practice coding questions based on Object-Oriented Programming (OOP) principles in Python:

Computer Programming-II (BCSC 0163)

Question 1: Shape Hierarchy

Create a simple hierarchy of classes representing different shapes. The base class is Shape, and it should have a method area that is overridden by the derived classes.

```
class Shape:
        def area(self):
                pass
class Rectangle(Shape):
        def __init__(self, length, width):
                self.length = length
                self.width = width
        def area(self):
                return self.length * self.width
class Circle(Shape):
        def __init__(self, radius):
                self.radius = radius
        def area(self):
                return 3.14 * self.radius ** 2
# Test the classes
rectangle = Rectangle(5, 10)
circle = Circle(7)
print(f"Rectangle Area: {rectangle.area()}")
print(f"Circle Area: {circle.area()}")
```

GitHub Repo-Link https://github.com/amirkhan1092/Batch2023-24

Question 2: Book Library System

Create a simple library system with classes for Book, Author, and Library. Each book has an author, and the library can display all books.

```
class Author:
    def __init__(self, author_name):
        self.author_name = author_name
class Book:
    def __init__(self, title, author, ISBN):
        self.title = title
        self.author = author
        self.ISBN = ISBN
class Library:
    def __init__(self):
        self.books = []
    def add_book(self, book):
        self.books.append(book)
    def display_books(self):
        for book in self.books:
            print(f"Title: {book.title}, Author: {book.author.author_name}, ISBN: {book.ISBN}")
# Test the classes
author1 = Author("Jane Doe")
book1 = Book("Python Basics", author1, "123456789")
book2 = Book("Data Structures", author1, "987654321")
library = Library()
library.add_book(book1)
library.add_book(book2)
```

GitHub Repo-Link https://github.com/amirkhan1092/Batch2023-24

```
library.display_books()
```

Question 3: School Management System

Create a school management system with classes for Student, Teacher, and Course. Students can enroll in courses taught by teachers.

```
class Teacher:

def __init__(self, teacher_name, subject):

self.teacher_name = teacher_name

self.subject = subject

class Student:

        def __init__(self, student_name, student_id):

                self.student_name = student_name

                self.student_id = student_id

                self.courses_enrolled = []

        def enroll_course(self, course):

                self.courses_enrolled.append(course)

class Course:

        def __init__(self, course_name, teacher):

                self.course_name = course_name

                self.teacher = teacher

# Test the classes

teacher = Teacher("Mr. Smith", "Math")

student1 = Student("Alice", "S001")

student2 = Student("Bob", "S002")

math_course = Course("Mathematics 101", teacher)
```

GitHub Repo-Link https://github.com/amirkhan1092/Batch2023-24

```
physics_course = Course("Physics 101", Teacher("Mrs. Johnson", "Physics"))

student1.enroll_course(math_course)

student2.enroll_course(physics_course)

print(f"{student1.student_name} is enrolled in {math_course.course_name} taught by {teacher.teacher_name}")

print(f"{student2.student_name} is enrolled in {physics_course.course_name} taught by
{physics_course.teacher.teacher_name}")
```

These questions cover the basics of OOP in Python, including inheritance, polymorphism, and encapsulation. Feel free to modify and expand upon these examples to further enhance your understanding of object-oriented concepts in Python

GitHub Repo-Link https://github.com/amirkhan1092/Batch2023-24