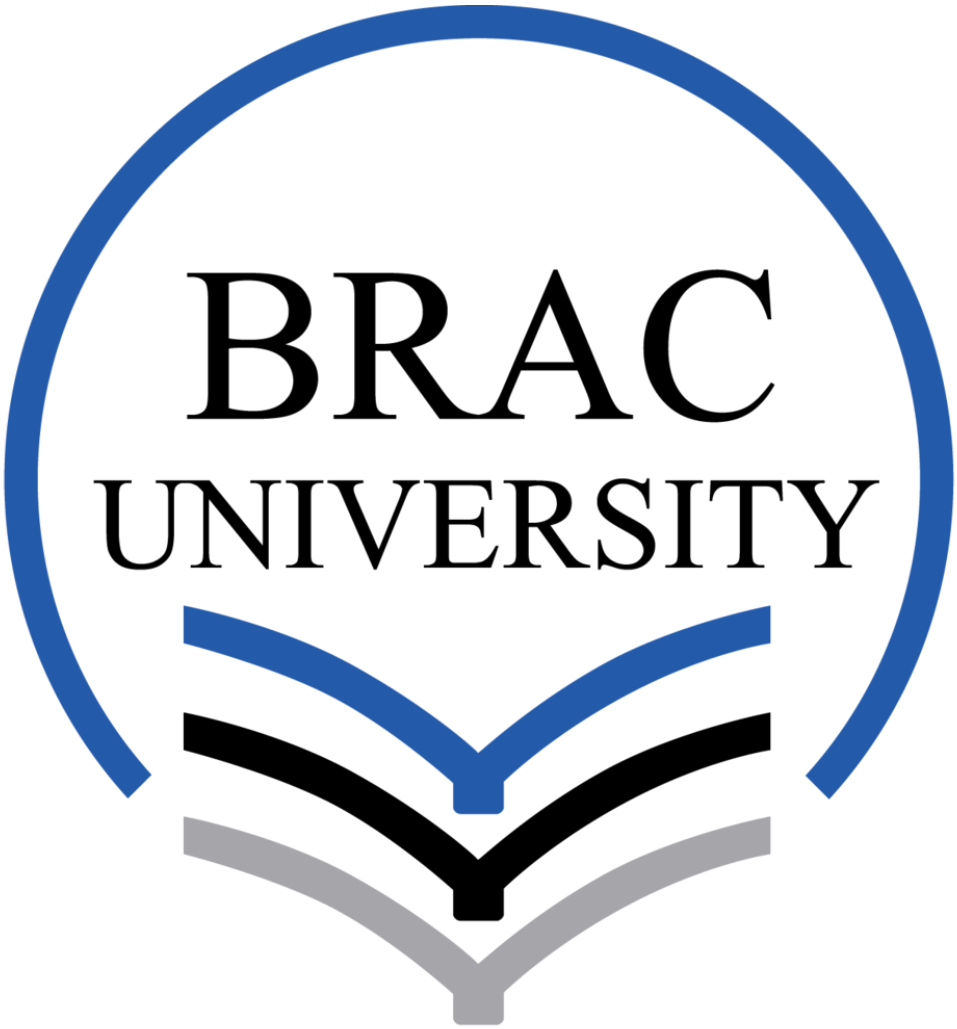

Inspiring Excellence

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Classwork No:	**04**
Topic:	**OOP (Instance method and overloading)**
Number of Tasks:	**5**

Task 1

Design the **Student** class in such a way so that the following code provides the expected output.
Hint:

- Write the constructor with appropriate default value for arguments.
- Write the dailyEffort() method with appropriate arguments.
- Write the printDetails() method. You can follow the printing suggestions below:
  - If hour <= 2 print 'Suggestion: Should give more effort!'
  - Else if hour <= 4 print 'Suggestion: Keep up the good work!'
  - Else print 'Suggestion: Excellent! Now motivate others.'

[You are not allowed to change the code below]

Driver Code	Output

`# Write your code here.`
 	
`harry = Student('Harry Potter', 123)`
`harry.dailyEffort(3)`
`harry.printDetails()`
`print('========================')`
`john = Student("John Wick", 456, "BBA")`
`john.dailyEffort(2)`
`john.printDetails()`
`print('========================')`
`naruto = Student("Naruto Uzumaki", 777, "Ninja")`
`naruto.dailyEffort(6)`
`naruto.printDetails()`
	`Name: Harry Potter`
	`ID: 123`
	`Department: CSE`
	`Daily Effort: 3 hour(s)`
	`Suggestion: Keep up the good work!`
	`========================`
	`Name: John Wick`
	`ID: 456`
	`Department: BBA`
	`Daily Effort: 2 hour(s)`
	`Suggestion: Should give more effort!`
	`========================`
	`Name: Naruto Uzumaki`
	`ID: 777`
	`Department: Ninja`
	`Daily Effort: 6 hour(s)`
	`Suggestion: Excellent! Now motivate others.`

Task 2

Write the **Farmer** class with the required constructor, methods to get the following output.

