
Project - 1 MA517M-Basic Programming Laboratory	Last Date: 09 November 2025
Name	Roll No.: MA25M013

Negative Binomial Random Variables using C++ Classes and Operator Overloading

Objective: To design a C++ program that implements negative binomial random variables using classes, computes PMF, CDF, mean, variance, and supports arithmetic and comparison operators.

A negative binomial random variable $X \sim \text{NegBin}(r, p)$ counts the number of trials required to achieve r successes, with success probability p .

The probability mass function (PMF) is

$$P(X = k) = \binom{k-1}{r-1} p^r (1-p)^{k-r}, \quad k = r, r+1, r+2, \dots$$

The cumulative distribution function (CDF) is

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

Mean and variance:

$$\mathbb{E}[X] = \frac{r}{p}, \quad \text{Var}(X) = \frac{r(1-p)}{p^2}$$

Problem Description

Design a class `NegativeBinomialRV` to represent a negative binomial random variable. The class should allow computation of PMF, CDF, mean, variance, arithmetic with independent variables (addition), and comparison using operator overloading.

Class Specification

- **Class Name:** `NegativeBinomialRV`
- **Private Data Members:**
 - `int r;`
 - `double p;`
- **Public Member Functions:**
 - `NegativeBinomialRV(int successes, 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 negative binomial random variables with the same p (resulting in a new `NegBin` variable with $r_1 + r_2$ successes)
- `operator ==(())`, `operator !=()`, `operator <()`, `operator >()` Compares mean or variance of two negative binomial variables
- `operator <<()` Displays the variable's parameters, mean, and variance

Tasks

1. Create negative binomial random variables $X \sim \text{NegBin}(r = 3, p = 0.4)$ and $Y \sim \text{NegBin}(r = 2, p = 0.4)$
2. Compute PMF and CDF for selected values of k
3. Compute mean and variance
4. Add $X + Y$ assuming independence and same p , display resulting variable
5. Compare X and Y using overloaded comparison operators
6. Display all results using the overloaded `<<` operator

Expected Output Example

```
X ~ NegBin(r=3, p=0.4)
PMF P(X=5) = 0.2304
CDF P(X<=5) = 0.682
Mean = 7.5, Variance = 11.25
```

```
Y ~ NegBin(r=2, p=0.4)
PMF P(Y=4) = 0.1536
CDF P(Y<=4) = 0.6
Mean = 5.0, Variance = 7.5
```

```
X + Y = NegBin(r=5, p=0.4)
Mean = 12.5, Variance = 18.75
```

```
X > Y : True
X == Y: False
```

Project - 2 : Word Shuffle Game Using C++ Classes

Problem Statement

Design and implement a **Word Shuffle Game** using **C++ classes**. The program should allow the user to unscramble letters to form meaningful words. The project should utilize object-oriented programming concepts such as classes, objects, encapsulation, and methods to handle word selection, shuffling, and user interaction.

Project Requirements

1. Create a **Word** class to represent a word in the game.
 - (a) Include a method to randomly select a word from a predefined list.
 - (b) Include a method to shuffle the letters of the word.
 - (c) Include a method to display the shuffled word to the user.
2. Create a **Game** class to manage gameplay.
 - (a) Display options for the user: **Play** or **Solution**.
 - (b) If the user chooses **Play**, allow them to enter guesses.
 - (c) Validate user input and provide feedback if the guess is correct or incorrect.
 - (d) Keep track of the number of attempts.
 - (e) Allow the user to quit by entering a special command (e.g., **Q/q**) and confirm before exiting.
3. If the user chooses **Solution**, display the original word and the correct sequence of letters.
4. Ensure proper encapsulation of word logic and gameplay operations within the respective classes.

Suggested Class Structure

1. **Word Class**:
 - Data member: `string originalWord`, `string shuffledWord`
 - Methods: `selectWord()`, `shuffleWord()`, `displayWord()`
2. **Game Class**:
 - Data member: `Word` object, user choice, attempt counter
 - Methods: `playGame()`, `showSolution()`, `processGuess(string guess)`, `confirmQuit()`