You will write **three Python programs** named **hw2a.py**, **hw2b.py**, and **hw2c.py**. Those programs will fulfill all the requirements given in parts a, b, and c below. You will place those four python files into a single compressed file named hw2.zip. You will upload hw2.zip to the Homework 2 dropbox.

How we will grade:

You will test your functions and using the numerical values and mathematical functions given. When we grade your assignment, we will run your program with those given numerical values, looking for correct answers. Then we will change the numerical values (including changing the SIZES of the arrays) and look for correct answers for those modified values as well. We will only use numerical values, array sizes and functions that make sense. We will not be testing your program to see how it handles bad data. As always, commenting your code effectively is important and will be part of the grading. I expect the use of **docstrings** and clear description of the arguments/parameters to functions.

In this assignment, you must use variables, loops, if statements, your own function definitions, and function calls to write the required functions. For now, you may not use any of the powerful functions available in python modules, with these exceptions: you may import and use functions from the **math**, **copy**, and **random** modules.

See your MAE 3013 textbook for a reminder of:

The Simpson’s 1/3 rule for numerical integration (§19.5, p831)

The Secant Method for finding the solution (root or zero) of a nonlinear equation (§19.2, p805)

The use of cofactors and minor matrices for finding determinant of a matrix (§7.7)

The Cramer’s method for solving a matrix equation(§7.7)

a) Write a function defined as: def Probability(PDF, args, c, GT=True):

PDF: is the callback function for the Gaussian/normal probability density function , which takes 1 argument in the form of a tuple containing 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

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.

Write and call a **main()** function that uses your **Probability** function to find:

P(x<1|N(0,1)): probability x<1 given a normal distribution of x with μ=0, σ=1

P(x>μ+2σ|N(175, 3))

Print you findings to the console in the following format:

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

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

b) 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 the callback **fcn**(x), in the neighborhood of x0 and x1.

fcn: the callback 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)

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) Write a function defined as: def Cramer(Aaug):

Purpose: use Cramer’s method to find 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.

return x: the solution vector.

Write and call a main() function that uses your Cramer function to solve and print the solutions to the following sets of linear equations:



