Assignment 4 – Due Sunday March 31, 2021 at 11:55 pm

Each program must start with a multi-line comment as shown below. Replace each occurrence of DentStew and Stew Dent with your name. Replace yyyy/mm/dd with the date you completed the program.

```
# DentStewAxQy.py
#
# Course:      COMP 1012
# Instructor:  Ramin
# Assignement: x Question y
# Author:      Stew Dent
# Version:     yyyy/mm/dd
#
# Purpose:     The purpose of the program.
```

Begin each program with the following statement.

```
from time import ctime
```

The name of each program should be of the form:

LastnameFirstnameAnQm.py

If your name is Stew Dent and the program is for assignment 3 question 1 then the name of the program would be:

DentStewA4Q1.py

No email submission will be accepted, zero will be granted otherwise as the mark for the assignment. Attach the .py file for each program. There should be only one submission in which all of the programs are written and separated with appropriate comments. **Do NOT** use zip, rar or any other packaging of your files!

Question 1

You must use dictionary in this question. Do not use arrays in this question!

The purpose of this question is to write a Python program that can work with lists and dictionaries. Assume that you want to use dictionary to find different solutions of the equation below:

$$2x-3y+z=5$$

Pass a list of the coefficients to a function. For example for the above equation the list of equations would be :

Coeff= [2,-3,1]

Also, we know that x,y, and z are integer and generally in the range of -9 to 9. The range of each x,y and z is entered by user. Write a function that generates all the possible described situation with the header:

```
def buildPossibleAnswers(Coeff,Xmin, Xmax, Ymin, Ymax, Zmin, Zmax):
```

Call this function to build all possible answer for this equation. The function displays the *prompt* to tell the user what to enter. If the user does not enter valid input display the appropriate error message as described below and display the *prompt* again asking for more input from the user. Repeat this until the user enters valid input. When the user enters valid input return possible combinations of the answers in a list.

Error situations:

- If the user presses enter/return without entering anything display the message 'Missing input!'
- If the inputs are not *integers* display the value of the input and the message 'is not valid!'
- If the inputs are not in the desired range, print 'Out of range!'

Write a function that begins with the following header:

```
def computeError(a, k):
```

Given the value of **a**, the list that is the output of the "buildPossibleAnswers" function and "k" is the desired right-hand side of the equation; for the example above, k=5. The output of "computeErrors" should be a dictionary in which the possible answers are the key element and the difference between the output of the equation because of those values and the expected output is the value for that key element.

Note: Keep in mind that a key in a dictionary is in "*string";* therefore, you have to convert possible answer to string before adding it to your dictionary.

For example if we call two functions *buildPossibleAnswers* and *computeError* respectively, the answer should like below for the mentioned equation:

```
a=buildPossibleAnswers([2,-3,1],-9, 9, -9, 9, -9, 9)
computeError(a, 5)
```

Your possible answers and errors for the equation 2x-3y+z=5 are:

```
{'[1, -1, -1]': 1,
 '[1, -1, -2]': 2,
 '[1, -1, -3]': 3,
 '[1, -1, -4]': 4,
 '[1, -1, -5]': 5,
 '[1, -1, 0]': 0,
 '[1, -1, 1]': -1,
 '[1, -1, 2]': -2,
 '[1, -1, 3]': -3,
 '[1, -1, 4]': -4,
 '[1, -1, 5]': -5,
 '[1, -1, 6]': -6,
 '[1, -2, -1]': -2,
 '[1, -2, -2]': -1,
 '[1, -2, -3]': 0,
 '[1, -2, -4]': 1,
 '[1, -2, -5]': 2,
 '[1, -2, 0]': -3,
 '[1, -2, 1]': -4,
 '[1, -2, 2]': -5,
 '[1, -2, 3]': -6,
 '[1, -2, 4]': -7,
 '[1, -2, 5]': -8,
 '[1, -2, 6]': -9,
 '[1, -3, -1]': -5,
 '[1, -3, -2]': -4,
 '[1, -3, -3]': -3,
... }
```

('...' means that the list might be longer and may continue)

In the next step, you have to choose the options that makes the error between the value of the inserted numbers and the actual value zero. In the other word, you have to find the answers of this equation.
Write a function that begins with the following header:

```
def findSolutions (R):
```

where ***R*** is the output of the previous function *"computeErrors"*

In the final step, write a main script and call all these functions in it and print the possible answers like below:

Note: Make sure that your program returns back the answers in *list*.

```
----------------------------------------------------------------------------------------------
There are total 12 possible answers for this equations in the desired range:
  X   Y   Z
1:[1, -2, -3]
2:[1, -1, 0]
3:[1, 0, 3]
4:[1, 1, 6]
5:[2, -2, -5]
6:[2, -1, -2]
7:[2, 0, 1]
8:[2, 1, 4]
9:[3, -1, -4]
10:[3, 0, -1]
11:[3, 1, 2]
12:[3, 2, 5]

Programmed by Stew Dent.
Date: Sun Mar  7 20:20:22 2021
End of processing.
```

The main program does the following:

- Using string replication display a line of dashes.
- Calls all the function that you wrote in the previous steps
- Display a message that indicates number of possible solution in the desired range of user
- Showing the time and date at the end of the process

Note: There is NO function named *main* in this program!

