Inspiring Excellence

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Homework No:	**03**
Topic:	**OOP(Classes and objects)**
Submission Type:	**Hard Copy (Only submit the part of the code that you have been instructed to write. DO NOT write any given code.)**
Resources:	**1. Class lectures** **2. BuX lectures** **a. English: https://shorturl.at/dhjAZ** **b. Supplementary: https://shorturl.at/wMPRU**

Task 1

Design the **CellPackage** class and write suitable driver code to produce the output:
Subtasks:

- (#1) **Assign** the arguments into appropriate attributes: **data**, **talk_time**, **messages**, **cashback**, **validity** and **price** via a parameterized constructor. All the attributes should be of int data type. Note that **data** is stored in *Megabytes* (*1 GB = 1024 MB*) and the **cashback** amount is calculated from a percentage value.
- (#2,3,4) **Implement** driver code to display all the information of a package. **Check** if any particular attribute does not exist (is equal to 0), do not print that attribute. Attributes **validity** and **price** are always printed.

```
# Driver Code
# Subtask 1: Write the CellPackage Class

pkg = CellPackage(150, '6 GB', 99, 20, '7%', 7)
print('============= Package 1 =============')
# Subtask 2: Check each attribute and print

pkg2 = CellPackage(700, '35 GB', 700, 0, '10%', 30)
print('============= Package 2 =============')
# Subtask 3: Check each attribute and print

pkg4 = CellPackage(120, '0 GB', 190, 0, '0%', 10)
print('============= Package 3 =============')
# Subtask 4: Check each attribute and print
```

Output:

```
============= Package 1 =============
Data = 6144 MB
Talktime = 99 Minutes
SMS/MMS = 20
Validity = 7 Days
--> Price = 150 tk
Buy now to get 10 tk cashback.
============= Package 2 =============
Data = 35840 MB
Talktime = 700 Minutes
Validity = 30 Days
--> Price = 700 tk
Buy now to get 70 tk cashback.
============= Package 3 =============
Talktime = 190 Minutes
Validity = 10 Days
--> Price = 120 tk
```

Task 2

Part A:

Write the **box** class so that the given driver code gives the expected output.

[You are not allowed to change the code below]

Driver Code	Output

`# Write your class code here`
`print("Box 1")`
`b1 = box([10,10,10])`
`print("==========")`
`b1.boxDescription()`
`print(b1.volume())`
`print("----------")`
`print("Box 2")`
`b2 = box((30,10,10))`
`print("==========")`
`b2.boxDescription()`
`print(b2.volume())`
`b2.height = 300`
`print("Updating Box 2!")`
`print("Height:", b2.height)`
`print("Width:", b2.width)`
`print("Breadth:", b2.breadth)`
`volume = b2.height * b2.width * b2.breadth`
`print(f"Volume of the box is {volume} cubic units.")`
`print("----------")`
`print("Box 3")`
`b3 = b2`
`b3.boxDescription()`
`print(b3.volume())`
	`Box 1`
	`Creating a Box!`
	`==========`
	`Height: 10`
	`Width: 10`
	`Breadth: 10`
	`Volume of the box is 1000 cubic units.`
	`----------`
	`Box 2`
	`Creating a Box!`
	`==========`
	`Height: 30`
	`Width: 10`
	`Breadth: 10`
	`Volume of the box is 3000 cubic units.`
	`Updating Box 2!`
	`Height: 300`
	`Width: 10`
	`Breadth: 10`
	`Volume of the box is 30000 cubic units.`
	`----------`
	`Box 3`
	`Height: 300`
	`Width: 10`
	`Breadth: 10`
	`Volume of the box is 30000 cubic units.`

Part B

After the given driver code, if we run the following lines of code:

```
one = (b3 == b2)

b3.width = 100
two = (b3 == b2)
```

1. What will be the values for variables `one` and `two`? Explain your answer briefly in text.

2. What will be the value of `b2.width`? Has that value changed since the driver code ran? If yes, explain why in brief text.

Task 3

Read the following **Vector3D** class that represents a vector in 3D space. The x-axis, y-axis, and z-axis components of the vector are represented by the attributes *x, y,* and *z* respectively.

[You are not allowed to change the code below]

```
class Vector3D:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z
        print(f'Vector <{self.x}, {self.y}, {self.z}>
created.')

# Write your driver code here
```

Your task is to write the driver code that:

