Inspiring Excellence

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Homework No:	**08**
Topic:	**Inheritance**
Submission Type:	**Submission Link:** https://docs.google.com/forms/d/e/1FAIpQLSfAVmet1WHiNsGoe_6nYz9ZfG0E19A2ZYZ1-chdy5GVvaKlcA/viewform
Resources:	**1. Class lectures** **2. BuX lectures** **a. English:** **i. Inheritance: here**

	b. Supplementary: **i. Inheritance: here**

Task 1

Write the **Mango** and the **Jackfruit** classes so that the following code generates the output below:

```
class Fruit:
    def __init__(self, formalin=False, name=''):
        self.__formalin = formalin
        self.name = name

    def getName(self):
        return self.name

    def hasFormalin(self):
        return self.__formalin

class testFruit:
    def test(self, f):
        print('----Printing Detail----')
        if f.hasFormalin():
            print('Do not eat the',f.getName(),'.')
            print(f)
        else:
            print('Eat the',f.getName(),'.')
            print(f)

m = Mango()
j = Jackfruit()
t1 = testFruit()
t1.test(m)
t1.test(j)
```

OUTPUT:

```
----Printing Detail-----
Do not eat the Mango.
Mangos are bad for you
----Printing Detail-----
Eat the Jackfruit.
Jackfruits are good for you
```

Task 2

Write the **ScienceExam** class so that the following code generates the output below:

```
class Exam:
    def __init__(self,marks):
        self.marks = marks
        self.time = 60

    def examSyllabus(self):
        return "Maths , English"
    def examParts(self):
        return "Part 1 - Maths\nPart 2 - English\n"


engineering = ScienceExam(100,90,"Physics","HigherMaths")
print(engineering)
print('---------------------------------')
print(engineering.examSyllabus())
print(engineering.examParts())
print('=================================')
architecture =
ScienceExam(100,120,"Physics","HigherMaths","Drawing")
print(architecture)
print('---------------------------------')
print(architecture.examSyllabus())
print(architecture.examParts())
```

OUTPUT:

```
Marks: 100 Time: 90 minutes Number
of Parts: 4
---------------------------------
Maths , English , Physics ,
HigherMaths
Part 1 - Maths
Part 2 - English
Part 3 - Physics
Part 4 - HigherMaths
=================================
Marks: 100 Time: 120 minutes Number
of Parts: 5
---------------------------------
Maths , English , Physics ,
HigherMaths , Drawing
Part 1 - Maths
Part 2 - English
Part 3 - Physics
Part 4 - HigherMaths
Part 5 - Drawing
```

Task 3

Write the **PokemonExtra** class so that the following code generates the output below:

```
class PokemonBasic:

  def __init__(self, name = 'Default', hp = 0,
weakness = 'None', type = 'Unknown'):
    self.name = name
    self.hit_point = hp
    self.weakness = weakness
    self.type = type

  def get_type(self):
    return 'Main type: ' + self.type

  def get_move(self):
    return 'Basic move: ' + 'Quick Attack'

  def __str__(self):
    return "Name: " + self.name + ", HP: " +
str(self.hit_point) + ", Weakness: " + self.weakness

print('\n------------Basic Info:--------------')
pk = PokemonBasic()
print(pk)
print(pk.get_type())
print(pk.get_move())

print('\n------------Pokemon 1 Info:-------------')
charmander = PokemonExtra('Charmander', 39, 'Water',
'Fire')
print(charmander)
print(charmander.get_type())
print(charmander.get_move())

print('\n------------Pokemon 2 Info:-------------')
charizard = PokemonExtra('Charizard', 78, 'Water',
'Fire', 'Flying', ('Fire Spin', 'Fire Blaze'))
print(charizard)
print(charizard.get_type())
print(charizard.get_move())
```

OUTPUT:

```
------------Basic Info:--------------
Name: Default, HP: 0, Weakness: None
Main type: Unknown
Basic move: Quick Attack

------------Pokemon 1 Info:--------------
Name: Charmander, HP: 39, Weakness: Water
Main type: Fire
Basic move: Quick Attack

------------Pokemon 2 Info:--------------
Name: Charizard, HP: 78, Weakness: Water
Main type: Fire, Secondary type: Flying
Basic move: Quick Attack
Other move: Fire Spin, Fire Blaze
```

Task 4

A renowned Bakery shop recently launched cheesecakes into their cakes menu. Cheesecakes will have all the general attributes of the regular cakes but it has some special features. Design the **Cakes** (parent) and **Cheese_Cakes** (child) classes so that the following output is produced. Note that:

- 1kg regular cake price is 1200 Taka and 1 kg cheese-cake price is 1500 Taka
- As cheese-cakes are newly launched, they need user feedback. For this reason, if a customer gives feedback on cheese-cakes he'll get 10% discounts on his next purchase.

