# United International University
## B.Sc. in Data Science

DS 1115: Object-Oriented Programming for Data Science

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

**Any examinee found adopting unfair means will be expelled from the trimester/program as per UIU disciplinary rules.**

1. Inter-department cricket matches are arranged every year in UIU. This year, the students of CSE has planned to develop a simple scoreboard application to keep track of the scores. Read the scenario carefully and design the application. You are expected to write Python code.

   A cricket match has many players. Each player has a *name* which can be stored as a string, a *run* which is an integer storing how many points a player has scored, and a boolean *out* which is set to *True* if the player has been dismissed. The player class has a method *update_run(runs)* which takes a parameter *runs* as input and adds it to the player's current run. For example, if a player currently scored 25 runs, calling the method *update_run(5)* will take the total runs to 30.

   Each team consists of 11 players. The team must have a *name* and a list of *players*. There should be a method *add_player(player)* that takes a player object and adds him to the team. You can also remove a player from the team using a *remove_player(player)* method. There should be a method *get_total_runs()* that returns the team's total run. The team's total run is simply the sum of all the players' runs. There should also be a method named *get_highest_scorer()* that returns the object of the player who has scored the most runs.

   Finally, there should be a function named *get_results(team1, team2)* which **does NOT belong to any class**. This function will take two team objects, determine the winner based on who scored more runs, and also print the name of the highest scorer of the winning team. You can print something simple like "Team-1 is the winner and Player-1 is their highest scorer." [10]

2. (a) You are tasked with designing your own Date class. The class will have only 3 parameters: day, month, and year. Overload the following operators based on the requirements:

i. __add__(self, other): Takes a Date object and an integer representing the number of days to be added to the given date, and returns a new Date object. After addition, if the number of days exceed the total number of days for that month, increase the value of the month. Again if the number of months exceeds 12, add 1 to the year. For simplicity, you may assume that *other* will be an integer between 1 and 25. [3]

ii. __sub__(self, other): Takes two Date objects and returns an integer indicating the difference between the two dates. For example, the difference between "9/11/24" and "5/11/24" is 4. The difference between "9/11/24" and "10/12/24" is 31. For simplicity, you may assume that the two dates are from the same year. [3]

iii. __eq__(self, other): Takes two Date objects and returns *True* if the corresponding day, month, and year are the equal. [2]

(b) Find the output of the following program. [2]

```
class Point3D:
    def __init__(self, x, y, z):
        self.x=x
        self.y=y
        self.z=z
    def copy(self):
        return Point3D(self.x, self.y, self.z)
    def __str__(self):
        return f"({self.x},{self.y}, {self.z})"
    def scale(self, n):
        self.x *= self.x
        self.y *= self.y
        self.z *= self.z
```

```
p1 = Point3D(1, 2, 3)
p2 = p1
p3 = p2.copy()
p1.scale(2)
p2.scale(2)
p3.scale(2)
print(p1, p2, p3)
```

(1,1) (4,4) (3,3)

3. (a) You are tasked with creating a class hierarchy to represent different types of smart devices in a smart home management system. The system should support various types of devices such as lights and security cameras. Write code for the following classes:

   - Define an abstract base class **SmartDevice** with the attributes *name*, *device_id*, and *status* which can be either "on" or "off". There are two abstract methods *calculate_energy_consumption(hours)* and *get_device_name()*. [3]
   - **SmartLight** is a concrete subclass of SmartDevice with additional attribute *brightness_level*. To get the energy consumption of the light, multiply the number of hours it has been used for with 0.5. [1]
   - **SmartSecurityCamera** is a concrete subclass of SmartDevice with additional attribute *resolution*. To get the energy consumption of the camera, multiply the number of hours it has been used for with 0.75. [1]

(b)
```
from abc import ABC, abstractmethod
class A(ABC):
    def __init__(self, x, y):
        self.x=x
        self.y=y
    @abstractmethod
    def fx(self, x):
```

```
        pass
class B(A):
    def __init__(self, x, y, z):
        super().__init__(x, y)
        self.z = z
    def fy(self, y):
        self.y = y
    def fz(self, z):
        self.z = z

b = B(1, 2, 3)
```

Running this code will raise an error. What is the error? What is the reason behind this error? How will you fix it (rewrite the code including the correction)? [1+2+2]