Chapter-1:

R-1.2: Write a short Python function, is even(k), that takes an integer value and returns True if k is even, and False otherwise. However, your function cannot use the multiplication, modulo, or division operators.

**def even(k):**

**check = True**

**counter1 = 0**

**counter2 = 0**

**while k != counter1 || k != counter2:**

**check = Not check**

**counter1 += 1**

**counter2 -= 1**

**return check**

R-1.4: Write a short Python function that takes a positive integer n and returns the sum of the squares of all the positive integers smaller than n.

**def sum\_squares(n):**

**if n <= 0:**

**return 0**

**sum = 0**

**for I in range(1,n)**

**sum += pow(I,2)**

**return sum**

C-1.15: Write a Python function that takes a sequence of numbers and determines if all the numbers are different from each other (that is, they are distinct).

**def is\_duplicates(s):**

**for I in s:**

**counter = 0:**

**for j in s:**

**if j == i:**

**counter += 1**

**if counter > 1:**

**return True**

**return False**

C-1.17: Had we implemented the scale function (page 25) as follows, does it work properly?

def scale(data, factor):

  for val in data:

  val \*= factor

 Explain why or why not.

**No because using a for loop in that regard isn’t accessing the reference of the value in the list but is shallow copying it to the temporary variable “val”. When running this code instead of modifying the value in the list it simply repeats the iteration of the list.**

C-1.24: Write a short Python function that counts the number of vowels in a given character string.

**def num\_vowels(s):**

**s = s.lower()**

**n\_vowels = 0**

**vowels = (“a”,”I”,”o”,”u”,”e”)**

**for I in vowels:**

**n\_vowels += s.count(i)**

**return n\_vowels**

Chapter-2:

R-2.5: Use the techniques of Section 1.7 to revise the charge and make payment methods of the CreditCard class to ensure that the caller sends a number as a parameter.

**def charge(self, price):**

**if (type(price) != int || type(price) != float): raise TypeError(“Input element Price must be a number.”)**

**if price + self.\_balance > self.\_limit**

**return False**

**else:**

**self.\_balance += price**

**return True**

**def make\_payment(self, amount):**

**if (type(price) != int || type(price) != float): raise TypeError(“Input element Price must be a number.”)**

**self.\_balance -= amount**

R-2.8: Modify the declaration of the first for loop in the CreditCard tests, from Code Fragment 2.3, so that it will eventually cause exactly one of the three credit cards to go over its credit limit. Which credit card is it?

**……**

**……**

**for val in range(1,1000):**

**check1 = wallet[0].charge(val)**

**if not check1: break**

**check2 = wallet[1].charge(2\*val)**

**if not check2: break**

**check3 = wallet[2].charge(3\*val)**

**if not check3: break**

**……**

**……**

R-2.10: Implement the neg method for the Vector class of Section 2.3.3, so that the expression −v returns a new vector instance whose coordinates are all the negated values of the respective coordinates of v.

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

**n\_coords = [0] \* len(self.\_coords)**

**for I in range(len(self.\_coords):**

**n\_coords[i] = self.\_coords[i] \* -1**

**return n\_coords**