CS3201: OBJECT ORIENTED PROGRAMMING LABORATORY

***Topic*: Constructors, Destructors, Friend Functions,**
Static Functions and Function Overloading

Date: 19/02/2025

SPOT QUESTIONS

Answer any three out of four questions:

1. Create a class **Vector3D** that represents a 3D vector with private members *x, y, and z*. Implement the following functionalities: A constructor that initializes the **Vector3D** object using three arguments. A friend function *DotProduct* to compute the dot product of two **Vector3D** objects. Extend the functionality to compute the *cumulative dot product of three **Vector3D** objects* (without using arrays or vectors). Create a friend function to compute the cross product of two **Vector3D** objects and return the result as a new **Vector3D**. Write a main function to demonstrate all the above.

2. Design a **BankAccount** class with the following specifications: Private data members for *accountNumber*, *balance*, and a static variable *totalAccounts* to keep track of the total number of accounts. A constructor that increments *totalAccounts* every time an account is created. A static member function *GetTotalAccounts()* that returns the number of accounts created so far. A static variable *interestRate* and a static member function *SetInterestRate(**double** rate)* to modify the interest rate for all accounts. A member function *ApplyInterest()* that applies the current interest rate to the account's balance. Implement a function *TransferAmount(**double** amount, **BankAccount**& toAccount)* that transfers money between two accounts without accessing private data directly. Write a main function to create multiple accounts, set and modify the interest rate, apply interest, transfer money between accounts, and demonstrate proper usage of static and non-static members.

3. Create a **Matrix** class that represents a 2D matrix. Implement the following functionalities: A *default constructor* that initializes a 2x2 matrix with all elements set to 0. A *parameterized constructor* that accepts row and column dimensions and initializes the matrix with a specific value. A function *AddMatrices* that takes two matrices of the same size as arguments and returns a new matrix containing the sum of corresponding elements. A function *DisplayMatrix* to print the matrix elements row by row. A copy constructor that performs a deep copy of a given matrix. A destructor to release allocated memory. Write a main function to create matrices using various constructors, demonstrate matrix addition, and verify that the deep copy works correctly by modifying the original matrix and checking that the copied matrix remains unchanged.

4. Create a **Calculator** class that demonstrates advanced function overloading with the following requirements: Overload the ***Add function*** to handle two integers and return their sum, three floating-point values and return their sum, and two fixed-size arrays of integers (e.g., int arr1[5] and int arr2[5]). This function should compute the element-wise sum and return a new array. Overload the ***Multiply function*** to handle two integers and return their product, two floating-point numbers and return their product, and a fixed-size 2D array and a scalar to multiply each element of the array by the scalar. Write a main function to demonstrate all overloaded functions.