Richard Hayes Crowley

06/15/2021

CSC\_242\_Lab\_03 Q&A   
  
**1.** A **while()** loop was used in the stater code for repetition control purposes. Could a **for()** loop be used instead of a **while()** loop for our application? Which one do you prefer and why?

**Because the number of iterations needed to find the square root of a number using the Babylonian method is indefinite, a while(True) loop is most appropriate (you *could* use an infinite for loop and then break out of it once a condition is met, but it would be hacky).**

**2.** Can your program be modified to compute the fourth roots of numbers, such as 4 √ [ 200 ] ? Explain your answer.   
  
**Because the fourth power of a number is (n^2)^2 a la the product rule of exponents, we can deduce that the fourth root of a number is sqrt((sqrt(2))). Using our Babylonian function, we can do the same by calling it twice: Babylonian(Babylonian(n))**

**3.** Can your program compute the square root of a non - negative number that is less than 1 ? Explain your answer.  
  
**Yes, my program includes logic that transforms the number into its absolute value using abs(n) and then when printing the output, appends the imaginary decorator *i* to the radical if the number inputted was negative (because the sqrt of a negative number results in an imaginary or complex number).**

**4.** What type of error would be generated when this line of code is executed?

**# print the square root of -3.5**

**print (math.sqrt(-3.5))  
  
A ValueError would result.**

**5.** The square root of a negative number is computed in association with the imaginary unit *i* = √ [ − 1 ] . For example, √ [ − 169 ] is written as 13 *i* . How would you accommodate your program code to account for negative numbers that may be entered into your Babylonian Method application?  
  
**Already done, see answer to question 3.**