# CSC 3310 Concepts of Programming Languages

Fall 2018

| | |
|---|---|
| Assignment Name | Scheme Programming Assignment |
| Due Date | November 21st, 2018 |
| Delivery Method | Through Canvas Only |
| Points | 10% of final grade |

## Motivation

Around 1930's one mathematical model of computation was created called the Lambda Calculus, with time it was proven to be equivalent to the Turing Machine model, meaning that both models have the same computational capability. Functional programming is an elaboration based on Lambda Calculus, where programming is declarative as opposed to imperative as most programming languages students have been exposed so far. At a point in the development of Computer Science, functional programming was thought to be applicable to solve AI problems. Besides being an academic type of programming, industry also uses functional programming, for instance Erlang is used in the telecommunications industry. There are modern functional programming languages like Scala and F#.

In this assignment, students will develop simple programs in Scheme in order to practice this language and expose them to a completely different paradigm of programming. Up to this moment, most of the students have been working and coding with imperative and object oriented languages, and a declarative language should pose a new challenge, as it requires a different way to think.

## Description

Write a Scheme program that implements the following functions:

- `(make-point x-cor y-cor)`
  This function needs to create a "list" that will have two elements: `x-cor` and `y-cor`. You will need to use the `cons` function. Additionally, it is suggested that you make a `(get-x point)` and a `(get-y point)` functions, using the `car` and `cdr` functions to retrieve the x and y values of a point.
- `(is-line point1 point2 point3)`
  Using the created points (using make-point) this function returns true (`#t`) if the three points form a line or false (`#f`) if they form a triangle.
- `(distance point1 point2)`
  Calculates the distance between two points
- `(perimeter point1 point2 point3)`
  Calculates the perimeter of a triangle defined by the three points
- `(area point1 point2 point3)`
  Calculates the area of a triangle defined by the three points
- `(calculate-triangle point1 point2 point3)`
  Calculates the perimeter, area and interior angles of the triangle formed by the three points.

### Assignment Requirements

- Good programming practices
    - Correct and readable indentation
- This is an strictly individual assignment

### Delivery Method

- Files to be uploaded
    - threepoints.scm [You MUST name your program this, failure will result in zero grade]
- Uploaded in Canvas

### Assessment and Grading

Assessment will consider the following factors in the grading of the project:

- Adherence to instructions
- Correct function of the program
- No runtime errors
- Late deliveries will have a zero mark
- Plagiarism will have a double zero mark (in addition to losing 10% of your final grade, the group that plagiarizes will lose an additional 10% of their final grade), besides there will be a report filed in the students' academic record.
- Each program will be loaded in Scheme and test it to check if the functions are in working order.

## Extra Challenge (2 bonus points)

**In addition** to the assigned procedures, write a third procedure: `(check-isbn isbn)` This procedure will return true if the ISBN is valid, false otherwise. It should work for both ISBN-10 and ISBN-13. Sample:

```
(check-isbn “0156012197”)
value: #t
(check-isbn “9780156012195”)
value: #t
(check-isbn “9710156012195”)
value: #f
```

## Sample Output

```
local-admins-MacBook-Pro:scheme-assignments arias$ scheme -load threepoints.scm
MIT/GNU Scheme running under OS X
Type `^C' (control-C) followed by `H' to obtain information about interrupts.

Copyright (C) 2014 Massachusetts Institute of Technology
This is free software; see the source for copying conditions. There is NO warranty; not even for
MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.

Image saved on Saturday May 17, 2014 at 2:39:25 AM
  Release 9.2 || Microcode 15.3 || Runtime 15.7 || SF 4.41 || LIAR/x86-64 4.118 || Edwin 3.116
;Loading "threepoints.scm"... done

1 ]=> (make-point 2 3)

;Value 13: (2 . 3)

1 ]=> (is-line (make-point 1 3) (make-point 2 6) (make-point 3 9))

;Value: #t

1 ]=> (distance (make-point 1 3) (make-point 2 6))

;Value: 3.1622776601683795

1 ]=> (perimeter (make-point 2 2) (make-point 5 2) (make-point 3.5 -2))

;Value: 11.54400374531753

1 ]=> (area (make-point 2 2) (make-point 5 2) (make-point 3.5 -2))

;Value: 5.999999999999999

1 ]=> (calculate-triangle (make-point 2 2) (make-point 5 2) (make-point 3.5 -2))
Side 1 = 3
Side 2 = 4.272
Side 3 = 4.272
Perimeter = 11.544
Area = 6.
Angle 1 = .71754   41.11209
Angle 2 = 1.21203   69.44395
Angle 3 = 1.21203   69.44395

;Unspecified return value

1 ]=>
```