

| **Course Code:** | **CSE111** |
| --- | --- |
| **Course Title:** | **Programming Language II** |
| **Homework No:** | **11** |
| **Topic:** | **Inheritance** |
| **Submission Type:** | **Will be notified later on.** |
| **Resources:** | 1. **Class lectures** 2. **BuX lectures**    1. **English:**        1. **Inheritance:** [**here**](https://apps.bux-home.bracu.ac.bd/learning/course/course-v1:buX+CSE111+2023_Summer/block-v1:buX+CSE111+2023_Summer+type@sequential+block@70efe1e3c5b9425b85e1a0fb11fbc579/block-v1:buX+CSE111+2023_Summer+type@vertical+block@361fdbddd216449bb3c0fc18dc52ad71)    2. **Supplementary:**        1. **Inheritance:** [**here**](https://apps.bux-home.bracu.ac.bd/learning/course/course-v1:buX+CSE111+2023_Summer/block-v1:buX+CSE111+2023_Summer+type@sequential+block@70efe1e3c5b9425b85e1a0fb11fbc579/block-v1:buX+CSE111+2023_Summer+type@vertical+block@6061c0ea080344b0a2f6e31682e01b3e) |

**Task 1**

A multinational company has two special types of regular employees. One is Foreign employees and another one is Part time employees. Design the Employee (parent), Foreign\_employee(child) and Parttime\_employee(child) classes so that the following output is produced. The Foreign\_employee and Parttime\_employee classes should inherit the Employee class. Note that:

* Basic salary of a Regular, Foreign employee is 30,000 and for Part-time employees basic is 15,000.
* Regular employees get 10% increment on their salary and Foreign employees get 15% increment on their basic salary.
* Employees from the HR department will collect their work distribution load from the manager, and others will collect their work distribution load from the HR department.

| Driver Code | Output |
| --- | --- |
| print("1------------------------------------------")  emp1=Employee("Nawaz Ali", 102, "Marketing")  print("2------------------------------------------")  emp1.employeeDetails()  print("3------------------------------------------")  emp1.workDistribution("Marketing")  print("4------------------------------------------")  emp1.increment()  emp1.employeeDetails()  print("5------------------------------------------")  f\_emp=Foreign\_employee("Nadvi", 311, "Human Resource")  f\_emp.employeeDetails()  print("6------------------------------------------")  f\_emp.workDistribution("Human Resource")  print("7------------------------------------------")  f\_emp.increment()  f\_emp.employeeDetails()  print("8------------------------------------------")  p1\_emp=Part\_time\_employee("Asif", 210, "Sales")  p2\_emp=Part\_time\_employee("Olive", 223, "Accounts")  print("9------------------------------------------")  p1\_emp.employeeDetails()  print("10------------------------------------------")  p1\_emp.workDistribution("Sales")  print("11------------------------------------------")  p2\_emp.increment()  print("12------------------------------------------")  p2\_emp.employeeDetails() | 1------------------------------------------  2------------------------------------------  Name: Nawaz Ali, Dept Marketing  Employee id: 102, Salary: 30000  3------------------------------------------  Collect work distribution loads from the HR department.  4------------------------------------------  Name: Nawaz Ali, Dept Marketing  Employee id: 102, Salary: 33000.0  5------------------------------------------  Name: Nadvi, Dept Human Resource  Employee id: 311, Salary: 30000  Employee Type: Foreign  6------------------------------------------  Collect work distribution details from the manager.  7------------------------------------------  Name: Nadvi, Dept Human Resource  Employee id: 311, Salary: 34500.0  Employee Type: Foreign  8------------------------------------------  9------------------------------------------  Name: Asif, Dept Sales  Employee id: 210, Salary: 15000  Employee Type: Part Time  10------------------------------------------  Collect work distribution loads from the HR department.  11------------------------------------------  Sadly, there is no increment for the part time employees!!  12------------------------------------------  Name: Olive, Dept Accounts  Employee id: 223, Salary: 15000  Employee Type: Part Time |

