

Project - 1 **MA517M-Basic Programming Laboratory** Last Date: 14 November 2026
Name _____ Roll No.: **MA26M014**

Polynomial Class with Integer or Finite Field Coefficients using C++ Classes

Objective: To design a C++ program that implements polynomials with integer or finite field coefficients, supports arithmetic operations, evaluation, derivation, and demonstrates operator overloading.

A polynomial of degree n is of the form

$$P(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

where $a_i \in \mathbb{Z}$ or $a_i \in GF(p)$ (finite field of order p). Operations include addition, subtraction, multiplication, evaluation at a point, derivation, and comparison of polynomials.

Problem Description

Design a class `Polynomial` to represent polynomials with integer or finite field coefficients. The class should support arithmetic operations, evaluation at a point, derivation, and equality comparison using operator overloading.

Class Specification

- **Class Name:** `Polynomial`
- **Private Data Members:**
 - `int degree;`
 - `vector<int> coeffs;`
 - `int modulus;`
- **Public Member Functions:**
 - `Polynomial(vector<int> c, int p=0);` Constructor initializes polynomial with coefficients and modulus
 - `Polynomial derivative() const;` Returns derivative polynomial
 - `int evaluate(int x) const;` Evaluates polynomial at given x (mod p if finite field)
 - `void display() const;` Displays polynomial in standard form

Operator Overloading

- operator `+`, `-` Polynomial addition and subtraction
- operator `*` Polynomial multiplication
- operator `==` Compares two polynomials
- operator `<<` Displays polynomial in readable form

Tasks

1. Create polynomials $P_1(x) = 1 + 2x + 3x^2$ and $P_2(x) = 3 + x + 4x^3$ over integers or $GF(5)$
2. Compute $P_1 + P_2$, $P_1 - P_2$, $P_1 * P_2$
3. Evaluate P_1 and P_2 at $x = 2$
4. Compute derivatives P_1' and P_2'
5. Compare polynomials using the overloaded equality operator

Expected Output Example

```
P1(x) = 1 + 2x + 3x^2
P2(x) = 3 + 1x + 0x^2 + 4x^3 (mod 5)

P1 + P2 = 4 + 3x + 3x^2 + 4x^3 (mod 5)
P1 - P2 = 3 + 1x + 3x^2 + 1x^3 (mod 5)
P1 * P2 = 3 + 0x + 3x^2 + 3x^3 + 2x^4 + 2x^5 (mod 5)

P1(2) = 3 (mod 5)
P2(2) = 0 (mod 5)

Derivative P1' = 2 + 6x
Derivative P2' = 1 + 12x^2
P1 == P2 : False
```

Project - 2: Connect Four Using C++ Classes

Problem Statement

Design and implement **Connect Four** using **C++ classes**. Connect Four is a two-player game where players take turns dropping colored discs into a vertical 6×7 grid. The goal is to connect four discs in a row, column, or diagonal before the opponent does. The project should utilize object-oriented programming concepts such as classes, objects, encapsulation, and methods for handling game logic and player interaction.

Project Requirements

1. Create a **Disc** class to represent an individual disc.
 - (a) Data member: color or player identifier.
 - (b) Methods: `getColor()`, `setColor()`.
2. Create a **Board** class to represent the game grid.
 - (a) Initialize a 6×7 grid with empty slots.
 - (b) Display the current state of the grid in a readable format.
 - (c) Include a method to drop a disc into a specified column.
 - (d) Include a method to check for a winning condition (four connected discs horizontally, vertically, or diagonally).
 - (e) Include a method to check if the board is full (draw condition).
3. Create a **Game** class to manage gameplay.
 - (a) Allow two players to take turns dropping discs.
 - (b) Display options for the user: **Play** or **Solution**.
 - (c) After each move, update the board and display it.
 - (d) Allow the user to quit and confirm before exiting.
 - (e) Keep track of player turns and optionally score.
4. Ensure proper encapsulation of disc and board operations within the respective classes.

Suggested Class Structure

1. **Disc Class:**
 - Data member: `char color` or `int playerID`
 - Methods: `getColor()`, `setColor()`
2. **Board Class:**
 - Data member: 2D array of **Disc** objects
 - Methods: `initializeBoard()`, `displayBoard()`, `dropDisc(int column, Disc disc)`, `checkWin()`, `isFull()`
3. **Game Class:**
 - Data member: **Board** object, `currentPlayer`, `user choice`
 - Methods: `playGame()`, `showSolution()`, `switchPlayer()`, `confirmQuit()`

Reference

For more details about Connect Four, visit: https://en.wikipedia.org/wiki/Connect_Four