United International University
B.Sc. in Data Science

DS 1115: Object-Oriented Programming for Data Science

Mid Exam: Spring 2024 Time: 1 Hour 30 Minutes Marks: 30

Answer all of the following questions.

1. (a) Your university is opening a new banking system and they are asking for your help. Read the instructions given below and design the system following Object-Oriented Principles.

   i. There should be a class `Account` which will have a unique ID starting from 1000 and each time an account gets created the ID will increase by one. [3]

   ii. Each Account will have a list of `Transactions`, a starting `balance` of 500 and a `username`. An `Account` can have two types of transactions: withdraw and deposit. Each `Transaction` will have an `amount`, a `date`, and a `type`. Every time *deposit()* or *withdraw()* gets called, the account will store a new transaction. [3]

   iii. No account should be able to withdraw a greater amount than its balance. [1]

   (b) Consider the following program and find the exact output: [3]

```
from abc import ABC, abstractmethod
import copy


class Ride(ABC):
    def __init__(self, base_fare, per_km_fare):
        self.base_fare = base_fare
        self.per_km_fare = per_km_fare
    @abstractmethod
    def calculateFare(self, distance, greeting):
        greeting()
        return self.base_fare + self.per_km_fare * distance
```

```
class Car(Ride):
    def __init__(self,base_fare, per_km_fare):
        super().__init__(base_fare, per_km_fare)

    def calculateFare(self, distance, greeting):
        total_fare = super().calculateFare(distance, greeting)

        if(distance > 10):
            return total_fare * 0.9
        else:
            return total_fare
def welcomeUser():
    print("Welcome to Uthao")
    print("Here is your fare breakdown")

variable = welcomeUser
car1 = Car(40, 15)
car1_copy = car1
car1_copy.base_fare = 45

car2 = Car(50, 20)
car2_copy = copy.deepcopy(car2)
car2_copy.base_fare = 55

print(car1_copy.calculateFare(14 , variable))
print(car2_copy.calculateFare(14 , variable))
```

2. Imagine you are a software architect working on developing a complex online finance management system named "FinTechMaster." As part of the design, you are incorporating object-oriented programming principles with a focus on inheritance.

   (a) Design a base class named `User` with common attributes such as `username`, `password`, and `email`. This `User` class will have a method named `change_password(new_password)` that will check whether the old password matches the newly given one; if not, the user will be allowed to set the password, otherwise not. [3]

   (b) Create a child class named `Customer`. Specify attributes unique to the `Customer` class, such as `account_balance` and `transaction_history`. The `Customer` class should have a method named `check_balance()` that returns the customer's balance. Create another child class named `Employee`. Define attributes specific to the `Employee` class, such as `employee_id` and `department`. The `Employee` class should have a method `generate_report()`. You can print "Report generated" to implement this method. [3]

   (c) Add a new base class named `VIPStatus`, which holds an attribute `loyalty_points`. You now need to create a class named `VIPCustomer` with properties of both `Customer` and `VIPStatus`. What type of inheritance should you use in this case? What attributes and methods does this `VIPCustomer` class have? [4]

3. (a) Consider this simple class as in the following.

```
class Student:
    def __init__(self):
        self.studentid=0
        self.name=""
        self.completed_credits=0
        self.cgpa=0
```

   The program coordinator of the department has a task to sort the students according

to CGPA and then completed_credits. You have to overload."greater than" operator to help in this task. Please rewrite the class with this added functionality. [5]

(b) Find the output of the following program. [3]

```
class A:
    def __init__(self,a,b):
        self.a=a
        self.b=b
    def copy(self):
        return A(self.a,self.b)
    def __str__(self):
        return f"({self.a},{self.b})"
    def multiply(self,n):
        self.a=self.a*n
        self.b=self.b*n
s1 = A(12,13)
s2 = s1
s3 = s2.copy()
s1.multiply(2)
s2.multiply(2)
s3.multiply(2)
print(s1,s2,s3)
```

(c) What is the difference between __str__() and __repr__() methods? Explain with an example. [2]