BRAC UNIVERSITY

Inspiring Excellence

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Homework No:	**04**
Topic:	**Instance method and overloading**
Submission Type:	**Hard Copy**
Resources:	**1. Class lectures** **2. BuX lectures** **a. English: https://www.youtube.com/watch?v=eUQZYzszZf8** **b. Supplementary: https://shorturl.at/uwENV**

Task 1

Analyze the given code below to write the **HackathonTeam** class to get the output shown.

Hints:

- Remember, the constructor is a special method. Here, you have to deal with constructor overloading, which is similar to method overloading.
- A team can have a maximum of three members.
- Your class should have two variables.
- You may need to use the keyword ‘None’.

Driver Code	Output

# Write your codes for subtasks here. 	
team_1 = HackathonTeam("Atlantean", "Aquaman")
team_1.information()
 	
print("=================")
 	
team_2 = HackathonTeam("Avengers", "Ironman", "Thor", "Hulk")
team_2.information()
 	
print("=================")
 	
team_3 = HackathonTeam("X-Men", "Storm", "Mystique")
team_3.information()
	Team name: Atlantean
	Participants:
	Aquaman
	=================
	Team name: Avengers
	Participants:
	Ironman
	Thor
	Hulk
	=================
	Team name: X-Men
	Participants:
	Storm
	Mystique

Task 2

Design the **Foodie** class with the necessary properties so that the given output is produced for the provided driver code.

Notes:

1. **Your code should work for any number of strings passed to order() method.**
2. **Total spent by a foodie is calculated by adding the total prices of all the ordered foods and the waiter's tips (if any).**
3. **Global variable 'menu' can be accessed directly from inside the class.**

Driver Code

```
menu = {'Chicken Lollipop':15,'Beef
Nugget':20,'Americano':180,'Red
Velvet':150,'Prawn Tempura':80,'Saute
Veg':200}

f1 = Foodie('Frodo')
print(f1.show_orders())
print('1----------------------')
f1.order('Chicken Lollipop-3','Beef
Nugget-6','Americano-1')
print('2----------------------')
print(f1.show_orders())
print('3----------------------')
f1.order('Red Velvet-1')
print('4----------------------')
f1.pay_tips(20)
print('5----------------------')
print(f1.show_orders())
f2 = Foodie('Bilbo')
print('6----------------------')
f2.order('Prawn Tempura-6','Saute
Veg-1')
print('7----------------------')
f2.pay_tips()
print('8----------------------')
print(f2.show_orders())
```

Output

```
Frodo has 0 item(s) in the cart.
Items: []
Total spent: 0.
1----------------------
Ordered - Chicken Lollipop, quantity - 3, price (per
Unit) - 15.
Total price - 45
Ordered - Beef Nugget, quantity - 6, price (per Unit)-
20.
Total price - 120
Ordered - Americano, quantity - 1, price (per Unit)-
180.
Total price - 180
2----------------------
Frodo has 3 item(s) in the cart.
Items: ['Chicken Lollipop', 'Beef Nugget',
'Americano']
Total spent: 345.
3----------------------
Ordered - Red Velvet, quantity - 1, price (per Unit)-
150.
Total price - 150
4----------------------
Gives 20/- tips to the waiter.
5----------------------
Frodo has 4 item(s) in the cart.
Items: ['Chicken Lollipop', 'Beef Nugget',
'Americano', 'Red Velvet']
Total spent: 515.
6----------------------
Ordered - Prawn Tempura, quantity - 6, price (per
Unit)- 80.
```

	Total price - 480 Ordered - Saute Veg, quantity - 1, price (per Unit)- 200. Total price - 200 7---------------------- No tips to the waiter. 8---------------------- Bilbo has 2 item(s) in the cart. Items: ['Prawn Tempura', 'Saute Veg'] Total spent: 680.

Task 3

Write a class called **Farmer** with the required constructor and methods to get the following output.

