LAB0

1. Based on what you can see when going through the process, describe how your graphing calculator approximates a root (or min/max/PoI/etc). What information must the user provide? In what order? Does it matter? Etc. Use diagrams to support/illustrate your description.

Now, consider this algorithm for approximating the square root of n...
a) Make a guess
b) Divide n by the guess
c) Add the result of (b) to your guess
d) Halve the result of (c)

2. Test this algorithm for n=4, n=9, n=16... making accurate and not-so-accurate guesses. *Note anything notable*.

3. Encode this algorithm as a Scheme procedure named foo, taking 2 inputs (n, g) representing a number and a guess. Use at least the following test cases to test your procedure:

```
(foo 4 2)  "...2"
(foo 4 1)  "...?"
(foo 4 5)  "...?"
(foo 9 3)  "...3"
(foo 9 1)    ?
(foo 9 10)   ?
(foo 16 4) "...4"
(foo 16 1)   ?
(foo 16 15)  ?
(foo 16 900)  ?
```

4. What, in essence, does the above function do?

5. Rename your last function and use it as a helper function to facilitate defining `(mySqrt n)`, which will return an approximation of the square root of n.

6. Explain your development process and finished product.

Nota bene: You may the code below useful. If so, explain the traditional use of epsilon in the sciences, and how the code below figured into your implemented solution.

```
(define epsilon .1)
(define isGoodEnuf? (lambda (a b) (< (abs (- a b)) epsilon)))
```