Air University
(Mid-Term Examination: Spring 2024)

Subject:	**Object Oriented Programming Lab**	Total Marks:	**30**
Course Code:	**CS-112L**	Date:	
Class:	**BS-CYS-**	Time:	
Semester:	**2**	Duration:	**1 Hours**
Section:	**B**	FM Name:	**Mr. Mahaz Khan**

HoD Signatures:__________________________ FM Signatures:______________________

Note:
- **All questions must be attempted.**
- **This examination carries 15% weight towards the final grade.**
- **Submit source file of all 3 questions and also compile a complete report in MS Word.**

	Q. No. 1 (CLO)	**20 Marks**
A	You are tasked with implementing a C++ program that demonstrates the concepts of **composition and aggregation** in object-oriented programming. The program should define classes for representing trains, passengers, and ticket bookings. • The Ticket and Passenger classes should **showcase composition**. • The Train and Ticket classes should **demonstrate aggregation**. **Tasks:** **Implement the Train class with the following specifications:** • **Private data members:** ○ trainNumber (integer). ○ destination (string). ○ availableSeats (integer). • **Constructor:** Accepts parameters to initialize attributes. • **Accessor functions:** Separate get functions for each data member. • **Overload the - operator** to decrement seat count upon booking. **Define the Passenger class with the following specifications:** • **Private data members:** ○ passengerID (integer). ○ name (string).	

  - age (integer).

- **Constructor:** Accepts parameters.
- **Accessor function:** Get function for each data member.

Create the Ticket class with the following specifications:

- **Private data members:**
  - **Composition:** A Passenger object (each ticket has a permanently associated passenger).
  - **Aggregation:** A Train object (a ticket is linked to a train but does not own it).
  - ticketPrice (floating-point number).
- **Constructor:** Accepts parameters to initialize attributes.
- **Accessor function:** getTicketDetails() to return ticket details.
- **Overload the += operator** to add a discount to the ticket price.

In the main() function:

- Create trains and passengers.
- Book tickets for passengers, reducing available seats using operator-.
- Apply a discount to a ticket using operator+=.
- Display ticket details.

B **Analyze the given C++ program and identify any logical, syntactical, or structural errors.**

```
#include <iostream>
using namespace std;

// Account class
class Account {
private:
   int accountNumber;
   float balance;

public:
   // Default Constructor
    {
      accountNumber = 0;
      balance = 0.0;
   }

   // Parameterized Constructor
```

```
    Account(int accNum, float bal) {
        accountNumber = accNum;
        balance = bal;
    }

    // Destructor
    ~Account() {
        cout << "Account " << accountNumber << " is closed.\n";
    }

    // Deposit method
    void deposit(float amount) {
        balance += amount;
        cout << "Deposited: $" << amount << "\n";
    }

    // Withdraw method with balance check
    void withdraw(float amount) {
        if (balance >= amount) {
            balance -= amount;
            cout << "Withdrawn: $" << amount << "\n";
        }
{
            cout << "Insufficient balance!\n";
        }
    }

    // Display account details
    void display() const {
        cout << "Account Number: " << accountNumber << "\nBalance: $" << balance <<
"\n";
    }
};

// SavingsAccount class (Standalone)
class SavingsAccount {
private:
    int accountNumber;
    float balance;
    float interestRate;

public:
    // Default Constructor
    SavingsAccount() {
        accountNumber = 0;
        interestRate = 0.0;
    }

    // Parameterized Constructor
    SavingsAccount(int accNum, float bal, float rate) {
        accountNumber = accNum;
        balance = bal;
        interestRate = rate;
    }
    // Destructor
    ~SavingsAccount() {
        cout << "Savings Account " << accountNumber << " is closed.\n";
    }

    // Apply interest to balance
    void applyInterest() {
        float interest = balance * (interestRate / 100);
        balance += interest;
        cout << "Interest Added: $" << interest << "\n";
    }

    // Display account details
    void display() const {
        cout << "Savings Account Number: " << accountNumber << "\nBalance: $" <<
balance
             << "\nInterest Rate: " << interestRate << "%\n";
    }
};

// CheckingAccount class (Standalone)
class CheckingAccount {
private:
    int accountNumber;
    float balance;
    float transactionFee;

public:
    // Default Constructor
    CheckingAccount() {
        accountNumber = 0;
        balance = 0.0;
        transactionFee = 0.0;
    }

    // Parameterized Constructor
    CheckingAccount(int accNum, float bal, float fee) {
        accountNumber = accNum;
        balance = bal;
        transactionFee = fee;
    }

    // Destructor
    ~CheckingAccount() {
        cout << "Checking Account " << accountNumber << " is closed.\n";
    }

    // Withdraw with transaction fee
    {
        float totalDeduction = amount + transactionFee;
        if (balance >= totalDeduction) {
            balance -= totalDeduction;
            cout << "Withdrawn: $" << amount << " (Fee: $" << transactionFee << ")\n";
        } else {
            cout << "Insufficient balance for withdrawal and fee!\n";
        }
    }

    // Display account details
    void display() const {
        cout << "Checking Account Number: " << accountNumber << "\nBalance: $" <<
balance
             << "\nTransaction Fee: $" << transactionFee << "\n";
    }
};

// Main function to test the implementation
int main() {
    // Creating and testing Account
    Account acc1(1001, 500.0);
    acc1.deposit(200);
    acc1.withdraw(100);
    acc1.display();

    cout << "\n";

    // Creating and testing SavingsAccount
    SavingsAccount savAcc(2001, 1000.0, 5.0);
    savAcc.applyInterest();
    savAcc.display();

    cout << "\n";

    // Creating and testing CheckingAccount
    CheckingAccount chkAcc(3001, 1500.0, 2.0);
    chkAcc.withdraw(100);
    chkAcc.display();

    return 0;
}
```

**************** End of Question Paper ****************