**Deadline to submit practice problems: 18th April**

**Topic: OOP (Before Mid)**

**Coding Problems:**

**Question 1:**

| **Driver Code** | **Output** |
| --- | --- |
| **#Write your code here**  customer= []  customer\_01 = Customer('Sam')  print('-------------------------------')  customer\_01.add\_to\_cart("Chips")  print(customer\_01.show\_cart())  print('-------------------------------')  customer\_01.add\_to\_cart("Mangoes", "Cookies")  print(customer\_01.show\_cart())  print('-------------------------------')  customer\_01.remove\_from\_cart("Chips")  print(customer\_01.show\_cart())  print('-------------------------------')  customer\_01.confirm\_order()  print('-------------------------------')  customer.append(Customer('David'))  print('-------------------------------')  customer[0].add\_to\_cart("Chips", "Cake")  print(customer[0].show\_cart())  print('-------------------------------')  customer[0].add\_to\_cart("Chocolates")  print(customer[0].show\_cart())  print('-------------------------------')  customer[0].remove\_from\_cart("Cake", "Chocolates")  print(customer[0].show\_cart())  print('-------------------------------')  customer[0].confirm\_order()  print('-------------------------------') | Sam just entered the shop  -------------------------------  Items in cart: Chips  -------------------------------  Items in cart: Chips, Mangoes, Cookies  -------------------------------  Items in cart: Mangoes, Cookies  -------------------------------  Sam placed his order at Super Shop.  Thank you for your purchase!  -------------------------------  David just entered the shop  -------------------------------  Items in cart: Chips, Cake  -------------------------------  Items in cart: Chips, Cake, Chocolates  -------------------------------  Items in cart: Chips  -------------------------------  David placed his order at Super Shop.  Thank you for your purchase!  ------------------------------- |

**Question 2:**

| **Driver Code** | **Output** |
| --- | --- |
| **#Write your code here**  c1 = Course('CSE110', 'CSE111')  c2 = Course('CSE474', 'CSE370')  c3 = Course('PHY111', 'MAT110', 'STA201')  t1 = Teacher('Mr X')  t2 = Teacher('Mr Y')  t1.add\_course(c1)  t1.detail()  t2.add\_course(c2)  t2.detail()  t1.add\_course(c2)  t1.detail() | Mr X wants to take the following courses CSE110, CSE111  Mr Y wants to take the following courses CSE474, CSE370  Mr X wants to take the following courses CSE110, CSE111, CSE474, CSE370 |

**Question 3:**

| **Driver Code** | **Output** |
| --- | --- |
| **#Write your code here**  student1 = Student("Carol", 12345, 2017)  student2 = Student("Bob", 78956, 2018)  student3 = Student("Smith", 89123, 2023)  student4 = Student("Harry", 55667, 2021)  student5 = Student("Alice", 23234, 2015)  department1 = Department("CSE")  department2 = Department("BBA")  department1.add\_student(student1)  department1.add\_student(student2)  department1.add\_student(student3)  department2.add\_student(student4)  department2.add\_student(student5)  department = {'Department 1' : department1 , 'Department2' : department2}  print('-------------------------------')  for k,v in department.items():  print(k)  print(v.department\_info())  print('-------------------------------') | -------------------------------  Department 1  Department Name: CSE  Student Info:  Name: Carol  ID: 12345  Enrollment year: 2017  Name: Bob  ID: 78956  Enrollment year: 2018  Name: Smith  ID: 89123  Enrollment year: 2023  <----------------------->  Department2  Department Name: BBA  Student Info:  Name: Harry  ID: 55667  Enrollment year: 2021  Name: Alice  ID: 23234  Enrollment year: 2015  <----------------------->  ------------------------------- |

**Question 4:**

**Note:** You will need at least 3 classes here. You can create more classes if needed without changing the driver code. Other than the methods in the driver code, you can create new methods inside each class if needed as well and call those from inside the class code. (Again, you must not change the driver code.)   
  