**Task 2**

Write the **ScienceExam** class so that the following code generates the output below:

| class Exam:     def \_\_init\_\_(self,marks):         self.marks = marks         self.time = 60              def examSyllabus(self):         return "Maths , English"     def examParts(self):         return "Part 1 - Maths\nPart 2 - English\n"           engineering = ScienceExam(100,90,"Physics","HigherMaths") print(engineering)  print('----------------------------------')  print(engineering.examSyllabus()) print(engineering.examParts())  print('==================================')  architecture = ScienceExam(100,120,"Physics","HigherMaths","Drawing") print(architecture)  print('----------------------------------')  print(architecture.examSyllabus()) print(architecture.examParts()) | ***OUTPUT:***  Marks: 100 Time: 90 minutes Number of Parts: 4  ----------------------------------  Maths , English , Physics , HigherMaths  Part 1 - Maths  Part 2 - English  Part 3 - Physics  Part 4 - HigherMaths  ==================================  Marks: 100 Time: 120 minutes Number of Parts: 5  ----------------------------------  Maths , English , Physics , HigherMaths , Drawing  Part 1 - Maths  Part 2 - English  Part 3 - Physics  Part 4 - HigherMaths  Part 5 - Drawing |
| --- | --- |

**Task 3**

Write the **PokemonExtra** class so that the following code generates the output below:

| class PokemonBasic:    def \_\_init\_\_(self, name = 'Default', hp = 0, weakness = 'None', type = 'Unknown'):      self.name = name      self.hit\_point = hp      self.weakness = weakness      self.type = type    def get\_type(self):      return 'Main type: ' + self.type    def get\_move(self):      return 'Basic move: ' + 'Quick Attack'    def \_\_str\_\_(self):      return "Name: " + self.name + ", HP: " + str(self.hit\_point) + ", Weakness: " + self.weakness  print('\n------------Basic Info:--------------')  pk = PokemonBasic()  print(pk)  print(pk.get\_type())  print(pk.get\_move())  print('\n------------Pokemon 1 Info:-------------')  charmander = PokemonExtra('Charmander', 39, 'Water', 'Fire')  print(charmander)  print(charmander.get\_type())  print(charmander.get\_move())  print('\n------------Pokemon 2 Info:-------------')  charizard = PokemonExtra('Charizard', 78, 'Water', 'Fire', 'Flying', ('Fire Spin', 'Fire Blaze'))  print(charizard)  print(charizard.get\_type())  print(charizard.get\_move()) | ***OUTPUT:***  ------------Basic Info:--------------  Name: Default, HP: 0, Weakness: None  Main type: Unknown  Basic move: Quick Attack  ------------Pokemon 1 Info:--------------  Name: Charmander, HP: 39, Weakness: Water  Main type: Fire  Basic move: Quick Attack  ------------Pokemon 2 Info:--------------  Name: Charizard, HP: 78, Weakness: Water  Main type: Fire, Secondary type: Flying  Basic move: Quick Attack  Other move: Fire Spin, Fire Blaze |
| --- | --- |

**Task 4**

A renowned Bakery shop recently launched cheesecakes into their cakes menu. Cheesecakes will have all the general attributes of the regular cakes but it has some special features. Design the **Cakes** (parent) and **Cheese\_Cakes** (child) classes so that the following output is produced.

Note that:

* 1kg regular cake price is 1200 Taka and 1 kg cheese-cake price is 1500 Taka
* As cheese-cakes are newly launched, they need user feedback. For this reason, if a customer gives feedback on cheese-cakes he'll get 10% discounts on his next purchase.

Write the classes **Cakes** and **Cheese\_Cakes** to generate the following output.

