Lab Assignment – 4

Q1. Design a C++ program for a **Rectangle Geometry Application** that demonstrates the use of **friend functions**.

The program must satisfy the following requirements:

Define a class Rectangle with the following **private members and public member functions**:

- float length
- float breadth
- A **getdata() function** to initialize length and breadth.
- A **display()** function to show the dimensions of the rectangle.

Create the following **friend functions**:

1. float **calculateArea**(Rectangle r) – returns the area of a rectangle.
2. float **calculatePerimeter**(Rectangle r) – returns the perimeter of a rectangle.
3. Rectangle **compareArea**(Rectangle r1, Rectangle r2) – compares two rectangles and returns the rectangle with the larger area.

Assume the rectangles represent **plots of land** in a real-estate system. The program should:

1. Accept dimensions of two different plots (rectangles).
2. Display the **area and perimeter** of each plot.
3. Compare the plots and display **which plot is larger** in terms of area.
4. If areas are equal, display that the plots are of the **same size**.

Q2. Design a C++ program that simulates a **simple e-commerce discount system** using the concept of **friend functions**. The system should allow customers to see the **original price** of products, apply a **discount**, and display the **final payable price**.

Define a class Product with **private data members and public member functions**:

- string name – name of the product
- float price – base/original price
- float discountPercentage – discount on the product in %
- A **getdata** to initialize product details.
- A **displayProduct()** function to show product name, price, and discount.

Write a **friend function**: float applyDiscount(Product p). This function should calculate the **final price** after applying the discount and return the final price to the calling program.

In the main() function:

1. Create an **array (or vector) of Product objects**.
2. Accept details of multiple products from the user.
3. Display each product's:
    - Original Price
    - Discount Percentage
    - Final Price after Discount

Q3. Create two classes DM and DB which store the value of distances. DM stores distances in metres and centimetres and DB in feet and inches. Write a program that can read values for the class objects and add one object of DM with another object of DB.

Use a friend function to carry out the addition operation. The object that stores the results may be a DM object or DB object, depending on the units in which the results are required.

The display should be in the format of feet and inches or metres and centimetres depending on the object on display.

Q4. Develop a **Bank Transaction System** in C++ that simulates how deposits and withdrawals are managed in a bank account.

The program should use **two classes**:

- **Account** → stores balance
- **Transaction** → manages deposit, withdrawal, and balance checking

A **friend function** will allow the **Transaction class** to directly access and update the **private balance** inside Account.

- **Account Class**

Define a class Account with the following private members and public member function:

- string accountHolderName
- int accountNumber
- float balance
- A **InitializeData()** function to initialize account details with an opening balance.
- A function **displayAccount()** to show account details.

- **Transaction Class**

Define a class Transaction that will perform banking operations. It should contain **friend functions**:

1. void deposit(*Account &a, float amount*)
    - Adds amount to balance.
    - Prints confirmation message.
2. void withdraw(*Account &a, float amount*)

- Deducts amount if sufficient balance exists.
- If insufficient balance, show error message.

3. void checkBalance(*Account &a*)
   - Displays the current balance of the account.

➢ Scenario:

- The system represents **real banking operations**.
- Balance is **private and confidential**, so only **authorized transactions** (via friend functions) can modify it.
- Deposit simulates **adding funds**.
- Withdraw simulates **ATM/cash withdrawal**.
- Check balance simulates **balance inquiry** at ATM or bank portal.

➢ **Workflow:**

1. Create an Account object with details (account number, holder name, and opening balance).
2. Perform a series of transactions (deposit, withdraw, balance check).
3. Display results after each operation.