Matrix

Write a **matrix** class that represents a 2D array with the following description.

You ***must***:

1. Implement a **constructor** that takes two numbers; the number of columns and rows of the matrix; and initiates matrix elements to zero **.**

2. Implement **constructor** that takes the number of columns and rows of the matrix and a float number as an initial value to all the matrix elements **.**

3. Implement **constructor** that takes the number of columns and rows of the matrix and an enumerator called **MatrixType** that contains one of the following values

   - **Identity**: initiate the matrix to be identity (each diagonal element is one and other elements are zero).
   - **Random**: the matrix elements are initiated with random values.

4. Implement a print function that prints the whole matrix in suitable format.

5. Implement two functions **get** and **set** to access one location inside the matrix. (Note: you may write one function to access the location)

6. Implement the **+ and - operators** that adds or subtracts two matrices.

7. Implement **= = and != operators**.

8. Implement the *** operator** that multiplies two matrices.(use matrix multiplication rules)

9. Implement a function called **transpose** that return the transpose of the matrix. (Formed by turning rows into columns or columns into rows).

10. Implement two functions called **Row** and **Column** that returns the count of matrix Rows and columns.

11. Implement the following **Boolean functions**

    - **IsIdentity** that returns true if the matrix is identity
    - **IsIdempotent:** idempot matrix is a square matrix A as $A^2 = AA = A$
    - **IsSquare** that returns true if the matrix is square
    - **IsSymmetric:** such that the elements symmetric about the ***main diagonal*** (from the upper left to the lower right) are equal, that is, $a_{ij}=a_{ji}$.
    - **IsUpperTriangle** such that upper triangle matrix has zero elements below the diagonal
    - **IsLowerTriangle** such that upper triangle matrix has zero elements above the diagonal

    (Note: All these matrices should be square)

12. Implement Fill function that fills the entire matrix with a value.

13. Implement the *, / operators that multiplies and divides the matrix by a float value.

14. Implement **MaxElement** and **MinElement** functions to return the max and min element in the Matrix.

You ***may (bonus)***:

1. Write a function called **ToString** that returns a string containing the matrix in a suitable form. (Hint: You may use sprint or use ostrstream as in P.621 in Lafore)
2. Implement **determinate** function that return a float number represent the determinate of the matrix, **Note** the matrix should be square.
3. 3. Implement the **/ operator** that divided two matrix (hint: you may use Gauss Elimination method)

Hint: write a main function to test this class.