| Driver Code: | Output: |
| --- | --- |
| order\_1=Cakes("Chocolate",500)  order\_2=Cakes("Vanilla",800)  print("(1)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  order\_1.cake\_details()  print("(1.1)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  print(Cakes.order\_list)  print("(2)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  order\_2.add\_customization("Zero","Butter Cream")  order\_2.cake\_details()  print("(3)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  Cakes.give\_feedbacks("Chocolate Cake","Very Delicious")  Cakes.give\_feedbacks("Chocolate Cake","Yummy")  print("(4)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  Cakes.show\_feedbacks()  print("(5)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  ch\_order1=Cheese\_Cakes("Red velvet",700)  ch\_order1.cake\_details()  print("(6)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  ch\_order1.add\_customization()  print("(7)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  ch\_order2=Cheese\_Cakes("Blue Berry",900,"No Bake")  ch\_order2.cake\_details()  print("(8)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  print(Cakes.order\_list)  print("(9)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  Cheese\_Cakes.give\_feedbacks("Red velvet Cheese Cake","Average")  Cheese\_Cakes.give\_feedbacks("Blue Berry Cheese Cake","Liked it")  print("(10)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  Cakes.show\_feedbacks() | (1)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Flavor: Chocolate Cake, Weight: 500 gm  Sweetness: Moderate sugar, Cream Type: Whipped Cream  Price: 600.0 Taka  (1.1)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  {'Chocolate Cake 500gm': 1, 'Vanilla Cake 800gm': 1}  (2)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Flavor: Vanilla Cake, Weight: 800 gm  Sweetness: Zero sugar, Cream Type: Butter Cream  Price: 960.0 Taka  (3)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Thanks for your valuable feedback!  Thanks for your valuable feedback!  (4)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  {'Chocolate Cake': ['Very Delicious', 'Yummy']}  (5)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Flavor: Red velvet Cheese Cake, Weight: 700 gm  Sweetness: Moderate sugar, Cream Type: Cream Cheese  Cake Type:baked, Price: 1050.0 Taka  (6)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Sorry! No customization available for cheese cakes.  (7)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Flavor: Blue Berry Cheese Cake, Weight: 900 gm  Sweetness: Moderate sugar, Cream Type: Cream Cheese  Cake Type:No Bake, Price: 1350.0 Taka  (8)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  {'Chocolate Cake 500gm': 1, 'Vanilla Cake 800gm': 1, 'Red velvet Cheese Cake 700gm': 1, 'Blue Berry Cheese Cake 900gm': 1}  (9)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Thanks for your valuable feedback!  You will get 10% discount for your next purchase!  Thanks for your valuable feedback!  You will get 10% discount for your next purchase!  (10)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  {'Chocolate Cake': ['Very Delicious', 'Yummy'], 'Red velvet Cheese Cake': ['Average'], 'Blue Berry Cheese Cake': ['Liked it']} |

**Task 5**

| **1** | **class A:** |
| --- | --- |
| **2** | **temp = 3** |
| **3** | **def \_\_init\_\_(self):** |
| **4** | **self.sum = 0** |
| **5** | **self.y = 0** |
| **6** | **self.y = A.temp - 1** |
| **7** | **self.sum = A.temp + 2** |
| **8** | **A.temp -= 2** |
| **9** |  |
| **10** | **def methodA(self, m, n):** |
| **11** | **x = 0** |
| **12** | **n[0] += 1** |
| **13** | **self.y = self.y + m + A.temp** |
| **14** | **A.temp += 1** |
| **15** | **x = x + 2 + n[0]** |
| **16** | **n[0] = self.sum + 2** |
| **17** | **print(f"{x} {self.y} {self.sum}")** |
| **18** |  |
| **19** | **class B(A):** |
| **20** | **x = 1** |
| **21** | **def \_\_init\_\_(self, b = None):** |
| **22** | **super().\_\_init\_\_()** |
| **23** | **self.sum = 2** |
| **24** | **if b == None:** |
| **25** | **self.y = self.temp + 1** |
| **26** | **B.x = 3 + A.temp + self.x** |
| **27** | **A.temp -= 2** |
| **28** | **else:** |
| **29** | **self.sum = self.sum + self.sum** |
| **30** | **B.x = b.x + self.x** |
| **31** | **def methodB(self, m, n):** |
| **32** | **y = [0]** |
| **33** | **self.y = y[0] + self.y + m** |
| **34** | **B.x = self.y + 2 +  self.temp - n** |
| **35** | **self.methodA(self.x, y)** |
| **36** | **self.sum = self.x + y[0] + self.sum** |
| **37** | **print(f"{self.x} {y[0]} {self.sum}")** |

**Write the output of the following code:**

| **x = [23]**  **a1 = A()**  **b1 = B()**  **b2 = B(b1)**  **a1.methodA(1, x)**  **b2.methodB(3, 2)**  **a1.methodA(1, x)** | **Output:** | | |
| --- | --- | --- | --- |
| **x** | **y** | **sum** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |