



Inspiring Excellence

Course Title: Programming Language II

Course Code: CSE 111

Lab Assignment no: 3 & 4 Merged

Task 1

Implement the design of the **Patient** class so that the following output is produced:

[For BMI, the formula is $BMI = \text{weight}/\text{height}^2$, where weight is in kg and height in meters]

Driver Code	Output
<pre># Write your code here p1 = Patient("A", 55, 63.0, 158.0) p1.printDetails() print("=====") p2 = Patient("B", 53, 61.0, 149.0) p2.printDetails()</pre>	<pre>Name: A Age: 55 Weight: 63.0 kg Height: 158.0 cm BMI: 25.236340330075304 ===== Name: B Age: 53 Weight: 61.0 kg Height: 149.0 cm BMI: 27.476239809017613</pre>

Task 2

Design a class Shape for the given code below.

- Write a class Shape.
- Write the required constructor that takes 3 parameters and initialize the instance variables accordingly.
- Write a method area() that prints the area.

Hint: the area method can calculate only for the shapes: Triangle, Rectangle, Rhombus, and Square. So, you have to use conditions inside this method

For this task, assume that --

- for a triangle, the arguments passed are the base and height
- for a rhombus, the arguments passed are the diagonals
- for a square or rectangle, the arguments passed are the sides.

Driver Code	Output
<pre># Write your code here triangle = Shape("Triangle",10,25) triangle.area() print("=====") square = Shape("Square",10,10) square.area() print("=====") rhombus = Shape("Rhombus",18,25) rhombus.area() print("=====") rectangle = Shape("Rectangle",15,30) rectangle.area() print("=====") trapezium = Shape("Trapezium",15,30) trapezium.area()</pre>	<pre>Area: 125.0 ===== Area: 100 ===== Area: 225.0 ===== Area: 450 ===== Area: Shape unknown</pre>

Task 3

Implement the design of the **Calculator** class so that the following output is produced:

Driver Code	Output
<pre># Write your code here c1 = Calculator() print("=====") val = c1.calculate(10, 20, '+') print("Returned value:", val) c1.showCalculation() print("=====") val = c1.calculate(val, 10, '-') print("Returned value:", val) c1.showCalculation() print("=====") val = c1.calculate(val, 5, '*') print("Returned value:", val) c1.showCalculation() print("=====") val = c1.calculate(val, 16, '/') print("Returned value:", val) c1.showCalculation()</pre>	<pre>Calculator is ready! ===== Returned value: 30 10 + 20 = 30 ===== Returned value: 20 30 - 10 = 20 ===== Returned value: 100 20 * 5 = 100 ===== Returned value: 6.25 100 / 16 = 6.25</pre>

Task 4

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

Hint:

- o Write the constructor with appropriate printing and multiple arguments.
- o Write the addExp() method with appropriate printing and argument.
- o Write the printDetails() method

[You are not allowed to change the code below]

<pre># Write your code here. p1 = Programmer("Ethen Hunt", "Java", 10) p1.printDetails() print('-----') p2 = Programmer("James Bond", "C++", 7) p2.printDetails() print('-----') p3 = Programmer("Jon Snow", "Python", 4) p3.printDetails() p3.addExp(5) p3.printDetails()</pre>	<p>OUTPUT:</p> <p>Horray! A new programmer is born Name: Ethen Hunt Language: Java Experience: 10 years. -----</p> <p>Horray! A new programmer is born Name: James Bond Language: C++ Experience: 7 years. -----</p> <p>Horray! A new programmer is born Name: Jon Snow Language: Python Experience: 4 years. Updating experience of Jon Snow Name: Jon Snow Language: Python Experience: 9 years.</p>
--	---

Task 5

Implement the design of the **UberEats** class so that the following output is produced:

[For simplicity, you can assume that a customer will always order exact 2 items]

Driver Code	Output
<pre># Write your code here order1 = UberEats("Shakib", "01719658xxx", "Mohakhali") print("=====") order1.add_items("Burger", "Coca Cola", 220, 50) print("=====") print(order1.print_order_detail()) print("=====") order2 = UberEats ("Siam", "01719659xxx", "Uttara") print("=====") order2.add_items("Pineapple", "Dairy Milk", 80, 70) print("=====") print(order2.print_order_detail())</pre>	<pre>Shakib, welcome to UberEats! ===== ===== User details: Name: Shakib, Phone: 01719658xxx, Address: Mohakhali Orders: {'Burger': 220, 'Coca Cola': 50} Total Paid Amount: 270 ===== Siam, welcome to UberEats! ===== ===== User details: Name: Siam, Phone: 01719659xxx, Address: Uttara Orders: {'Pineapple': 80, 'Dairy Milk': 70} Total Paid Amount: 150</pre>

Task 6

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

Subtasks:

1. Create a class called Customer.
2. Create the required constructor.
3. Create a method called **greet** that works if no arguments are passed or if one argument is passed. (*Hint: You may need to use the keyword NONE*)
4. Create a method called **purchase** that can take as many arguments as the user wants to give.

[You are not allowed to change the code below]

Write your codes for subtasks 1-4 here.

```
customer_1 = Customer("Sam")
customer_1.greet()
customer_1.purchase("chips", "chocolate", "orange juice")
print("-----")
customer_2 = Customer("David")
customer_2.greet("David")
customer_2.purchase("orange juice")
```

OUTPUT:

```
Hello!
Sam, you purchased 3 item(s):
chips
chocolate
orange juice
-----
Hello David!
David, you purchased 1 item(s):
orange juice
```

Task 7

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

Hints:

- *Remember, the constructor is a special method. Here, you have to deal with constructor overloading which is similar to method overloading.*
- *You may need to use the keyword None*
- *Your class should have 2 variables*

[You are not allowed to change the code below]

<i>#Write your code here</i>	<i>OUTPUT</i>
<pre>c1 = Cat() c2 = Cat("Black") c3 = Cat("Brown", "jumping") c4 = Cat("Red", "purring") c1.printCat() c2.printCat() c3.printCat() c4.printCat() c1.changeColor("Blue") c3.changeColor("Purple") c1.printCat() c3.printCat()</pre>	<pre>White cat is sitting Black cat is sitting Brown cat is jumping Red cat is purring Blue cat is sitting Purple cat is jumping</pre>

Task 8

Design the **Student** class 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. For printing suggestions check the following instructions.
 - ☐ If hour <= 2 print 'Suggestion: Should give more effort!'
 - ☐ 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]

<p># Write your code here.</p> <pre>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()</pre>	<p>OUTPUT:</p> <pre>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.</pre>
--	--

Task 9

Implement the design of the **Batsman** class so that the following output is produced:

Hint: Batting strike rate (s/r) = runsScored / ballsFaced x 100.

Driver Code	Output
<pre># Write your code here b1 = Batsman(6101, 7380) b1.printCareerStatistics() print("=====") b2 = Batsman("Liton Das", 678, 773) b2.printCareerStatistics() print("-----") print(b2.battingStrikeRate()) print("=====") b1.setName("Shakib Al Hasan") b1.printCareerStatistics() print("-----") print(b1.battingStrikeRate())</pre>	<pre>Name: New Batsman Runs Scored: 6101 , Balls Faced: 7380 ===== Name: Liton Das Runs Scored: 678 , Balls Faced: 773 ----- 87.71021992238033 ===== Name: Shakib Al Hasan Runs Scored: 6101 , Balls Faced: 7380 ----- 82.66937669376694</pre>

Task 10

Implement the design of the **Author** class so that the following output is produced:

Driver Code	Output
<pre># Write your code here auth1 = Author('Humayun Ahmed') auth1.addBooks('Deyal', 'Megher Opor Bari') auth1.printDetails() print("=====") auth2 = Author() print(auth2.name) auth2.changeName('Mario Puzo') auth2.addBooks('The Godfather', 'Omerta', 'The Sicilian') print("=====") auth2.printDetails() print("=====") auth3 = Author('Paolo Coelho', 'The Alchemist', 'The Fifth Mountain') auth3.printDetails()</pre>	<pre>Author Name: Humayun Ahmed ----- List of Books: Deyal Megher Opor Bari ===== Default ===== Author Name: Mario Puzo ----- List of Books: The Godfather Omerta The Sicilian ===== Author Name: Paolo Coelho ----- List of Books: The Alchemist The Fifth Mountain</pre>

Task 11

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

Implement the design of the **TaxiLagbe** class so that the following output is produced:

Hint:

1. Each taxi can carry maximum 4 passengers
2. addPassenger() method takes the last name of the passenger and ticket fare for that person in an underscore (-) separated string.

Driver Code	Output
<pre># Write your code here # Do not change the following lines of code. taxi1 = TaxiLagbe('1010-01', 'Dhaka') print('-----') taxi1.addPassenger('Walker_100', 'Wood_200') taxi1.addPassenger('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') taxi2.addPassenger('Parker_215') print('-----') taxi2.printDetails()</pre>	<pre>----- 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 cover only 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 cover only 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 cover only Khulna area. Total passengers: 2 Passenger lists: Ronald, Parker Total collected fare: 330 Taka</pre>

Task 12

Implement the design of the **Account** class so that the following output is produced:

Driver Code	Output
<pre># Write your code here a1 = Account() print(a1.details()) print("-----") a1.name = "Oliver" a1.balance = 10000.0 print(a1.details()) print("-----") a2 = Account("Liam") print(a2.details()) print("-----") a3 = Account("Noah",400) print(a3.details()) print("-----") a1.withdraw(6930) print("-----") a2.withdraw(600) print("-----") a1.withdraw(6929)</pre>	<pre>Default Account 0.0 ----- Oliver 10000.0 ----- Liam 0.0 ----- Noah 400.0 ----- Sorry, Withdraw unsuccessful! The account balance after deducting withdraw amount is equal to or less than minimum. ----- Sorry, Withdraw unsuccessful! The account balance after deducting withdraw amount is equal to or less than minimum. ----- Withdraw successful! New balance is: 3071.0</pre>

Task 13

Implement the design of the **StudentDatabase** class so that the following output is produced:

GPA = Sum of (Grade Points * Credits)/ Credits attempted

Driver Code	Output
<pre># Write your code here # Do not change the following lines of code. 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()</pre>	<pre>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</pre>

Task 14

1	<code>class Test3:</code>
2	<code> def __init__(self):</code>
3	<code> self.sum, self.y = 0, 0</code>
4	<code> def methodA(self):</code>
5	<code> x, y = 2, 3</code>
6	<code> msg = [0]</code>
7	<code> msg[0] = 3</code>
8	<code> y = self.y + msg[0]</code>
9	<code> self.methodB(msg, msg[0])</code>
10	<code> x = self.y + msg[0]</code>
11	<code> self.sum = x + y + msg[0]</code>
12	<code> print(x, y, self.sum)</code>
13	<code> def methodB(self, mg2, mg1):</code>
14	<code> x = 0</code>
15	<code> self.y = self.y + mg2[0]</code>
16	<code> x = x + 33 + mg1</code>
17	<code> self.sum = self.sum + x + self.y</code>
18	<code> mg2[0] = self.y + mg1</code>
19	<code> mg1 = mg1 + x + 2</code>
20	<code> print(x, self.y, self.sum)</code>

Write the output of the following code: <pre> t3 = Test3() t3.methodA() t3.methodA() t3.methodA() t3.methodA() </pre>	x	y	sum

Task 15

1	<code>class Test5:</code>
2	<code> def __init__(self):</code>
3	<code> self.sum, self.y = 0, 0</code>
4	<code> def methodA(self):</code>
5	<code> x = 0</code>
6	<code> z = 0</code>
7	<code> while (z < 5):</code>
8	<code> self.y = self.y + self.sum</code>
9	<code> x = self.y + 1</code>
10	<code> print(x, self.y, self.sum)</code>
11	<code> self.sum = self.sum + self.methodB(x, self.y)</code>
12	<code> z += 1</code>
13	<code> def methodB(self, m, n):</code>
14	<code> x = 0</code>
15	<code> sum = 0</code>
16	<code> self.y = self.y + m</code>
17	<code> x = n - 4</code>
18	<code> sum = sum + self.y</code>
19	<code> print(x, self.y, sum)</code>
20	<code> return self.sum</code>

<p>Write the output of the following code:</p> <p>t5 = Test5()</p> <p>t5.methodA()</p>	x	y	sum

