INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

Project - 1 **MA517M-Basic Programming Laboratory** Last Date: 09 November 2025
Name **Roll No.: MA25M001**

Gaussian Elimination Game

1. Objective

The objective of this project is to design and implement a console-based **Matrix Puzzle Game** using basic `C++` programming constructs such as `structures`, `classes`, `loops`, and `conditional statements`. The player is presented with a random square matrix and must perform a sequence of valid **elementary row operations** to convert it into the **identity matrix**.

This project allows students to:

- Apply the concepts of **matrix algebra** computationally.
- Understand the process of **Gaussian elimination** through an interactive environment.
- Strengthen problem-solving and logical reasoning skills.

2. Mathematical Background

A matrix is a rectangular array of numbers arranged in rows and columns. A square matrix $A \in \mathbb{R}^{n\times n}$ is said to be **invertible** if there exists another matrix B such that

$$AB = BA = I_n,$$

where I_n is the $n \times n$ identity matrix.

To convert a given matrix A to I_n, one can perform the following **elementary row operations**:

1. Swapping two rows $R_i \leftrightarrow R_j$,
2. Multiplying a row by a non-zero scalar $R_i \to kR_i$,
3. Adding a scalar multiple of one row to another $R_i \to R_i + kR_j$.

These operations correspond to left-multiplication by **elementary matrices**. The determinant and rank properties of matrices can also be explored as part of the gameplay.

3. Game Description

The program begins by generating a random 3×3 or 4×4 matrix with integer entries. The user is shown the matrix and a menu of allowed operations.

The goal is to reach the identity matrix within the fewest possible steps. Each operation updates the matrix and displays the intermediate result. The game ends when the matrix becomes an identity matrix or after a fixed number of moves.

4. Algorithm

1. Initialize a random $n \times n$ matrix A.
2. Display A to the user.
3. Repeat until $A = I_n$ or the number of moves exceeds the limit:
   (a) Display menu:
      1. Swap two rows
      2. Multiply a row by a scalar
      3. Add a multiple of one row to another
      4. Check progress
   (b) Take user input and perform the chosen operation.
   (c) Update and display the matrix.
4. If $A = I_n$, display "Congratulations!"; otherwise, show "Game Over".

5. Program Structure (C++ Skeleton)

```
struct Matrix {
    double a[4][4];
    int n;
};

class MatrixGame {
private:
    Matrix M;
    int moves;
public:
    void initialize();
    void display();
    void swapRows(int r1, int r2);
    void scaleRow(int r, double k);
    void addRows(int r1, int r2, double k);
    bool isIdentity();
    void play();
};
```

6. Sample Interaction

Output:

```
Welcome to the Matrix Puzzle Game!
Initial Matrix:
[ 2  1  3 ]
```

```
[ 0  1  2 ]
[ 1  0  1 ]

Choose an operation:
1. Swap Rows
2. Scale Row
3. Add Rows
> 2
Enter row and scalar: 1 0.5

Updated Matrix:
[ 1  0.5  1.5 ]
[ 0  1    2   ]
[ 1  0    1   ]
Moves used: 1
```

Goal: Continue performing operations until the matrix becomes the identity matrix.

7. Possible Extensions

- Implement a scoring system based on the number of moves.
- Allow $n \times n$ matrices of arbitrary size.
- Add an option to compute and display the determinant after each move.
- Introduce a leaderboard for competitive play.

Project - 2: Number Slide Game Using C++ Classes

Problem Statement

Design and implement a **Number Slide Game** using **C++ classes**. The program should simulate a 4×4 sliding puzzle where the goal is to arrange numbers from 1 to 15 in order, leaving one empty space. The project should utilize object-oriented programming concepts such as classes, objects, encapsulation, and methods for handling game logic.

Project Requirements

1. Create a `Matrix` class to represent the 4×4 game board.
   (a) Include a method to generate a random matrix of size 4×4 using `srand()` in the range [1, 16].
   (b) Include a method to check that all entries of the matrix are distinct.
   (c) Remove the entry with the value 16 to represent the empty space.
2. Create a `Game` class to manage gameplay.

(a) Display options for the user: `Play` or `Solution`.

(b) If the user chooses `Play`, show the navigation commands:

- `A/a` for Left
- `S/s` for Down
- `D/d` for Right
- `W/w` for Up
- `Q/q` for Quit

(c) When the user presses `Q/q`, confirm again before quitting.

(d) After each move, display the updated matrix in a nicely formatted way.

3. If the user chooses `Solution`, display the sequence of movements leading to the solved puzzle, and show the final output.

4. Ensure proper encapsulation of game logic and board operations within the respective classes.

Suggested Class Structure

1. `Matrix` Class:

   - Data member: 4×4 integer array representing the board
   - Methods: generateMatrix(), checkDistinct(), removeEmpty(), displayMatrix()

2. `Game` Class:

   - Data member: `Matrix` object, user choice
   - Methods: playGame(), showSolution(), processMove(char move), confirmQuit()

Reference

For more details about the game, visit: http://www.artbylogic.com/puzzles/numSlider/numberShuffle.htm