
Project - 1 MA517M-Basic Programming Laboratory	Last Date: 09 November 2025
Name	Roll No.: MA25M007

Binomial Random Variables using C++ Classes and Operator Overloading

Objective: To design a C++ program that implements binomial random variables using classes, computes PMF, CDF, mean, variance, and supports operator overloading for comparison and addition of independent variables.

A binomial random variable $X \sim \text{Bin}(n, p)$ counts the number of successes in n independent Bernoulli trials with success probability p .

The probability mass function (PMF) is

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}, \quad k = 0, 1, \dots, n$$

The cumulative distribution function (CDF) is

$$F(k) = P(X \leq k) = \sum_{i=0}^k \binom{n}{i} p^i (1 - p)^{n-i}$$

Mean and variance:

$$\mathbb{E}[X] = np, \quad \text{Var}(X) = np(1 - p)$$

Problem Description

Design a class `BinomialRV` to represent a binomial random variable. The class should allow computation of PMF, CDF, mean, variance, comparison with other binomial random variables, and addition of independent binomial variables (resulting in a new binomial variable if they share the same p).

Class Specification

- **Class Name:** `BinomialRV`
- **Private Data Members:**
 - `int n;`
 - `double p;`
- **Public Member Functions:**
 - `BinomialRV(int trials, double prob);`
 - `double pmf(int k) const;`

- `double cdf(int k) const;`
- `double mean() const;`
- `double variance() const;`
- `void display() const;`

Operator Overloading

- `operator +()` Adds two independent binomial random variables with the same p
- `operator ==(())`, `operator !=()`, `operator <()`, `operator >()` Compares the mean values of two binomial variables
- `operator <<()` Displays binomial parameters and statistics

Tasks

1. Create binomial random variables $X \sim \text{Bin}(5, 0.6)$ and $Y \sim \text{Bin}(3, 0.6)$
2. Compute PMF and CDF for selected values
3. Compute mean and variance
4. Add $X + Y$ and display resulting binomial variable
5. Compare X and Y using overloaded comparison operators
6. Display all results using the overloaded `<<` operator

Expected Output Example

```
X ~ Bin(5, 0.6)
PMF P(X=3) = 0.3456
CDF P(X<=3) = 0.6826
Mean = 3.0, Variance = 1.2

Y ~ Bin(3, 0.6)
PMF P(Y=2) = 0.432
CDF P(Y<=2) = 0.784
Mean = 1.8, Variance = 0.72

X + Y = Bin(8, 0.6)
Mean = 4.8, Variance = 1.92

X > Y : True
X == Y: 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