***But your code must be dynamic/ should not be driver code specific. It should work for all cases. (To know if your code is dynamic, try creating more Book objects, Reader objects, call the methods more times and see if your code still works.)***

You can consider this to be a *mini project*. If you can solve this, kudos to you! If not, it’s completely fine as well.

| **Driver Code** | **Output** |
| --- | --- |
| **#Write your code here**  Book1 = Book("Harry Potter and the Goblet of Fire", 2000, "J. K. Rowling")  Book2 = Book("If Tomorrow Comes", 1985, "Sidney Sheldon")  Book3 = Book("Diary of a Wimpy Kid", 2007, "Jeff Kinney")  Book4 = Book("Sapiens: A Brief History of Humankind", 2011, "Yuval Noah Harari")  directory = BookDirectory()  directory.add\_book(Book1)  directory.add\_book(Book2)  directory.add\_book(Book3)  directory.add\_book(Book4)  Reader1 = Reader("Bob")  Reader1.add\_review(Book1, 5, "The best book I ever read!")  Reader1.add\_review(Book2, 4, "One of the best thrillers")  Reader2 = Reader("Carol")  Reader2.add\_review(Book1, 4, "Love a good magical story!")  Reader2.add\_review(Book3, 3, "I somewhat liked the book but the genre is not my type.")  Reader2.add\_review(Book4, 5, "Beautifully written.")  Reader3 = Reader("Harry")  Reader3.add\_review(Book4, 5, "What a fantastic book!")  directory.add\_reader(Reader1)  directory.add\_reader(Reader2)  directory.add\_reader(Reader3)  highest\_rated\_books = directory.get\_highest\_rated\_books(2)  print('-------------------------------')  print("Highest rated books:")  for book in highest\_rated\_books:  print(book)  print('-------------------------------')  highest\_rated\_books = directory.get\_highest\_rated\_books(3)  print('-------------------------------')  print("Highest rated books:")  for book in highest\_rated\_books:  print(book)  print('-------------------------------') | -------------------------------  Highest rated books:  Sapiens: A Brief History of Humankind (2011) by Yuval Noah Harari  Harry Potter and the Goblet of Fire (2000) by J. K. Rowling  -------------------------------  -------------------------------  Highest rated books:  Sapiens: A Brief History of Humankind (2011) by Yuval Noah Harari  Harry Potter and the Goblet of Fire (2000) by J. K. Rowling  If Tomorrow Comes (1985) by Sidney Sheldon  ------------------------------- |

**Topic: OOP (After Mid)**

**Type: Design Problem**

You need to design both the driver code and class code by yourself for the following tasks.

**Note: Your code must work for all possible scenarios. You should call the methods atleast once to show it properly works. Read the instructions carefully before coding as the classes are linked with one another.**

**Question 5:**

Design a Restaurant Management System consisting of two classes: *Menu and Customer*. Note that you have the freedom to create more classes and methods as you wish but you need to tentatively follow the following steps:

**Menu Class:**

1. The Menu class should store information about the name of the menu and about available food items for each menu. Note that there can be several menus meaning different objects of Menu class. Each menu can carry different food items.
2. Menu class should have the following methods:
3. add\_food\_items() to add any food item available on the menu.
4. remove\_food\_items() to remove any food item on the menu.
5. show\_menu() to show the items on the menu.

**Customer Class:**

1. Customer class should store the name of the customer only.
2. A Customer can select which menu he wants and that menu should be shown to him from the Menu class. You should take user input from the customer for this.

*You can do this in the driver code or design methods inside the Customer class.*

The output should be like the following:

Dear David,  
The menu you selected contains the following items:

Beef Burger

Chicken Cordon Bleu

NewYork Cheescake

**Question 6:**

You all are familiar with USIS. You need to create a version of USIS called AXIS using OOP concepts. You will need the concept of ***class variables*** in order to solve this. Note that you have the freedom to create more classes and methods as you wish but you need to tentatively follow the following steps:

**AXIS Class:**

1. Create a class named AXIS which students should register in to add courses using the Student class. *If a student is not registered in AXIS, they will not be allowed to add any courses using the Student class*.
2. The AXIS class should keep a list of registered students. **(Hint:** Use class variable)
3. The AXIS class should have the following methods:
4. register() that takes the name, email and password of a student to register.
5. register\_info() that shows if the Student is registered or not.
6. show\_students\_info() that shows the total number of Students who registered in AXIS, number of students who were able to take courses this semester, number of students who will get scholarships and variations of scholarships (like how many students will get 75%, 50%, 25%, 10% scholarship).