Task 16

1	<code>class FinalT6A:</code>
2	<code>def __init__(self, x, p):</code>
3	<code>self.temp, self.sum, self.y = 4, 0, 1</code>
4	<code>self.temp += 1</code>
5	<code>self.y = self.temp - p</code>
6	<code>self.sum = self.temp + x</code>
7	<code>print(x, self.y, self.sum)</code>
8	<code>def methodA(self):</code>
9	<code>x = 0</code>
10	<code>y = 0</code>
11	<code>y = y + self.y</code>
12	<code>x = self.y + 2 + self.temp</code>
13	<code>self.sum = x + y + self.methodB(self.temp, y)</code>
14	<code>print(x, y, self.sum)</code>
15	<code>def methodB(self, temp, n):</code>
16	<code>x = 0</code>
17	<code>temp += 1</code>
18	<code>self.y = self.y + temp</code>
19	<code>x = x + 3 + n</code>
20	<code>self.sum = self.sum + x + self.y</code>
21	<code>print(x, self.y, self.sum)</code>
22	<code>return self.sum</code>

<p>What is the output of the following code sequence?</p> <pre>q1 = FinalT6A(2,1) q1.methodA() q1.methodA()</pre>	x	y	sum

Task 17

1	<code>class Test5:</code>
2	<code> def __init__(self):</code>
3	<code> self.sum = 0</code>
4	<code> self.y = 0</code>
5	<code> def methodA(self):</code>
6	<code> x=y=k=0</code>
7	<code> msg = [5]</code>
8	<code> while (k < 2):</code>
9	<code> y += msg[0]</code>
10	<code> x = y + self.methodB(msg, k)</code>
11	<code> self.sum = x + y + msg[0]</code>
12	<code> print(x, " ", y, " ", self.sum)</code>
13	<code> k+=1</code>
14	<code> def methodB(self, mg2, mg1):</code>
15	<code> x = 0</code>
16	<code> self.y += mg2[0]</code>
17	<code> x = x + 3 + mg1</code>
18	<code> self.sum += x + self.y</code>
19	<code> mg2[0] = self.y + mg1</code>
20	<code> mg1 += x + 2</code>
21	<code> print(x, " ", self.y, " ", self.sum)</code>
22	<code> return mg1</code>

What is the output of the following code sequence? <pre>t1 = Test5() t1.methodA() t1.methodA() t1.methodA()</pre>	x	y	sum

Task 18

1	<code>class Test4:</code>
2	<code>def __init__(self):</code>
3	<code>self.sum, self.y = 0, 0</code>
4	<code>def methodA(self):</code>
5	<code>x, y = 0, 0</code>
6	<code>msg = [0]</code>
7	<code>msg[0] = 5</code>
8	<code>y = y + self.methodB(msg[0])</code>
9	<code>x = y + self.methodB(msg, msg[0])</code>
10	<code>self.sum = x + y + msg[0]</code>
11	<code>print(x, y, self.sum)</code>
12	<code>def methodB(self, *args):</code>
13	<code>if len(args) == 1:</code>
14	<code>mg1 = args[0]</code>
15	<code>x, y = 0, 0</code>
16	<code>y = y + mg1</code>
17	<code>x = x + 33 + mg1</code>
18	<code>self.sum = self.sum + x + y</code>
19	<code>self.y = mg1 + x + 2</code>
20	<code>print(x, y, self.sum)</code>
21	<code>return y</code>
22	<code>else:</code>
23	<code>mg2, mg1 = args</code>
24	<code>x = 0</code>
25	<code>self.y = self.y + mg2[0]</code>
26	<code>x = x + 33 + mg1</code>
27	<code>self.sum = self.sum + x + self.y</code>
28	<code>mg2[0] = self.y + mg1</code>
29	<code>mg1 = mg1 + x + 2</code>
30	<code>print(x, self.y, self.sum)</code>
31	<code>return self.sum</code>

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

Task 19

1	<code>class msgClass:</code>
2	<code> def __init__(self):</code>
3	<code> self.content = 0</code>
4	<code>class Q5:</code>
5	<code> def __init__(self):</code>
6	<code> self.sum = 1</code>
7	<code> self.x = 2</code>
8	<code> self.y = 3</code>
9	<code> def methodA(self):</code>
10	<code> x, y = 1, 1</code>
11	<code> msg = []</code>
12	<code> myMsg = msgClass()</code>
13	<code> myMsg.content = self.x</code>
14	<code> msg.append(myMsg)</code>
15	<code> msg[0].content = self.y + myMsg.content</code>
16	<code> self.y = self.y + self.methodB(msg[0])</code>
17	<code> y = self.methodB(msg[0]) + self.y</code>
18	<code> x = y + self.methodB(msg[0], msg)</code>
19	<code> self.sum = x + y + msg[0].content</code>
20	<code> print(x, " ", y, " ", self.sum)</code>
21	<code> def methodB(self, mg1, mg2 = None):</code>
22	<code> if mg2 == None:</code>
23	<code> x, y = 5, 6</code>
24	<code> y = self.sum + mg1.content</code>
25	<code> self.y = y + mg1.content</code>
26	<code> x = self.x + 7 + mg1.content</code>
27	<code> self.sum = self.sum + x + y</code>
28	<code> self.x = mg1.content + x + 8</code>
29	<code> print(x, " ", y, " ", self.sum)</code>
30	<code> return y</code>

