a string as a parameter and return a list containing the characters of the string. (Hint: think helper function)

```
> (explode “Hello”)
‘(“H” “e” “l” “l” “o”)
```

I) [10% Spec] Create a function/series of functions called implode that will take a list as a parameter and return a string containing the list elements. (Hint: think helper function) (Double Hint: this is very much a re-write of H)

```
> (implode ‘(“H” “e” “l” “l” “o”))
“Hello”
```

Grading of Programming Assignment

The TA will grade your program following these steps:

(1) Compile the code. If it does not compile you will receive a U on the **Specifications** in the Rubric

(2) The TA 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.

Rubric:

	Levels of Achievement						
Criteria	A	B	C	D	E	U	F
Specifications Weight 50.00%	100 % The program works and meets all of the specifications.	85 % The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	75 % The program produces mostly correct results but does not display them correctly and/or missing some specifications	65 % The program produces partially correct results, display problems and/or missing specifications	35 % Program compiles and runs and attempts specifications, but several problems exist	20 % Code does not compile and run. Produces excessive incorrect results	0 % Code does not compile. Barely an attempt was made at specifications.
Code Quality Weight 20.00%	100 % Code is written clearly	85 % Code readability is less	75 % The code is readable only by someone who knows what it is supposed to be doing.	65 % Code is using single letter variables, poorly organized	35 % The code is poorly organized and very difficult to read.	20 % Code uses excessive single letter identifiers. Excessively poorly organized.	0 % Code is incomprehensible
Documentation Weight 15.00%	100 % Code is very well commented	85 % Commenting is simple but solid	75 % Commenting is severely lacking	65 % Bare minimum commenting	35 % Comments are poor	20 % Only the header comment exists identifying the student.	0 % Non existent
Efficiency Weight 15.00%	100 % The code is extremely efficient without sacrificing readability and understanding.	85 % The code is fairly efficient without sacrificing readability and understanding.	75 % The code is brute force but concise.	65 % The code is brute force and unnecessarily long.	35 % The code is huge and appears to be patched together.	20 % The code has created very poor runtimes for much simpler faster algorithms.	0 % Code is incomprehensible

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:

```
hw6.rkt (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.

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.