Course Title: Programming Language II
Course Code: CSE 111
Lab Assignment no: 3

Task 1

Suppose your little sibling wants your help to check his math homework. He is done with his homework but wants you to see if all his results are correct. Since the student with all correct results gets 3 stars. However, you want your brother to check this on his own. So, you design a calculator for him in python. You could have given your scientific calculator but you wanted to give him a basic calculator and also wanted to see if you can even design one.

Subtasks:

1. Create a class called Calculator.
2. Your class shall have 1 constructor and 4 methods, namely add, subtract, multiply and divide.
3. Now, create an object of your class. After creating an object, it should print "Let's Calculate!"
4. Then take 3 inputs from the user: first value, operator, second value 5. Now based on the given operator, call the required method and print the result.

Sample Input:

1

+

2

Sample Output:

Let's Calculate!

Value 1: 1

Operator: +

Value 2: 2

Result: 3

Task 2

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

Driver Code	Output
# Write your code here c1 = Course("CSE110", "TBA", 8) c1.detail() print("==============") c2 = Course("CSE111", "TBA", 9) c2.detail()	CSE110 - TBA - 8 ============== CSE111 - TBA - 9

Task 3

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

[For BMI, the formula is BMI = weight/height^2, where weight is in kg and height in meters]

Driver Code	Output
# 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()	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

Task 4

Design a "**Vehicle**" class. A vehicle assumes that the whole world is a 2-dimensional graph paper. It maintains its x and y coordinates (both are integers). Any new object created of the Vehicle class will always start at the coordinates (0,0).

It must have methods to move up, down, left, right and a print_position() method for printing the current coordinate.

Note: All moves are 1 step. That means a single call to any move method changes the value of either x or y or both by 1.

[You are not allowed to change the code below]

# Write your class here	OUTPUT
car = Vehicle() car.print_position() car.moveUp() car.print_position() car.moveLeft() car.print_position() car.moveDown() car.print_position() car.moveRight()	(0,0) (0,1) (-1,1) (-1,0)

Task 5

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

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

Task 6

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

Driver Code	Output

# *Write your code here*	Calculator is ready!
| ```python
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()
``` | ```
==================
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
``` |

Task 7

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

Assume the credit for each course to be 3. For example: [3.3,4] can be calculated as:

CGPA = ((3.3 * 3) + (4 * 3)) / 6

[Here, for each course, the grade point is multiplied by 3. Total credit is the number of courses multiplied by 3. Since the example has 2 courses, therefore a total of 6 credits]

CGPA = sum of individual (grade point * credit) / total credit

Academic Standing Rule: [CGPA>3.80 Highest Distinction, CGPA>3.65 High Distinction, CGPA>3.50 Distinction, CGPA>2.00 Satisfactory, CGPA<2.00 Can't Graduate]

Driver Code	Output
# Write your code here s1 = Student('Dora', '15995599','CSE', [4,3.7,3.7,4]) s1.calculate_CGPA() print("=========================") s1.print_details() print("=========================") s2 = Student('Pingu', '12312322', 'EEE', [1.7,1.3,1.3,1.3,1]) s2.calculate_CGPA() print("=========================") s2.print_details() print("=========================") s3 = Student('Bob', '13311331', 'CSE', [2,3,3,3.7,2.7,2.7]) s3.calculate_CGPA() print("=========================") s3.print_details()	======================= == Name: Dora, ID: 15995599 Department: CSE CGPA: 3.85 Your academic standing is 'Highest Distinction' ======================= == ======================= == Name: Pingu, ID: 12312322 Department: EEE CGPA: 1.32 Sorry, you cannot graduate ======================= == ======================= == Name: Bob, ID: 13311331 Department: CSE CGPA: 2.85 Your academic standing is 'Satisfactory'

Task 8

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

- Write the constructor with appropriate default value for arguments. Set the initial salary and mission to 0.
- Write the changeRank() method with appropriate argument.
- Write the calSalary() method with appropriate argument. Check the following suggestions
  - ➢Update the number of mission from the given argument.
  - ➢If rank == 'Genin' then salary = #mission * 50
  - ➢If rank == 'Chunin' then salary = #mission * 100
  - ➢else salary = #mission * 500
- Write the printInfo() method with appropriate printing.

