**OOP LAB # 07 – BCS-2F – Spring 2025**

**TASK # 01**

A bank provides different types of accounts, such as **Savings Account** and **Current Account**. Each account has a method to calculate the **annual interest**, but the formula differs for different account types.

Additionally, the bank allows overloading the deposit method to handle different types of deposits (cash, check, online transfer). The withdraw method is overridden to implement specific rules for different accounts.

Moreover, the bank maintains **customer balances** in an object, and it supports operator overloading for **addition (+), subtraction (-), multiplication (\*), and division (/)** to handle balance calculations efficiently.

**Task Requirements:**

1. **Create a base class** BankAccount with attributes like accountNumber, balance, and a virtual function calculateInterest().
2. **Implement two derived classes**:
   * SavingsAccount (Interest: 5% per year)
   * CurrentAccount (Interest: No interest)
3. **Overload the deposit method** in both derived classes to handle:
   * Deposit via **cash**
   * Deposit via **check**
   * Deposit via **online transfer**
4. **Override the withdraw method** in both derived classes with different rules:
   * **SavingsAccount:** Cannot withdraw if balance is below **$500**
   * **CurrentAccount:** Allows overdraft up to **$1000**
5. **Implement operator overloading** in BankAccount:
   * + to **add two account balances**
   * - to **subtract an amount from balance**
   * \* to **apply interest on balance**
   * / to **calculate equal installment payments from balance**
6. **Create objects** of both SavingsAccount and CurrentAccount, demonstrate method overloading, overriding, and operator overloading with real transaction examples.

**TASK # 02**

An **E-commerce website** provides multiple ways to calculate the **total price** of a customer's order based on the type of product. There are two main product categories:

1. **Electronics:** Tax rate **15%**
2. **Clothing:** Tax rate **5%**

The website supports different **discount methods** based on the type of user (Regular or Premium). Additionally, the website allows combining multiple carts and applying arithmetic operations to calculate total prices dynamically.

**Task Requirements:**

1. **Create a base class** Product with attributes like productID, price, and a virtual function calculatePrice().
2. **Implement two derived classes**:
   * Electronics (Applies **15% tax**)
   * Clothing (Applies **5% tax**)
3. **Overload the applyDiscount method** to handle different discount types:
   * **Percentage discount** (e.g., 10% off)
   * **Fixed discount** (e.g., $20 off)
   * **Buy One Get One Free**
4. **Override the calculatePrice method** to apply category-specific tax rates.
5. **Implement operator overloading** for the ShoppingCart class:
   * + to **merge two shopping carts**
   * - to **remove an item from the cart**
   * \* to **apply bulk discounts**
   * / to **split cart total among multiple users**
6. **Create objects** for Electronics and Clothing categories, demonstrate method overloading, overriding, and operator overloading by simulating an online shopping experience.