Driver Code	Output

`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("--------------------")`
	`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 3

Using the **TaxiLagbe** app, users can share a single taxi with multiple people.

Implement the design of the **TaxiLagbe** class with the necessary properties so that the given output is produced for the provided driver code:

[Hint: 1. Each taxi can carry a maximum of 4 passengers

2. The addPassenger() method takes the last name of the passenger and ticket fare for that person in an underscore (_)-separated string.**]**

Driver Code	Output

`# Write your code here`
 	
`taxi1 = TaxiLagbe('1010-01', 'Dhaka')`
`print('---------------------------------')`
`taxi1.addPassenger('Walker_100',`
`'Wood_200','Matt_100')`
`taxi1.addPassenger('Wilson_105')`
`print('---------------------------------')`
`taxi1.printDetails()`
`print('---------------------------------')`
`taxi1.addPassenger('Karen_200')`
`print('---------------------------------')`
`taxi1.printDetails()`
`print('---------------------------------')`
`taxi2 = TaxiLagbe('1010-02', 'Khulna')`
`taxi2.addPassenger('Ronald_115', 'Parker_215')`
`print('---------------------------------')`
`taxi2.printDetails()`
	`-----------------------------------`
	`Dear Walker! Welcome to TaxiLagbe.`
	`Dear Wood! Welcome to TaxiLagbe.`
	`Dear Matt! Welcome to TaxiLagbe.`
	`Dear Wilson! Welcome to TaxiLagbe.`
	`-----------------------------------`
	`Trip info for Taxi number: 1010-01`
	`This taxi can only cover the Dhaka area.`
	`Total passengers: 4`
	`Passenger lists:`
	`Walker, Wood, Matt, Wilson`
	`Total collected fare: 505 Taka`
	`-----------------------------------`
	`Taxi Full! No more passengers can be added.`
	`-----------------------------------`
	`Trip info for Taxi number: 1010-01`
	`This taxi can only cover the Dhaka area.`
	`Total passengers: 4`
	`Passenger lists:`
	`Walker, Wood, Matt, Wilson`
	`Total collected fare: 505 Taka`
	`-----------------------------------`
	`Dear Ronald! Welcome to TaxiLagbe.`
	`Dear Parker! Welcome to TaxiLagbe.`
	`-----------------------------------`
	`Trip info for Taxi number: 1010-02`
	`This taxi can only cover the Khulna area.`
	`Total passengers: 2`
	`Passenger lists:`
	`Ronald, Parker`
	`Total collected fare: 330 Taka`

Task 4

Design the **Sphere** class such that the following output is produced. **Hints:**

- Volume of the sphere = $\frac{4}{3} * \pi * r^3$, where r = radius of the sphere and π = 3.1416.
- Merging spheres together conserves the total volume. The volume of the bigger sphere can be calculated by adding the volume of the spheres being merged. [see pictures for details]. Pay attention to how the object is updated.
- When spheres of different colors are merged together then the merged sphere will have '**Mixed Color**' instead of one particular color.
- Your code should work for any number of Sphere objects passed to the **merge_sphere()** method.
- The default value of the radius r is 1.

r1 = 6 **r2 = 4** **v3 = v1+v2=1172.864**
v1 = 904.7808 **v2 = 268.0832**

#Write your code here	**Output:**
sphere1 = Sphere("Sphere 1") print("1***************") sphere1.printDetails() print("2***************") sphere2 = Sphere("Sphere 2", 3) print("3***************") sphere2.printDetails() print("4***************") sphere3 = Sphere("Sphere 3", 2) print("5***************") sphere3.printDetails() print("6***************") sphere3.merge_sphere(sphere1,sphere2) print("7***************") sphere3.printDetails() print("8***************") sphere4 = Sphere("Sphere 4", 5, "Purple") print("9***************") sphere4.merge_sphere(sphere3)	1*************** Sphere ID: Sphere 1 Color: White Volume: 4.1888 2*************** 3*************** Sphere ID: Sphere 2 Color: White Volume: 113.09759999999999 4*************** 5*************** Sphere ID: Sphere 3 Color: White Volume: 33.5104 6*************** Spheres are being merged 7*************** Sphere ID: Sphere 3 Color: White Volume: 150.7968

print("10***************") sphere4.printDetails()	8*************** 9*************** Spheres are being merged 10*************** Sphere ID: Sphere 4 Color: Mixed Color Volume: 674.3967999999999

Task 5

```
1  class ABC:
2      def __init__(self):
3          self.x = 3
4          self.y = 7
5          self.sum = 0
6      def methodA(self, x):
7          self.y = x + self.sum + self.x
8          self.sum = x + self.y
9          z = ABC()
10         z.sum = self.sum + self.y
11         self.methodB(z)
12         print(self.x, self.y, self.sum)
13     def methodB(self, a):
14         y = 3
15         a.x = self.x + self.sum
16         self.sum = a.x + a.y + y
17         print(a.x, a.y, a.sum)
18
```

Write the output of the following code: a = ABC()			

a.methodA(5)			