[You are not allowed to change the code below]

# Write your code here.	OUTPUT:
naruto = Shinobi("Naruto", "Genin") naruto.calSalary(5) naruto.printInfo() print('===================') shikamaru = Shinobi('Shikamaru', "Genin") shikamaru.printInfo() shikamaru.changeRank("Chunin") shikamaru.calSalary(10) shikamaru.printInfo() print('===================') neiji = Shinobi("Neiji", "Jonin") neiji.calSalary(5) neiji.printInfo()	Name: Naruto Rank: Genin Number of mission: 5 Salary: 250 =================== Name: Shikamaru Rank: Genin Number of mission: 0 Salary: 0 Name: Shikamaru Rank: Chunin Number of mission: 10 Salary: 1000 =================== Name: Neiji Rank: Jonin Number of mission: 5 Salary: 2500

Task 9

Design the **Programmer** class 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 prinDetails() method

[You are not allowed to change the code below]

# Write your code here.	OUTPUT:
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()	Horray! A new programmer is born Name: Ethen Hunt Language: Java Experience: 10 years. ------------------------- Horray! A new programmer is born Name: James Bond Language: C++ Experience: 7 years. ------------------------- 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.

Task 10

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
# 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())	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

Task 11

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

Driver Code	Output

<table>
<tr><td>

```
# Write your code here

user1 = Spotify(["See You Again", "Uptown Funk",
"Hello"]) print("==========================")
print(user1.playing_number(4))
user1.add_to_playlist("Dusk Till Dawn")
print(user1.playing_number(3))
print(user1.playing_number(4))
```

</td><td>

Welcome to Spotify!
========================== 4
number song not found. Your playlist
has 3 songs only.
#######################
Playing 3 number song for
you
Song name: Hello
#######################
Playing 4 number song for
you
Song name: Dusk Till Dawn

</td></tr>
</table>

Task 12

1	`class Test:`
2	`  def __init__(self):`
3	`  self.sum = 0`
4	`  self.y = 0`
5	
6	`  def methodA(self):`
7	`  x=0`
8	`  y =0`
9	`  y = y + 7`
10	`  x = y + 11`
11	`  self.sum = x + y`
12	`  print(x , y, self.sum)`

13	
14	`def methodB(self):`
15	`x = 0`
16	`self.y = self.y + 11`
17	`x = x + 33 + self.y`
18	`self.sum = self.sum + x + self.y`
19	`print(x , self.y, self.sum)`

Write the output of the following code:
 	
`t1 = Test()`
`t1.methodA()`
`t1.methodA()`
`t1.methodB()`
`t1.methodB()`
	x	y	sum

Task 13

1	`class Scope:`
2	`def __init__(self):`

```python
3      self.x, self.y = 1, 100

4      def met1(self):

5      x = 3

6      x = self.x + 1

7      self.y = self.y + self.x + 1

8      x = self.y + self.met2() + self.y

9      print(x, self.y)

10     def met2(self):

11     y = 0

12     print(self.x, y)

13     self.x = self.x + y

14     self.y = self.y + 200

15     return self.x + y
```

Write the output of the following code:	X	y
q2 = Scope() q2.met1()		
q2.met2()		
q2.met1() q2.met2()		

Task 14

1	`class Test3:`
2	`def __init__(self):`
3	`self.sum, self.y = 0, 0`
4	`def methodA(self):`
5	`x, y = 2, 3`
6	`msg = [0]`
7	`msg[0] = 3`
8	`y = self.y + msg[0]`
9	`self.methodB(msg, msg[0])`
10	`x = self.y + msg[0]`
11	`self.sum = x + y + msg[0]`
12	`print(x, y, self.sum)`
13	`def methodB(self, mg2, mg1):`
14	`x = 0`
15	`self.y = self.y + mg2[0]`

16	`x = x + 33 + mg1`
17	`self.sum = self.sum + x + self.y`
18	`mg2[0] = self.y + mg1`
19	`mg1 = mg1 + x + 2`
20	`print(x, self.y, self.sum)`

Write the output of the following code:
 	
`t3 = Test3()`
`t3.methodA()`
`t3.methodA()`
`t3.methodA()`
`t3.methodA()`
	x	y	sum

Task 15

1	`class Test5:`
2	`def __init__(self):`
3	`self.sum, self.y = 0, 0`
4	`def methodA(self):`
5	`x = 0`

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

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