- Creates 2 **Vector3D** objects. The first one is given as $V_1 =< 2, \ -3, \ 1 >$ and the second one is given as $V_2 =< -1, \ 4, \ 0 >$.
- Prints their components and magnitude as shown in the output. The magnitude of a vector $< a, \ b, \ c >$ is calculated as $| < a, \ b, \ c > | = \sqrt{a^2 + b^2 + c^2}$.
- Finds the **dot product** of the 2 Vectors. Dot product of 2 vectors $< a_1, \ a_2, \ a_3 >$ and $< b_1, \ b_2, \ b_3 >$ is calculated as $< a_1, \ a_2, \ a_3 > \ \cdot \ < b_1, \ b_2, \ b_3 > \ = a_1 b_1 + a_2 b_2 + a_3 b_3$.
- Finds the **Cross product** of the 2 Vectors. The cross product of 2 vectors $< a_1, \ a_2, \ a_3 >$ and $< b_1, \ b_2, \ b_3 >$ creates ***a new Vector3D Object*** which is calculated as $< a_1, \ a_2, \ a_3 > \ \times \ < b_1, \ b_2, \ b_3 > \ = \ < a_2 b_3 - a_3 b_2, \ a_3 b_1 - a_1 b_3, \ a_1 b_2 - a_2 b_1 >$.
- Generates the output as given below.
- Your program should run for any two 3D vectors.

Output:

```
Vector <2, -3, 1> has been created.
Vector <-1, 4, 0> has been created.
Magnitude of the first vector = 3.7416573867739413
Magnitude of the second vector = 4.123105625617661
Dot product of the two vectors = -14
Vector <-4, -1, 5> has been created.
Cross product of the two vectors = <-4, -1, 5>
```

Task 4

Design the following **abcTech** class so that it generates the following output:

Hints:

- If the working hour of the employee is more than 144 hours in the month, then he/she will be paid Tk. 800 for each extra hour worked along with the base salary.
- If the working hour of the employee is less than or equal to 144 hours, then the salary will be the same as the base salary.

Driver Code	Output

`print("-------------------------")`
`b1 =abcTech("Tamim Hasan", "Software Engineer", "Android Development")`
`print("-------------------------")`
`b1.addProgrammingSkills(["Java", "Python"])`
`b1.addProgrammingSkills(["Dart", "C++"])`
`b1.addFrameworks(["Express.js", "React"])`
`b1.printInfo()`
`print("-------------------------")`
`print(f"Your salary for this month is Tk. {b1.calculateSalary(45000, 156)}")`
`print("-------------------------")`
`print("-------------------------")`
`b2 =abcTech("Jahin Khandoker", "Senior Developer", "App Development")`
`print("-------------------------")`
`b2.addProgrammingSkills(["Java", "Dart", "Swift"])`
	`-------------------------`
	`Welcome to abcTech, Tamim Hasan!`
	`-------------------------`
	`Name:  Tamim Hasan`
	`Designation:  Software Engineer`
	`Department: Android Development`
	`Programming Skills:  Java, Python, Dart, C++`
	`Frameworks:  Express.js, React`
	`-------------------------`
	`Your salary for this month is Tk. 54600`
	`-------------------------`
	`-------------------------`
	`Welcome to abcTech, Jahin Khandoker!`
	`-------------------------`
	`Name:  Jahin Khandoker`
	`Designation:  Senior Developer`
	`Department:  App Development`
	`Programming Skills:  'Java', 'Dart', 'Swift'`
	`Frameworks:  'Flutter', 'React Native', 'Xamarin'`
	`-------------------------`

b2.addFrameworks(["Flutter", "React Native"]) b2.addFrameworks(["Xamarin"]) b2.printInfo() print("-------------------------") print(f"Your salary for this month is Tk. {b2.calculateSalary(103000, 123)}") print("-------------------------")	Your salary for this month is Tk. 103000 -------------------------

Task 5

Design **StudentDatabase** class so that the following output is produced: Calculation of GPA:
GPA = Sum of (Grade Points * Credits)/ Credits attempted

- Each course a student takes is of 3 credits.
- **For example:** Wanda has taken 3 courses in Summer 2022 semester. So her CGPA will be

  [(CSE111 GP × 3) + (CSE260 GP × 3) + (ENG101 GP × 3)] / (3 courses × 3)

  [(3.7 × 3) + (3.7 × 3) + (4.0 × 3)] / (3 × 3) = **3.8**

Driver Code	Output

# Write your code here 	
s1 = StudentDatabase('Pietro', '10101222')
s1.calculateGPA(['CSE230: 4.0', 'CSE220: 4.0', 'MAT110: 4.0'], 'Summer2020')
s1.calculateGPA(['CSE250: 3.7', 'CSE330: 4.0'], 'Summer2021')
print(f'Grades for {s1.name}\n{s1.grades}')
print('---------------------------------')
s1.printDetails()
s2 = StudentDatabase('Wanda', '10103332')
s2.calculateGPA(['CSE111: 3.7', 'CSE260: 3.7', 'ENG101: 4.0'], 'Summer2022')
print('---------------------------------')
print(f'Grades for {s2.name}\n{s2.grades}')
print('---------------------------------')
s2.printDetails()
	Grades for Pietro
	{'Summer2020': {('CSE230', 'CSE220', 'MAT110'): 4.0}, 'Summer2021': {('CSE250', 'CSE330'): 3.85}}
	--------------------------------- 	
	Name: Pietro
	ID: 10101222
	Courses taken in Summer2020:
	CSE230
	CSE220
	MAT110
	GPA: 4.0
	Courses taken in Summer2021:
	CSE250
	CSE330
	GPA: 3.85
	--------------------------------- 	
	Grades for Wanda