31	<code>else:</code>
32	<code> x = 1</code>
33	<code> self.y += mg2[0].content</code>
34	<code> mg2[0].content = self.y + mg1.content</code>
35	<code> x = x + 4 + mg1.content</code>
36	<code> self.sum += x + self.y</code>
37	<code> mg1.content = self.sum - mg2[0].content</code>
38	<code> print(self.x, " ", self.y, " ", self.sum)</code>
39	<code> return self.sum</code>

<p>What is the output of the following code sequence?</p> <pre>q = Q5() q.methodA()</pre>	x	y	sum

Practice Task (20 - 25) Ungraded

Task 20

Design a **Student** class so that the following output is produced upon executing the following code

Driver Code	Output
<pre># Write your code here # Do not change the following lines of code. s1 = Student() print("=====") s2 = Student("Carol") print("=====") s3 = Student("Jon", "EEE") print("=====") s1.update_name("Bob") s1.update_department("CSE") s2.update_department("BBA") s1.enroll("CSE110", "MAT110", "ENG091") s2.enroll("BUS101") s3.enroll("MAT110", "PHY111") print("#####") s1.printDetail() print("=====") s2.printDetail() print("=====") s3.printDetail()</pre>	<pre>Student name and department need to be set ===== Department for Carol needs to be set ===== Jon is from EEE department ===== ##### Name: Bob Department: CSE Bob enrolled in 3 course(s): CSE110, MAT110, ENG091 ===== Name: Carol Department: BBA Carol enrolled in 1 course(s): BUS101 ===== Name: Jon Department: EEE Jon enrolled in 2 course(s): MAT110, PHY111</pre>

Task 21

Design a **Student** class so that the following output is produced upon executing the following code:

[Hint: Each course has 3.0 credit hours. You must take at least 9.0 and at most 12.0 credit hours]

Driver Code	Output
<pre># Write your code here # Do not change the following lines of code. s1 = Student("Alice", "20103012", "CSE") s2 = Student("Bob", "18301254", "EEE") s3 = Student("Carol", "17101238", "CSE") print("#####") print(s1.details()) print("#####") print(s2.details()) print("#####") s1.advise("CSE110", "MAT110", "PHY111") print("#####") s2.advise("BUS101", "MAT120") print("#####") s3.advise("MAT110", "PHY111", "ENG102", "CSE111", "CSE230")</pre>	<pre>##### Name: Alice ID: 20103012 Department: CSE ##### Name: Bob ID: 18301254 Department: EEE ##### Alice, you have taken 9.0 credits. List of courses: CSE110, MAT110, PHY111 Status: Ok ##### Bob, you have taken 6.0 credits. List of courses: BUS101, MAT120 Status: You have to take at least 1 more course. ##### Carol, you have taken 15.0 credits. List of courses: MAT110, PHY111, ENG102, CSE111, CSE230 Status: You have to drop at least 1 course.</pre>

Task 22

Write the **Hotel** class with the required methods to give the following output as shown.

Driver Code	Output
<pre># Write your code here # Do not change the following lines of code. h = Hotel("Lakeshore") h.addStuff("Adam", 26) print("=====") print(h.getStuffById(1)) print("=====") h.addGuest("Carol",35,"123") print("=====") print(h.getGuestById(1)) print("=====") h.addGuest("Diana", 32, "431") print("=====") print(h.getGuestById(2)) print("=====") h.allStaffs() print("=====") h.allGuest()</pre>	<pre>Staff With ID 1 is added ===== Staff ID: 1 Name: Adam Age: 26 Phone no.: 000 ===== Guest With ID 1 is created ===== Guest ID: 1 Name: Carol Age: 35 Phone no.: 123 ===== Guest With ID 2 is created ===== Guest ID: 2 Name: Dianal Age: 32 Phone no.: 431 ===== All Staffs: Number of Staff: 1 Staff ID: 1 Name: Adam Age: 26 Phone no: 000 ===== All Guest: Number of Guest: 2 Guest ID: 1 Name: Carol Age: 35 Phone no.: 123 Guest ID: 2 Name: Dianal Age: 32 Phone no.: 431</pre>

Task 23

Write the **Author** class with the required methods to give the following outputs as shown.