Driver Code

```python
f1 = Farmer()
print("--------------------")
f1.addCrops('Rice', "Jute", "Cinnamon")
print("--------------------")
f1.addFishes()
print("--------------------")
f1.addCrops('Mustard')
print("--------------------")
f1.showGoods()
print("--------------------")
f2 = Farmer("Korim Mia")
print("--------------------")
f2.addFishes('Pangash', 'Magur')
print("--------------------")
f2.addCrops("Wheat", "Potato")
print("--------------------")
f2.addFishes("Koi", "Tuna", "Sardine")
print("--------------------")
f2.showGoods()
print("--------------------")
f3 = Farmer(2865127000)
print("--------------------")
f3.addCrops()
print("--------------------")
f3.addFishes("Katla")
print("--------------------")
f3.showGoods()
print("--------------------")
```

Output

```
Welcome to your farm!
--------------------
3 crop(s) added.
--------------------
No fish added.
--------------------
1 crop(s) added.
--------------------
You have 4 crop(s):
Rice,Jute,Cinnamon,Mustard
You don't have any fish(s).
--------------------
Welcome to your farm, Korim Mia!
 --------------------
2 fish(s) added.
--------------------
2 crop(s) added.
--------------------
3 fish(s) added.
--------------------
You have 2 crop(s):
Wheat,Potato
You have 5 fish(s):
Pangash,Magur,Koi,Tuna,Sardine
--------------------
Welcome to your farm. Your farm ID
is 2865127000!
--------------------
No crop(s) added.
--------------------
1 fish(s) added.
--------------------
You don't have any crop(s).
You have 1 fish(s):
Katla
--------------------
```

Task 4

Lucky winners have gotten a free one-day tour to Universal Studios. Being a programmer, you are asked to construct a class named “**UniversalStudiosUser**” that can easily store any visitor’s information and preferred rides. Your output should match the given output.

Instructions:

- Create a class called **UniversalStudiosUser**
- Create the required constructor
- Create a method called **selected_rides** that can take as many arguments as the user wants to give.
- Lucky winners (Special users) get a 20% discount if they select more than 3 rides. Normal users do not get any discounts.

Driver Code	Output
See code below	See output below

Driver Code

```python
customer_1 = UniversalStudiosUser("Alice", 21,
"Special")
print("--------- 1 ---------")
customer_1.selected_rides("Waterworld-100",
"Accelerator-200", "DinoSoarin-50")
print("--------- 2 ---------")
customer_1.show_details()

print("=================")

customer_2 = UniversalStudiosUser("Bob", 20,
"Normal")
print("--------- 3 ---------")
customer_2.selected_rides("Enchanted Airways-300",
"Jurassic Park-500", "Accelerator-200",
"DinoSoarin-50")
print("--------- 4 ---------")
customer_2.show_details()
```

Output

```
Welcome to Universal Studios.
--------- 1 ---------
Added ride(s) successfully.
--------- 2 ---------
Your information:
Name: Alice, Age: 21, Category: Special
Selected rides:
Ride: Waterworld, Amount: 100 dollar(s)
Ride: Accelerator, Amount: 200 dollar(s)
Ride: DinoSoarin, Amount: 50 dollar(s)
Please pay 350.0 dollar(s).
 =================
Welcome to Universal Studios.
--------- 3 ---------
Added ride(s) successfully.
--------- 4 ---------
Your information:
Name: Bob, Age: 20, Category: Normal
Selected rides:
Ride: Enchanted Airways, Amount: 300 dollar(s)
Ride: Jurassic Park, Amount: 500 dollar(s)
Ride: Accelerator, Amount: 200 dollar(s)
Ride: DinoSoarin, Amount: 50 dollar(s)
Please pay 1050.0 dollar(s).
=================
Welcome to Universal Studios.
```

```python
print("=================")

customer_3 = UniversalStudiosUser("Mark", 15, "Special")
print("--------- 5 ---------")
customer_3.selected_rides("Transformers-450", "Jurassic Park-500", "Waterworld-100", "DinoSoarin-50")
print("--------- 6 ---------")
customer_3.show_details()
```

```
--------- 5 ---------
Added ride(s) successfully.
--------- 6 ---------
Your information:
Name: Mark, Age: 15, Category: Special
Selected rides:
Ride: Transformers, Amount: 450 dollar(s)
Ride: Jurassic Park, Amount: 500 dollar(s)
Ride: Waterworld, Amount: 100 dollar(s)
Ride: DinoSoarin, Amount: 50 dollar(s)
Congratulations!!! You've got a 20% discount.
Please pay 880.0 dollar(s).
```

Task 5

Design the **Department** class with the necessary properties so that the given output is produced for the provided driver code.

