Lab:07.

Q1: Create a Book class and a Library class. The Library class should have methods to add a book, display all books, and search for a book by title.

Program:

```
class Book:
    def __init__(self, title, author, ISBN):
        self.title = title
        self.author = author
        self.ISBN = ISBN

    def __str__(self):
        return f"{self.title} by {self.author} (ISBN: {self.ISBN})"



class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        self.books.append(book)
        print(f"Book '{book.title}' added to the library.")

    def display_books(self):
        if not self.books:
            print("No books in the library.")
        else:
            print("Books in the library:")
            for book in self.books:
                print(book)
```

```
    def search_by_title(self, title):
        found_books = [book for book in self.books if book.title.lower() == title.lower()]
        if found_books:
            print(f"Found {len(found_books)} book(s) with the title '{title}':")
            for book in found_books:
                print(book)
        else:
            print(f"No books found with the title '{title}'.")



if __name__ == "__main__":
    library = Library()

    while True:
        print("\nLibrary Management System")
        print("1. Add a Book")
        print("2. Display All Books")
        print("3. Search for a Book by Title")
        print("4. Exit")

        choice = input("Enter your choice here: ")

        if choice == "1":
            title = input("Enter the title of the book: ")
            author = input("Enter the author of the book: ")
            ISBN = input("Enter the ISBN of the book: ")
            new_book = Book(title, author, ISBN)
            library.add_book(new_book)
```

```
elif choice == "2":
    library.display_books()

elif choice == "3":
    search_title = input("Enter the title of the book to search: ")
    library.search_by_title(search_title)

elif choice == "4":
    print("Exiting the program. Goodbye!")
    break

else:
    print("Invalid choice. Please enter a number between 1 and 4.")
```

Q2: Create a simple car rental system with classes for Car, Customer, and Rental. Implement methods to rent a car, return a car, and display the list of rented cars.

Program:

```
class Car:
    def __init__(self, car_id, make, model, year, available=True):
        self.car_id = car_id
        self.make = make
        self.model = model
        self.year = year
        self.available = available

    def __str__(self):
```

```
        status = "Available" if self.available else "Rented"
        return f"{self.year} {self.make} {self.model} (ID: {self.car_id}) - {status}"


class Customer:
    def __init__(self, customer_id, name):
        self.customer_id = customer_id
        self.name = name


class Rental:
    def __init__(self):
        self.cars = []
        self.customers = []
        self.rented_cars = []

    def add_car(self, make, model, year):
        car_id = len(self.cars) + 1
        new_car = Car(car_id, make, model, year)
        self.cars.append(new_car)
        print(f"Car '{new_car}' added to the rental system.")

    def display_available_cars(self):
        available_cars = [car for car in self.cars if car.available]
        if not available_cars:
```

```
            print("No cars are currently available.")
        else:
            print("Available Cars:")
            for car in available_cars:
                print(car)

    def rent_car(self, customer, car_id):
        self.display_available_cars()
        car_to_rent = next((car for car in self.cars if car.car_id == car_id and
car.available), None)
        if car_to_rent:
            car_to_rent.available = False
            self.rented_cars.append((customer, car_to_rent))
            print(f"{customer.name} has rented {car_to_rent}")
        else:
            print("Error: Invalid car selection or the car is not available.")

    def return_car(self, car_id):
        car_to_return = next((car for _, car in self.rented_cars if car.car_id ==
car_id), None)
        if car_to_return:
            car_to_return.available = True
            customer, _ = next((cust, car) for cust, car in self.rented_cars if
car.car_id == car_id)
            print(f"{customer.name} has returned {car_to_return}")
            self.rented_cars.remove((customer, car_to_return))
```

```
        else:
            print("Error: Car not found or not rented.")

    def display_rented_cars(self):
        if not self.rented_cars:
            print("No cars are currently rented.")
        else:
            print("Rented Cars:")
            for customer, car in self.rented_cars:
                print(f"{customer.name} - {car}")


if __name__ == "__main__":
    rental_system = Rental()

    while True:
        print("\nCar Rental System")
        print("1. Add a Car")
        print("2. Rent a Car")
        print("3. Return a Car")
        print("4. Display Rented Cars")
        print("5. Exit")

        choice = input("Enter your choice here: ")
```

```
        if choice == "1":
            make = input("Enter the make of the car: ")
            model = input("Enter the model of the car: ")
            year = input("Enter the year of the car: ")
            rental_system.add_car(make, model, year)

        elif choice == "2":
            customer_name = input("Enter your name: ")
            rental_system.display_available_cars()
            car_id = int(input("Enter the ID of the car you want to rent: "))
            customer = Customer(len(rental_system.customers) + 1,
customer_name)
            rental_system.rent_car(customer, car_id)

        elif choice == "3":
            car_id = int(input("Enter the ID of the car you want to return: "))
            rental_system.return_car(car_id)

        elif choice == "4":
            rental_system.display_rented_cars()

        elif choice == "5":
            print("Exiting the program. Goodbye!")
            break

        else:
```

```
print("Invalid choice. Please enter a number between 1 and 5.")
```