*Your submission for this assignment **must include your full name** (as it appears on cuLearn) and you nine-digit **student number as a comment** at the top of **every source file you submit**.*

*Your submission (containing every question) must be a **source file** with a **file name of 'comp3007_f18_#########_a3.hs'** (with the number signs replaced by your nine-digit student number). It **must be written using Haskell** and **must run in GHCi or WinGHCi**.*

*Your submission **must be accompanied by a readme file** containing your name and student number and instructions for the teaching assistant that specify which functions correspond to which assignment questions.*

Compress your submission** into a zip archive named **'comp3007_f18_#########_a3.zip'

***Late assignments will not be accepted** and will **receive a mark of 0**.*

***Submissions that crash** (i.e., terminate with an error) on execution will **receive a mark of 0**.*

The due date for this assignment is Saturday, November 10, 2018, by 11:00pm.

Question 1: "Pseudorandom Numbers"

Develop a ~~recursive~~ linear congruential number generating function that is a given an element of the sequence X_i as an argument and will produce the floating-point value between 0 and 1 (inclusive) that is to be the next element of the sequence. You will need to independently investigate linear congruential generators.

Question 2: "Recursive Structures"

Construct a recursive data structure to represent the tree associated with a simple arithmetic expression constructed entirely from floating-point values and the binary arithmetic operators for addition, subtraction, multiplication, and division.

Your structure must be able to handle division by zero without generating an error (for which you may import Data.Maybe) and it must also support the literal x (meaning your structure should be able to contain expressions like $3x + 4$).

Question 3: "Expression Tree Functions"

Develop an evaluating function that computes the result when one of your arithmetic expression trees (passed as the first argument) is evaluated using a specific value of x (passed as the second argument). As a clarifying example, the evaluation of the structure

```
(Addition (Value x) (Value 3))
```

using a value of 2 should result in a value of 5.

Develop an string representation function that converts one of your arithmetic expression trees (passed as the only argument) into a list of characters that is the way the expression would typically be "written". As a clarifying example, the string representation of the structure

```
(Addition (Value x) (Value 3))
```

should be `(x + 3)`.

Develop a drawing function that creates a graphical [[Char]] representation of a tree argument. Your objective here should be to generate a return value that, when passed as the argument `x` to `putStr (unlines x)`, appears to be "tree-like". As a clarifying example, the "tree-like" depiction of:

```
(Multiplication (Value 4) (Addition (Value x) (Value 3)))
```

could* appear, when printed using `putStr unlines`, as:

```
( * ) ----- 4
      |
      ---( + )---- x
             |
             --- 3
```

**Other visualizations are permitted - it doesn't need to appear exactly the same as the result depicted above, but your result must be a graphically-represented tree, drawn with every connection necessary to depict is as a connected graph. (To clarify, you should not expect full marks on this component if you do not have all the vertical and horizontal lines seen above).*

Question 4: "Expression Tree Mutation"

Develop a mutation function that will randomly mutate an instance of the arithmetic expression tree from the second question using the pseudorandom number generator from the first question. The different possible tree mutations that you must implement are:

a) Replace one literal value with another literal value

e.g., `Addition (Value 1) (Value 2)`

could mutate into

`Addition (Value 1) (Value 3)`

b) Replace a literal value with a randomly generated operation

e.g., `Addition (Value 1) (Value 2)`

could mutate into

`Addition (Value 1) (Subtraction (Value 3) (Value 4))`

c) Replace a subtree with a randomly generated literal value

e.g., `Addition (Value 1) (Subtraction (Value 3) (Value 4))`

could mutate into

`Addition (Value 1) (Value 2)`

You should approach this problem by walking through each node of the tree and using the random number generator to decide whether or not to perform one of the mutations at each node visited.

Everything you submit for this assignment must be a completely original works, authored by you and you alone, prepared for this offering (i.e., Fall 2018) of COMP3007. Do not discuss this (or any other) question with anyone except the instructor or the teaching assistants, and do not copy materials from the internet or any other source.

If you wish to receive a bonus mark for this assignment, you must complete the following additional task:

Question 5: "Trees of Best Fit"

Develop a "fitting" function that is provided with an arithmetic expression tree and another function as arguments, and determine how well the expression tree "fits" the function by providing a Float result. The value 1 should be returned if the tree is a perfect match for the function, and the value 0 should be returned if the tree is the worst possible match for the function.

To accomplish this, you would consider a single set of values, the results that would be produced when these values are each used as inputs to the function argument, and the results that would be produced when these values are used (along with the tree) as inputs to the calculating function from question 2. By considering the difference between the two lists of results you should be able to assess whether the expression represented in the tree is a good fit for the data.

Please note that, for this function, you will need to consider additional arguments that I have not specified, and you will need to investigate techniques for producing a Float-valued measure of fitness. These details are deliberately absent, but you should begin by investigating how correlation coefficients are calculated.