Lab 17 - Classes and operator overloading

**Q1**. Try to answer this question before running the code. You might run it to check your answer.

**class** TestClass(object):

**def** \_\_init\_\_(self,param\_str=**''**):

self.the\_str=**''**

**for** c **in** param\_str:

**if** c.isalpha():

self.the\_str += c

**def** \_\_add\_\_(self,param):

**if** type(param)==TestClass:

the\_str = self.the\_str + param.the\_str

**return** TestClass(the\_str)

**else**:

**return** self

**def** \_\_str\_\_(self):

**return 'Value: {}'**.format(self.the\_str)

inst1 = TestClass(**'abc'**)

inst2 = TestClass(**'123ijk'**)

sumInst1 = inst1 + inst2

sumInst2 = inst1 + **'xyz'**

print(inst1) *# Line 1*

print(sumInst1) *# Line 2*

print(sumInst2) *# Line 3*

print(isinstance(sumInst2,TestClass)) *# Line 4*

(a) What output is produced by # Line 1 of the above program?

(b) What output is produced by # Line 2 of the above program?

(c) What output is produced by # Line 3 of the above program?

(d) What output is produced by # Line 4 of the above program?

**Q2.**

Write a class called WholeNumber class. The whole numbers are the non-negative integers: 0,1,2, . . . Your class must handle addition, subtraction, and multiplication of whole numbers—no division or mixed-type (whole number and integer) operations need be handled. Your class must also handle printing—e.g., if x is an instance of the WholeNumber class, you must be able to print x.

Two cases must not be allowed:

(1) you must not be able to create a WholeNumber that has a negative value;

(2) an arithmetic operation cannot be allowed to have a negative result.

In both cases, an error message must be printed. Remember that arithmetic must return a whole number. That is, if x and y are whole numbers, the result of x + y must be a whole number. Include sample code that uses your class and demonstrates the use of all methods as well as error handling.

**Q3.** Write a class for linear equations. A generic linear equation is of the form y = mx + b where m and b are constants. Include the following methods:

(a) \_\_init\_\_, \_\_str\_\_, \_\_repr\_\_.

(b) value(x), which returns the value of the equation given x.

(c) compose(LinearEquation) that composes two linear equations. That is, if y = ax + b and z = cx + d, then y(z)= (a\*c)x +(a\*d + b) and will be called as y.compose(z). Note that the compose operation is not commutative.

(d) add returns the sum of two linear equations. That is, if y = ax + b and z = cx + d, then y + z = (a + c)x + (b + d).