You will complete the files Gauss\_Elim.py, NumericalMethods.py, hw2a.py, hw2b.py, and hw2c.py. Those programs will fulfill all of the requirements given in parts a, b, and c below. You must submit your homework by creating a public github repository and then submitting the URL for your repository on CANVAS. The TA will clone your repository for grading and you should not make any changes to the repository after the due date.

In this assignment, you must use variables, loops, if statements, function definitions, function calls and callback functions. For now, you may **NOT** use any of the powerful functions available in python modules, except those explicitly allowed: math, random, copy.

You must include docstrings for all your functions and reasonable comments throughout your code.

See your MAE 3013 textbook for a reminder of:

The Secant Method for finding the solution (root or zero) or a nonlinear equation.

The Simpson’s 1/3 rule for numerical integration.

The Gauss-Sidel method in section 20.3

1. In a file called NumericalMethods.py, write a function defined as:

def Probability(PDF, args, c, GT=True):

PDF: is a callback function for the Gaussian/normal probability density function . The callback function (PDF) should take a single argument as a tuple, which contains values for *x*, μ (population mean), and σ (population standard deviation).

args: is a tuple containing μ and σ

c: is a floating point value (i.e., the upper limit of integration)

GT: is a boolean indicating if we want the probability of x being greater than c (GT=True) or less than c (GT=False)

To find the probability of x<c you should use the Simpson’s 1/3 rule to integrate PDF between the limits of x=μ-5⋅σ to c.

In a file titled hw2a.py, write and call a main() function that uses your Probability function to find:

P(x<105|N(100,12.5)): probability x<105 given a normal distribution of x with μ=100, σ=12.5

P(x>μ+2σ|N(100, 3))

Print your findings to the *cli* in the following format:

P(x<1.00|N(0,1))=Y.YY

P(x>181.00|N(175,3))=Z.ZZ

b) In the file NumericalMethods.py from part a, write a function defined as:

def Secant(fcn, x0, x1, maxiter=10, xtol=1e-5):

Purpose: use the Secant Method to find the root of fcn(x), in the neighborhood of x0 and x1.

fcn: the function for which we want to find the root

x0 and x1: two x values in the neighborhood of the root

xtol: exit if the |xnewest - xprevious| < xtol

maxiter: exit if the number of iterations (***new x values***) equals this number

return value: the final estimate of the root (most recent new x value)

In a file titled hw2b.py, write and call a main() function that uses your Secant function to estimate and print the solution of:

; with x0=1, x1= 2, maxiter = 5 and xtol = 1e-4

; with x0=1, x1= 2, maxiter = 15 and xtol = 1e-8

 with x0=1, x1= 2, maxiter = 3 and xtol = 1e-8

c) In your NumericalMethods.py file, write a function defined as:

def GaussSeidel(Aaug, x, Niter = 15):

Purpose: use the Gauss-Seidel method to estimate the solution to a set of N linear equations expressed in matrix form as **A**x = **b**. Both **A** and **b** are contained in the function argument – Aaug.

Aaug: an augmented matrix containing [A | b ] having N rows and N+1 columns, where N is the number of equations in the set.

x: a vector (array) contains the values of the initial guess

Niter: the number of iterations (new x vectors) to compute

return value: the final new x vector.

In a file titled hw2c.py, write and call a main() function that uses your GaussSeidel function to estimate and print the solutions to the following sets of linear equations:





**Note: as a first step in the GaussSeidel method, make sure the matrix is diagonal dominant!**

**See** Gauss\_Elim.py