# Problem 1: Student and Course Management

# Question:

# Implement a simple student and course management system using OOP. Create classes ‘Student’ and ‘Course’. The ‘Student’ class should have attributes for student details (name, ID, etc.) and a method to calculate the GPA. The ‘Course’ class should have a list of enrolled students and methods to add a student, remove a student, and calculate the average GPA of all students in the course.

class Student:

def \_\_init\_\_(self, name, student\_id):

self.name = name

self.student\_id =student\_id

self.courses = {}

def enroll(self, course):

self.courses[course.course\_id] = course

course.add\_student(self)

def calculate\_gpa(self):

total\_grade\_points= 0

total\_credits = 0

for course\_id, course in self.courses.items():

total\_grade\_points += course.grade\_points \* course.credits

total\_credits += course.credits

if total\_credits == 0:

return 0

else:

return total\_grade\_points/total\_credits

def display\_courses(self):

print(f"Courses enrolled by {self.name}: ")

for course\_id, course in self.courses.items():

print(f"{course.title} (ID:{course\_id})")

class Course:

def \_\_init\_\_(self, title, course\_id):

self.title = title

self.course\_id = course\_id

self.students = {}

self.credits = credits

self.grade\_points = grade\_points

def add\_student(self, student):

self.students[student.student\_id] = student

def remove\_student(self, student):

if student\_id in self.students:

del self.students[student\_id]

def display\_students(self):

print(f"Students enrolled in {self.title}: ")

for student\_id, student in self.students.items():

print(f"{student.name} (ID:{student\_id})")

# Problem 2: Library System

# Question:

# Design a simple library system using OOP. Create classes ‘Book’, ‘ Author’, and ‘Library’. The ‘Book’ class should have attributes for book details (title, ISBN, etc.). The ‘Author’ class should have attributes for author details (name, birthdate, etc.). The ‘Library’ class should manage a collection of books and authors, allowing users to borrow and return books.

class Author:

def \_\_init\_\_(self, name, birthdate):

self.name = name

self.birthdate = birthdate

def \_\_str\_\_(self):

return f"Author: {self.name} (Born {self.birthdate})"

class Book:

def \_\_init\_\_(self, title, author, isbn):

self.title = title

self.author = author

self.isbn = isbn

self.is\_borrowed = False

def \_\_str\_\_(self):

return f"Title: {self.title}\nAuthor: {self.author}\nISBN: {self.isbn}\nBorrowed: {'Yes' if self.is\_borrowed else 'No'}"

class Library:

def \_\_init\_\_(self):

self.books = {}

def add\_book(self, book):

self.books[book.isbn] = book

def remove\_book(self, isbn):

if isbn in self.books:

del self.books[isbn]

print(f"Book with ISBN {isbn} removed from the library")

else:

print(f"No Book with ISBN {isbn} found in the library")

def display\_books(self):

print('Books available in the library:')

for isbn, book in self.books.items():

print(book)

def borrow\_book(self, isbn):

if isbn in self.books:

if not self.books[isbn].is\_borrowed:

self.books[isbn].is\_borrowed = True

print(f"Book with ISBN {isbn} borrowed successfully")

else:

print(f"Book with ISBN {isbn} is already borrowed.")

else:

print(f"No book found with ISBN {isbn}.")

def return\_book(self, isbn):

if isbn in self.books:

if self.books[isbn].is\_borrowed:

self.books[isbn].is\_borrowed = False

print(f"Book with ISBN {isbn} returned successfully")

else:

print(f"Book with ISBN {isbn} is not borrowed.")

else:

print(f"No book found with ISBN {isbn}.")

# Problem 4: Zoo Management

# Question:

# Create a zoo management system using OOP. Design classes for ‘Animal’, ‘Zoo’, and specific animal types like ‘Lion’, ‘Elephant’, and ‘Giraffe’. The ‘Animal’ class should have attributes for animal details, and each specific animal type should have a method to make a sound. The ‘Zoo’ class should manage a collection of animals and have a method to make all animals in the zoo make their respective sounds.

class Animal:

def \_\_init\_\_(self, name):

self.name = name

def make\_sound(self):

pass

class Lion(Animal):

def make\_sound(self):

return f"ROAR!"

class Elephant(Animal):

def make\_sound(self):

return f"elephant nosie here"

class Giraffe(Animal):

def make\_sound(self):

return f"giraffe noise here"

class Zoo:

def \_\_init\_\_(self):

self.animals = []

def add\_animal(self, animal):

self.animals.append(animal)

def display\_animals(self):

print("Animals in the Zoo:")

for animal in self.animals:

print(f"{animal.name}: {animal.make\_sound()}")

zoo = Zoo()

giraffe = Giraffe("Girafarig")

lion = Lion("Entei")

elephant = Elephant("Hollyphant")

zoo.add\_animal(giraffe)

zoo.add\_animal(lion)

zoo.add\_animal(elephant)

zoo.display\_animals()