**CS217 Object Oriented Programming**

**Week 9**

**Task-1:**

You are tasked with designing a class to manage bank accounts, called **BankAccount**. The class should include private data members for the **account holder's name**, **account number**, and **current balance**. Your goal is to create a friend function that allows for secure fund transfers between two bank accounts.

The **BankAccount** class should have the following private members:

* string **accountHolderName**: A string to store the account holder's name.
* string **accountNumber**: A string to store the account number.
* double **balance**: A double to store the current account balance.

Your task is to:

1. Design the `BankAccount` class with appropriate constructors and access functions.

2. Create a friend function, **friend bool TransferFunds(BankAccount& source, BankAccount& destination, double amount)**, that performs the following actions:

* Check whether the source account has a sufficient balance to transfer the specified `amount`. If not, the transfer should fail, and the function should return `false`.
* Deduct the `amount` from the source account's balance and add it to the destination account's balance.
* Log the transaction with a message containing the account details of both the source and destination accounts and the transferred amount.
* Return true to indicate that the transfer was successful.

- Note: The `TransferFunds` function should have access to the private members of both the source and destination `BankAccount` objects as a friend function.

**Task 2:**

Matrix Multiplication You are given a Matrix class with private members for the number of rows and columns, and a dynamically allocated 2D array to store the matrix elements. Create a friend function that can perform matrix multiplication between two Matrix objects. Make sure to handle cases where the dimensions are incompatible for multiplication.

**Task-3:**

In this task, you are tasked with developing a class to manage user information, including password encryption, named `User`. Your objective is to create a friend function to encrypt user passwords securely.

The `User` class should include private data members for the user's name and a private member for their password. Your goal is to design a friend class that can access and modify the password without exposing it directly to outside code.

The `User` class should have the following private members:

* string **username**: A string to store the user's name.
* string **password**: A string to store the user's password.

Your task is to:

1. Design the `User` class with appropriate constructors and access functions to set and retrieve the user's name.

2. Create a friend class, **PasswordEncryptor**, it will have an attribute **encryptedPassword** with the following functionalities:

* It should have access to the private password member of the User object.
* Encrypt the password using a simple encryption algorithm, i.e add 3 to the ascii of the the character, for example if password is Abc123 after encryption it will be Def456.
* Save the encrypted password in the encryptedPassword variable.

**Task-4:**

Design a class **Complex** for handling Complex numbers and include the following **private** data members:

* real: a double
* imaginary: a double

The class has the following member functions.

1. A constructor initializing the number with **default parameters**.
2. Overloaded Constructors.

* Complex(double r, double i)

Note\*:Use member function initialization for all data members.

* Complex(Complex & copy) // copy constructor

1. Getters and Setters of the class data members as given below

* void setReal(double r)
* double getReal()
* void setImaginary(double i)
* double getImaginary()

1. Overload the following member function in the class

**Complex operator+(double r)**: It adds r of type double to real part of complex number while imaginary part remains same. And returns newly generated complex number.

**Complex operator+ (const Complex &c1)**; It adds both complex numbers and returns newly generated complex number.

**Complex operator- (double r)**: It subtracts r of type double from real part of complex number while imaginary part remains same. And returns newly generated complex number.

**Complex operator- (const Complex &c1)**: It subtracts both complex numbers and returns newly generated complex number.

**Complex operator\*(double n):** It’s a scalar multiplication. Real and imaginary parts are multiplied by n. and returns newly generated complex number.

**Complex operator\*(const Complex &c1):** It multiplies both complex numbers and returns newly generated complex number. (a+bi)(c+di) = (ac−bd) + (ad+bc)i