	{'Summer2022': {('CSE111', 'CSE260', 'ENG101'): 3.8}} -------------------------------- Name: Wanda ID: 10103332 Courses taken in Summer2022: CSE111 CSE260 ENG101 GPA: 3.8

Task 6

Imagine your friend owns a grocery store and he is having trouble managing the day to day activities. So he wants a software that will track and manage the stock of his items, the current balance of the store etc. Now you have offered to help because you are a kind soul and also this will be a paid project. Design the **Store** class to generate the desired output:

Driver Code	Expected Output
print("==========") branch1 = Store(5000) print(f"Current Balance: {branch1.balance}") print(f"Total items: {branch1.total_items}") branch1.viewAllItems() branch1.viewAllItemDetails() print("==========") print(f"Current Balance: {branch1.balance}") branch1.add_item(["ChaCha Noodles", 10, 5, 8]) print(f"Current Balance: {branch1.balance}") branch1.add_item(["Sparrow Shampoo", 5, 10, 20]) print(f"Current Balance: {branch1.balance}") print("==========") branch1.viewAllItems() print() branch1.viewAllItemDetails() print() print("==========") print(f"Current Balance: {branch1.balance}\n")	========== New branch created! Current Balance: 5000 Total items: 0 There are no items in your inventory {} ========== Current Balance: 5000 Item added: ChaCha Noodles Current Balance: 4950 Item added: Sparrow Shampoo Current Balance: 4900 ========== All Items: ChaCha Noodles, Sparrow Shampoo {'ChaCha Noodles': {'stock': 10, 'buying_price': 5, 'selling_price': 8}, 'Sparrow Shampoo': {'stock': 5, 'buying_price': 10,

```
branch1.sell_item("ChaCha Noodles", 15)
print(f"Current Balance: {branch1.balance}\n")
branch1.viewAllItemDetails()
print()
branch1.sell_item("ChaCha Noodles", 10)
print()
print(f"Current Balance: {branch1.balance}\n")
branch1.viewAllItemDetails()
print()
print("==========================")
print(f"Current Balance: {branch1.balance}\n")
branch1.restock_item("ChaCha Noodles", 5)
print()
branch1.viewAllItemDetails()
print()
print(f"Current Balance: {branch1.balance}\n")
print("==========================")
```

```
'selling_price': 20}}

==========================
Current Balance: 4900

Sorry! ChaCha Noodles is not
available at your desired
quantity. Currently we have: 10
Current Balance: 4900

{'ChaCha Noodles': {'stock': 10,
'buying_price': 5,
'selling_price': 8}, 'Sparrow
Shampoo': {'stock': 5,
'buying_price': 10,
'selling_price': 20}}


Current Balance: 4980

{'ChaCha Noodles': {'stock': 0,
'buying_price': 5,
'selling_price': 8}, 'Sparrow
Shampoo': {'stock': 5,
'buying_price': 10,
'selling_price': 20}}

==========================
Current Balance: 4980

Restocked item: ChaCha Noodles,
Current Stock: 5

{'ChaCha Noodles': {'stock': 5,
'buying_price': 5,
'selling_price': 8}, 'Sparrow
Shampoo': {'stock': 5,
'buying_price': 10,
'selling_price': 20}}

Current Balance: 4955

==========================
```

Task 7

```
1  class Scope:
2      def __init__(self):
3          self.x, self.y, self.z = 1, 8, [1,2,3]
4      def met1(self):
5          x = 3
6          x = self.x + self.z[2] + self.met2(self.z[0])
7          self.y = self.y + self.x + 5
8          self.z[2] = self.y + self.met2(3) + self.y
9          print(x, self.y, self.z[2])
10     def met2(self, x=2):
11         y = x + self.z[0]
12         print(self.x, y, self.z[1])
13         self.x = self.x + y + self.z[1]
14         self.y = self.y + 13
15         self.z[1] = y + self.y
16         return self.x + y
```

Write the output of the following code:

```
q2 = Scope()
q2.met1()
q2.met2(8)
q1=q2
q2.met2()
```

x	y	z

Task 8

```
1  class Test7:
2    def __init__(self):
3        self.sum = 0
4        self.y = 0
5    def methodA(self):
6        x=y=k=0
7        msg = [5]
8        while (k < 2):
9            y += msg[0]
10           x = y + self.methodB(msg, k)
11           self.sum = x + y + msg[0]
12           print(x ," " , y, " " , self.sum)
13           k+=1
14   def methodB(self, mg2, mg1):
15         x = 0
16         self.y += mg2[0]
17         x = x + 3 + mg1
18         self.sum += x + self.y
19         mg2[0] = self.y + mg1
20         mg1 += x + 2
21         print(x , " " ,self.y, " " , self.sum)
22         return mg1
```

What is the output of the following code sequence? t1 = Test7() t1.methodA()	x	y	sum

t1.methodA()			