Ex: 10 students registered to AXIS, but 8 were able to add courses. Among them 5 might have received scholarships. 2 might have gotten 25%, 1 might have gotten 10%, 2 might have gotten 50% scholarships.:

**Student Class:**

1. Create a class named Student that stores the name, ID, CGPA, current semester, courses taken in the current semester of a student and his/ her scholarship information. Whether the student will get a scholarship or not is to be calculated but the rest of the information can be manually given.
2. The Student class should also keep track of the total number of Students who are able to take courses in this semester. (**Hint:** Use class variable)
3. The Student class should also keep track of the number of Students receiving scholarships and variations of scholarships. Ex: If 3 students will receive 75% scholarship or if 5 students will receive 50% scholarship etc. (**Hint:** Use class variable)
4. The Student class should contain the following methods:
5. add\_courses() thattakes a list of courses as parameters and *checks if a student is registered in AXIS or not first to allow him/ her to take courses*.

If registered, they can add a maximum of five courses if CGPA > 3.5 and a minimum of two courses. Else, if CGPA >= 1.5 but CGPA <3.5, they can add a maximum of four courses and a minimum of two courses. If CGPA < 1.5, they are not allowed to add courses this semester.

1. scholarship\_calculation() that calculates how much scholarship a student will receive based on their CGPA this semester. The criteria for scholarship is as follows:

| CGPA Range | Scholarship (%) |
| --- | --- |
| 4.00 | 100% |
| 3.95 - 3.99 | 75% |
| 3.90 - 3.94 | 50% |
| 3.85 - 3.89 | 25% |
| 3.80 - 3.84 | 10% |
| < 3.80 | No scholarship |

1. student\_details() that show the name, ID, CGPA, current semester, courses taken in the current semester of the student, whether she/ he will get a scholarship or not and if they do, how much scholarship. It should also show if the student could not add any course this semester.

**Inheritance, Variable and Method Overriding**

**Tracing Practice Tasks:**

**Task 1:**

**class A:**

**temp = 5**

**def \_\_init\_\_(self, x):**

**self.y = self.temp + x[0]**

**self.sum = A.temp - 2**

**self.temp -= 1**

**self.methodB(self.y, self.temp, x)**

**def methodB(self, m, n, z):**

**y = m + self.y + z[0]**

**self.sum = self.temp + y + n**

**self.temp = self.temp + A.temp**

**print(y, self.sum, self.temp)**

**class B(A):**

**x = 3**

**y = 4**

**temp = 10**

**def methodB(self, m, n, z):**

**z[0] -= 4**

**self.temp = self.temp + z[0]**

**A.temp = m + self.temp + n**

**print(self.temp, A.temp, self.sum)**

**self.sum = B.x + B.y + A.temp**

**B.x = B.x + self.y - n**

**B.y += z[0]**

**print(self.sum, B.x, B.y)**

**super().methodB(z[0], 12, z)**

**t = [5]**

**a1 = A(t)**

**b1 = B(t)**

**Task 2:**

**class A:**

**def m1(self):**

**print("Inside class A m1")**

**self.m4()**

**class B(A):**

**def m1(self):**

**super().m1()**

**print("Kothaye ashlaam?")**

**def m3(self):**

**self.m2()**

**print("Inside class B m3")**

**def m2(self):**

**print("Are you sure I am supposed to be here?")**

**def m4(self):**

**print("Inside class B m4")**

**class C(B):**

**def m1(self):**

**self.m3()**

**print("Inside class C m1")**

**def m2(self):**

**super().m1()**

**print("bhul")**

**c1 = C()**

**c1.m1()**