Write the classes **Cakes** and **Cheese_Cakes** to generate the following output.

Driver Code:	Output:
order_1=Cakes("Chocolate",500) order_2=Cakes("Vanilla",800) print("(1)**********") order_1.cake_details() print("(1.1)**********") print(Cakes.order_list) print("(2)**********") order_2.add_customization("Zero","Butter Cream") order_2.cake_details() print("(3)**********") Cakes.give_feedbacks("Chocolate Cake","Very Delicious") Cakes.give_feedbacks("Chocolate Cake","Yummy") print("(4)**********") Cakes.show_feedbacks() print("(5)**********") ch_order1=Cheese_Cakes("Red velvet",700) ch_order1.cake_details() print("(6)**********") ch_order1.add_customization() print("(7)**********") ch_order2=Cheese_Cakes("Blue Berry",900,"No Bake") ch_order2.cake_details() print("(8)**********") print(Cakes.order_list) print("(9)**********") Cheese_Cakes.give_feedbacks("Red velvet Cheese Cake","Average") Cheese_Cakes.give_feedbacks("Blue Berry Cheese Cake","Liked it")	(1)********** Flavor: Chocolate Cake, Weight: 500 gm Sweetness: Moderate sugar, Cream Type: Whipped Cream Price: 600.0 Taka (1.1)********** {'Chocolate Cake 500gm': 1, 'Vanilla Cake 800gm': 1} (2)********** Flavor: Vanilla Cake, Weight: 800 gm Sweetness: Zero sugar, Cream Type: Butter Cream Price: 960.0 Taka (3)********** Thanks for your valuable feedback! Thanks for your valuable feedback! (4)********** {'Chocolate Cake': ['Very Delicious', 'Yummy']} (5)********** Flavor: Red velvet Cheese Cake, Weight: 700 gm Sweetness: Moderate sugar, Cream Type: Cream Cheese Cake Type:baked, Price: 1050.0 Taka (6)********** Sorry! No customization available for cheese cakes. (7)********** Flavor: Blue Berry Cheese Cake, Weight: 900 gm Sweetness: Moderate sugar, Cream Type: Cream Cheese Cake Type:No Bake, Price: 1350.0 Taka (8)********** {'Chocolate Cake 500gm': 1, 'Vanilla Cake 800gm':

print("(10)**********") Cakes.show_feedbacks()	1, 'Red velvet Cheese Cake 700gm': 1, 'Blue Berry Cheese Cake 900gm': 1} (9)********** Thanks for your valuable feedback! You will get 10% discount for your next purchase! Thanks for your valuable feedback! You will get 10% discount for your next purchase! (10)********** {'Chocolate Cake': ['Very Delicious', 'Yummy'], 'Red velvet Cheese Cake': ['Average'], 'Blue Berry Cheese Cake': ['Liked it']}

Task 5

```
1   class A:
2     temp = 3
3     def __init__(self):
4       self.sum = 0
5       self.y = 0
6       self.y = A.temp - 1
7       self.sum = A.temp + 2
8       A.temp -= 2
9
10    def methodA(self, m, n):
11      x = 0
12      n[0] += 1
13      self.y = self.y + m + A.temp
14      A.temp += 1
15      x = x + 2 + n[0]
16      n[0] = self.sum + 2
```

```
17      print(f"{x} {self.y} {self.sum}")
18
19  class B(A):
20    x = 1
21    def __init__(self, b = None):
22      super().__init__()
23      self.sum = 2
24      if b == None:
25        self.y = self.temp + 1
26        B.x = 3 + A.temp + self.x
27        A.temp -= 2
28      else:
29        self.sum = self.sum + self.sum
30        B.x = b.x + self.x
31    def methodB(self, m, n):
32      y = [0]
33      self.y = y[0] + self.y + m
34      B.x = self.y + 2 +  self.temp - n
35      self.methodA(self.x, y)
36      self.sum = self.x + y[0] + self.sum
37      print(f"{self.x} {y[0]} {self.sum}")
```

Write the output of the following code:

```
x = [23]
a1 = A()
b1 = B()
b2 = B(b1)
a1.methodA(1, x)
b2.methodB(3, 2)
a1.methodA(1, x)
```

Output:

x	y	sum