Question 2

In this question you will learn how to plot two modulate sinus on each other. To do this, build a function that can generate a sinus wave with the specified frequency, over the specified time interval. The header of this function would be like below:

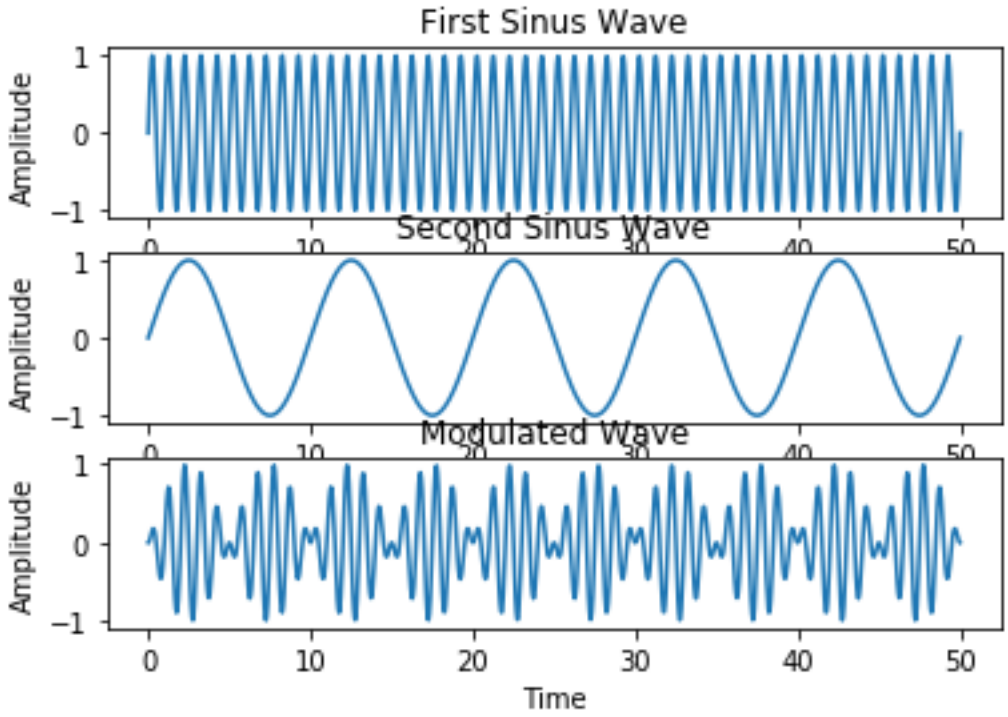
def sinTest(A,F,time,numel)

where A is amplitude, F is frequency of the sinus wave, time is the time length that you want to show in sinus wave in it and numel is the number of sinus values that you calculated in the time interval for that sinus wave.

Inside another function, *SinModule*, you have to call the *sinTest* twice for two different frequencies and amplitudes. Therefore, you need a function with a header like below:

def SinModule(A1,F1,A2,F2,time,numel)

where A1, F1,A2,F2 are the amplitudes and frequencies of the first and second sinus waves respectively. Afterwards, you have to multiply the sinus values of each wave element by element to build the modulate wave. Use *subplot* and plot the first, second and the modulate signals all in a figure and save the figure in *png* format. The output of your code for *SinModule(1,1,1,10,50,1000)*, should be like below:



Question 3

Note: Use sets and dictionary to solve this question!

In this question, you will learn how to read a file and recognize prime number in it. For this purpose, you have to first write a function that read the provided text file (A4_Q3.txt). Then, converts all the strings to the integer numbers and return the numbers in an array. The header of this function would be:

```
def   txt2Int(filename)
```

filename is the name of the text file that you want to read its contents. To recognize if any of the numbers in the file is prime, use the output of *txt2Int* function as the input for *PrimeRec* function.

```
def PrimeRec(Num)
```

where *Num* is the output of the *txt2Int* function. The *PrimeRec* function should be able to recognize the numbers that are not prime and return back one of their divisors in a dictionary. For this question, in the output:

1) The set of primes should be sorted.
2) No need to sort the dictionary of non-prime numbers
3) For the dictionary of non-primes, each non-prime number is a key and the divisor that is displayed as the value of each key, is the largest divisor not equal to the main number (key) itself.

Therefore, the output of the code would be like below:

```
--------------------------------------------------------------------------------
Prime set is: {2, 3, 5, 7, 11, and 13}

Dictionary of none prime numbers and a divisor: {12: [6], 14: [7], 10: [5], 50: [25], 123: [41], 18: [9], 124: [62],
9: [3], 145: [29], 632: [316], 32: [16], 253: [23], 819: [273]}

Programmed by Stew Dent.
Date: Sun Mar  7 21:20:22 2021
End of processing.
```