Hints:

1. Your code should work for any number of integers passed to the add_students() method. The method will calculate the average number of students if the number of integers passed is equal to the number of classes.
2. Your code should work for any number of Department objects passed to the merge_Department() method.
3. The average students of the mega department in the merge_Department() method are calculated in this way -

Total students of mega department= mega department average * mega department sections + department 1 average * department 1 sections + department 2 average * department 2 sections + department 3 average * department 3 sections + … … …

Average students of mega department = (Total students of mega department / mega department sections)

Driver Code	Output
d1 = Department()	The ChE Department has 5 sections.
print('1----------------------------------')	1----------------------------------
d2 = Department('MME Department')	The MME Department has 5 sections.
print('2----------------------------------')	2----------------------------------
d3 = Department('NCE Department', 8)	The NCE Department has 8 sections.
print('3----------------------------------')	3----------------------------------
d1.add_students(12, 23, 12, 34, 21)	The ChE Department has an average of 20.4 students in
print('4----------------------------------')	each section.
d2.add_students(40, 30, 21)	4----------------------------------
print('5----------------------------------')	The MME Department doesn't have 3 sections.
d3.add_students(12, 34, 41, 17, 30, 22, 32, 51)	5----------------------------------
print('6----------------------------------')	The NCE Department has an average of 29.88 students in
mega = Department('Engineering Department',	each section.
10)	6----------------------------------
print('7----------------------------------')	The Engineering Department has 10 sections.
mega.add_students(21,30,40,36,10,32,27,51,45,15)	7----------------------------------
print('8----------------------------------')	The Engineering Department has an average of 30.7
print(mega.merge_Department(d1, d2))	students in each section.
print('9----------------------------------')	8----------------------------------
print(mega.merge_Department(d3))	ChE Department is merged to Engineering Department.
	MME Department is merged to Engineering Department.
	Now the Engineering Department has an average of 40.9
	students in each section.
	9----------------------------------
	NCE Department is merged to Engineering Department.
	Now the Engineering Department has an average of 64.8
	students in each section.

Task 6

Implement the **StudentRoutineGenerator** class with the necessary properties so that the given output is produced for the following driver code.
[You are not allowed to change the driver code.]

Driver Code

```
print('##################################')
st1 = StudentRoutineGenerator('Harry', 4)
print('------------------------------')
st1.addCourses('CSE110-Mon/Wed-12:30',
'MAT110-Mon/Wed-2:00')
st1.addCourses('ENG101-Sun/Tue-12:30',
'CSE110-Mon/Wed-9:30')
st1.addCourses('PHY111-Sun/Tue-12:30')
print('------------------------------')
st1.showRoutine()
print('------------------------------')
st1.dropCourse('CSE110')
st1.dropCourse('PHY112')
print('------------------------------')
st1.showRoutine()
print('##################################')
st2 = StudentRoutineGenerator('John', 3)
print('------------------------------')
st2.addCourses('MAT110-Mon/Wed-8:00')
st2.addCourses('ENG101-Sat/Thurs-12:30',
'CSE110-Sun/Tue-9:30')
st2.addCourses('PHY111-Sun/Tue-12:30')
print('------------------------------')
st2.showRoutine()
```