Driver Code	Output
<pre># Write your code here # Do not change the following lines of code. a1 = Author() print("=====") a1.addBook("Ice", "Science Fiction") print("=====") a1.setName("Anna Kavan") a1.addBook("Ice", "Science Fiction") a1.printDetail() print("=====") a2 = Author("Humayun Ahmed") a2.addBook("Onnobhubon", "Science Fiction") a2.addBook("Megher Upor Bari", "Horror") print("=====") a2.printDetail() a2.addBook("Ireena", "Science Fiction") print("=====") a2.printDetail() print("=====")</pre>	<pre>===== A book can not be added without author name ===== Number of Book(s): 1 Author Name: Anna Kavan Science Fiction: Ice ===== ===== Number of Book(s): 2 Author Name: Humayun Ahmed Science Fiction: Onnobhubon Horror: Megher Upor Bari ===== Number of Book(s): 3 Author Name: Humayun Ahmed Science Fiction: Onnobhubon, Ireena Horror: Megher Upor Bari =====</pre>

Task 24

Implement the design of the **Hospital, Doctor and Patient** class so that the following output is produced:

Driver Code	Output
<pre># Write your code here # Do not change the following lines of code. h = Hospital("Evercare") d1 = Doctor("1d","Doctor", "Samar Kumar", "Neurologist") h.addDoctor(d1) print("=====") print(h.getDoctorByID("1d")) print("=====") p1 = Patient("1p","Patient", "Kashem Ahmed", 35, 12345) h.addPatient(p1) print("=====") print(h.getPatientByID("1p")) print("=====") p2 = Patient ("2p","Patient", "Tanina Haque", 26, 33456) h.addPatient(p2) print("=====") print(h.getPatientByID("2p")) print("=====") h.allDoctors() h.allPatients()</pre>	<pre>===== Doctor's ID: 1d Name: Samar Kumar Speciality: Neurologist ===== ===== Patient's ID: 1p Name: Kashem Ahmed Age: 35 Phone no.: 12345 ===== ===== Patient's ID: 2p Name: Tanina Haque Age: 26 Phone no.: 33456 ===== All Doctors: Number of Doctors: 1 {'1d': ['Samar Kumar', 'Neurologist']} All Patients: Number of Patients: 2 {'1p': ['Kashem Ahmed', 35, 12345], '2p': ['Tanina Haque', 26, 33456]}</pre>

Task 25

Design the **Vaccine** and **Person** class so that the following expected output is generated.

[N.B: Students will get vaccines on a priority basis. So, age for students doesn't matter]

Driver Code	Output
<pre> # Write your code here astra = Vaccine("AstraZeneca", "UK", 60) modr = Vaccine("Moderna", "UK", 30) sin = Vaccine("Sinopharm", "China", 30) p1 = Person("Bob", 21, "Student") print("=====") p1.pushVaccine(astra) print("=====") p1.showDetail() print("=====") p1.pushVaccine(sin, "2nd Dose") print("=====") p1.pushVaccine(astra, "2nd Dose") print("=====") p1.showDetail() print("=====") p2 = Person("Carol", 23, "Actor") print("=====") p2.pushVaccine(sin) print("=====") p3 = Person("David", 34) print("=====") p3.pushVaccine(modr) print("=====") p3.showDetail() print("=====") p3.pushVaccine(modr, "2nd Dose") </pre>	<pre> ===== 1st dose done for Bob ===== Name: Bob Age: 21 Type: Student Vaccine name: AstraZeneca 1st dose: Given 2nd dose: Please come after 60 days ===== Sorry Bob, you can't take 2 different vaccines ===== 2nd dose done for Bob ===== Name: Bob Age: 21 Type: Student Vaccine name: AstraZeneca 1st dose: Given 2nd dose: Given ===== ===== Sorry Carol, Minimum age for taking vaccines is 25 years now. ===== ===== 1st dose done for David ===== Name: David Age: 34 Type: General Citizen Vaccine name: Moderna 1st dose: Given 2nd dose: Please come after 30 days ===== 2nd dose done for David </pre>