1. The Matrix Summer

The Goal: Write a program that initializes a 3x4 integer array with values of your choice. Use nested loops to calculate the sum of all elements and the average value.

- **Key Concept:** Iterating through rows (i) and columns (j) using array.GetLength(0) and array.GetLength(1).
- **Bonus:** Print the sum of each individual row separately.

---

2. Multi-Dimensional Identity

The Goal: Create a square 2D array (e.g., 5x5). Use nested loops to fill the array so that it represents an **Identity Matrix**—where all diagonal elements (from top-left to bottom-right) are **1** and all other elements are **0**.

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad I_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Key Concept:** Using an if statement inside the loop to check if the current row index equals the current column index (i == j).

---

3. The Multiplication Table Generator

The Goal: Ask the user for a number N. Create a 2D array of size N * N and fill it with a multiplication table (e.g., the cell at [2,3] should contain 3 * 4 = 12 if you are using 1-based logic, or 2 * 3 = 6 for 0-based).

- **Key Concept:** Using loop indices as the data source for the array values.
- **Formatting Tip:** Use Console.Write(table[i,j] + "\t") to keep the columns aligned when printing.

---

4. Transpose the Grid

The Goal: Start with a 2x3 rectangular array (2 rows, 3 columns) filled with random numbers. Create a second array that is the **transpose** of the first (it should be 3x2).

- **Key Concept:** Mapping original[row, col] to transposed[col, row]. This is a classic test of how well you can visualize spatial data in code.

---

5. The "Game of Life" Style Neighbor Check

The Goal: Initialize a 5x5 grid of characters where most cells are '.' (empty) and a few are 'X' (entities). Pick a specific coordinate (e.g., row 2, col 2) and use nested loops to count how many 'X' entities are in the 8 cells immediately surrounding it.

- **Key Concept:** Implementing "offset" loops (running from -1 to 1) and adding safety checks to ensure you don't look for an index outside the array bounds (e.g., row < 0 or row >= length).