INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

Project - 1 **MA517M-Basic Programming Laboratory** Last Date: 09 November 2025

Name **Roll No.: MA25M002**

Vector Transformation Challenge

1. Objective

The aim of this project is to design and implement a console-based **Vector Transformation Challenge** game using basic `C++` constructs such as `structures`, `classes`, and `functions`.

In this game, the player applies a sequence of linear transformations represented by matrices to a given vector, with the goal of reaching a specified **target vector**.

This project helps students:

- Understand and apply concepts from **linear transformations** and **matrix-vector multiplication**.
- Strengthen algorithmic thinking through sequential transformations.
- Reinforce computational skills in C++ while manipulating vector and matrix data.

2. Mathematical Background

A linear transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ can be represented by a matrix $A \in \mathbb{R}^{n\times n}$ such that

$$T(\mathbf{x}) = A\mathbf{x}.$$

If several transformations $A_1, A_2, \ldots, A_k$ are applied successively to a vector $\mathbf{x}_0$, the result is:

$$\mathbf{x}_k = A_k A_{k-1} \cdots A_1 \mathbf{x}_0.$$

This concept forms the foundation of the game. Players can think of each matrix as a "move" that modifies the current vector.

3. Game Description

The program starts by generating:

- an initial vector $\mathbf{v}_0 \in \mathbb{R}^n$,
- a target vector $\mathbf{v}_{\text{target}}$,
- and a set of available transformation matrices (e.g., rotation, reflection, scaling, shear).

At each turn, the player chooses one transformation matrix to apply to the current vector. The objective is to reach (or get as close as possible to) the target vector within a limited number of moves.

4. Algorithm

1. Initialize:
   - dimension n (usually $n = 2$ or $n = 3$),
   - initial vector $\mathbf{v}_0$,
   - target vector $\mathbf{v}_{\text{target}}$,
   - a predefined list of transformation matrices A_i.
2. Display the initial and target vectors.
3. Repeat for a fixed number of moves:
   (a) Display menu of available transformations:
      1. Rotation by 90°
      2. Reflection across x-axis
      3. Scaling by 2
      4. Shear transformation
      5. Reset to initial vector
   (b) Take user input (choice of transformation).
   (c) Multiply the current vector by the chosen matrix.
   (d) Display the new vector and the distance to the target:

$$d = \|\mathbf{v}_{\text{current}} - \mathbf{v}_{\text{target}}\|.$$

   (e) If $d < \varepsilon$, display "You reached the target!" and end the game.
4. If the player exhausts all moves, display the final distance and a message.

5. Program Structure (C++ Skeleton)

```
struct Vector {
    double val[3];
    int n;
};

struct Matrix {
    double val[3][3];
    int n;
};

class TransformationGame {
private:
    Vector current, target;
    Matrix transformations[5];
```

```
    int moves;
public:
    void initialize();
    void displayVector(Vector v);
    Vector multiply(Matrix A, Vector v);
    void play();
    double distance(Vector a, Vector b);
};
```

6. Sample Interaction

Output:

```
Welcome to the Vector Transformation Challenge!
Initial vector: [1, 0]
Target vector:  [0, 1]

Choose a transformation:
1. Rotate 90 degrees
2. Reflect across x-axis
3. Scale by 2
4. Shear
> 1

New vector: [0, 1]
Distance to target: 0.00
Congratulations! You reached the target in 1 move!
```

7. Possible Extensions

- Introduce random target vectors for repeated play.
- Add 3D transformations (rotation about axes).
- Allow players to define custom transformation matrices.
- Introduce scoring based on minimal distance or number of moves.

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()