Output

```
##################################
Name: Harry
Maximum Number of Courses: 4
Initial Routine: {'Sat/Thurs': {},
'Sun/Tue': {}, 'Mon/Wed': {}}
------------------------------
Successfully added CSE110!
Successfully added MAT110!
Successfully added ENG101!
You already have CSE110 in your
routine
Can't take PHY111. It's clashing with
your ENG101
------------------------------
Updated Routine:
{'Sat/Thurs': {}, 'Sun/Tue': {'12:30':
'ENG101'}, 'Mon/Wed': {'12:30':
'CSE110', '2:00': 'MAT110'}}
Routine Details:
Sun/Tue:
12:30 - ENG101
Mon/Wed:
12:30 - CSE110
2:00 - MAT110
------------------------------
Successfully dropped CSE110
No such course in your routine
------------------------------
Updated Routine:
{'Sat/Thurs': {}, 'Sun/Tue': {'12:30':
'ENG101'}, 'Mon/Wed': {'2:00':
'MAT110'}}
Routine Details:
Sun/Tue:
12:30 - ENG101
Mon/Wed:
2:00 - MAT110
##################################
Name: John
Maximum Number of Courses: 3
Initial Routine: {'Sat/Thurs': {},
```

	'Sun/Tue': {}, 'Mon/Wed': {}} ------------------------------ Successfully added MAT110! Successfully added ENG101! Successfully added CSE110! You can't take more than 3 courses ------------------------------ Updated Routine: {'Sat/Thurs': {'12:30': 'ENG101'}, 'Sun/Tue': {'9:30': 'CSE110'}, 'Mon/Wed': {'8:00': 'MAT110'}} Routine Details: Sat/Thurs: 12:30 - ENG101 Sun/Tue: 9:30 - CSE110 Mon/Wed: 8:00 - MAT110

Task 7

```python
class Test4:
    def __init__(self):
        self.sum, self.y = 0, 0
    def methodA(self):
        x, y = 0, 0
        msg = [0]
        msg[0] = 5
        y = y + self.methodB(msg[0])
        x = y + self.methodB(msg, msg[0])
        self.sum = x + y + msg[0]
        print(x, y, self.sum)
    def methodB(self, *args):
        if len(args) == 1:
            mg1 = args[0]
            x, y = 0, 0
            y = y + mg1
            x = x + 33 + mg1
            self.sum = self.sum + x + y
            self.y = mg1 + x + 2
            print(x, y, self.sum)
            return y
        else:
            mg2, mg1 = args
            x = 0
            self.y = self.y + mg2[0]
            x = x + 33 + mg1
            self.sum = self.sum + x + self.y
            mg2[0] = self.y + mg1
            mg1 = mg1 + x + 2
            print(x, self.y, self.sum)
            return self.sum
```

t3 = Test4() t3.methodA() t3.methodA() t3.methodA() t3.methodA()	x	y	sum

Task 8

```
class msgClass:
    def __init__(self):
        self.content = 0
class Q5:
    def __init__(self):
        self.sum = 1
        self.x = 2
        self.y = 3
    def methodA(self):
        x, y = 1, 1
        msg = []
        myMsg = msgClass()
        myMsg.content = self.x
        msg.append(myMsg)
        msg[0].content = self.y + myMsg.content
        self.y = self.y + self.methodB(msg[0])
        y = self.methodB(msg[0]) + self.y
        x = y + self.methodB(msg[0], msg)
        self.sum = x + y + msg[0].content
        print(x," ", y," ", self.sum)
```

```python
    def methodB(self, mg1, mg2 = None):
        if mg2 == None:
            x, y = 5, 6
            y = self.sum + mg1.content
            self.y = y + mg1.content
            x = self.x + 7 +mg1.content
            self.sum = self.sum + x + y
            self.x = mg1.content + x +8
            print(x, " ", y," ", self.sum)
            return y
        else:
            x = 1
            self.y += mg2[0].content
            mg2[0].content = self.y + mg1.content
            x = x + 4 + mg1.content
            self.sum += x + self.y
            mg1.content = self.sum - mg2[0].content
            print(self.x, " ",self.y," ", self.sum)
            return self.sum
```

What is the output of the following code sequence?
 	
`q = Q5()`
`q.methodA()`
	x	y	sum