

SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

This question involves performing arithmetic operations on two-dimensional (2D) arrays of integers. You will write two static methods, both of which are in a class named `MatrixOp` (not shown).

(a) Write the method `diagonalOp`, which returns the sum of the products of the corresponding entries on the main diagonals of two given square 2D arrays that have the same dimensions. The main diagonal goes from the top-left corner to the bottom-right corner in a square 2D array.

For example, assume that `mat1` and `mat2` are properly defined 2D arrays containing the values shown below. The main diagonals have been shaded in gray.

mat1			mat2		
2	4	2	-1	8	9
8	5	1	6	3	5
4	2	4	5	1	2

After the call `int sum = MatrixOp.diagonalOp(mat1, mat2)`, `sum` would contain 21, as illustrated below.

$$\text{sum} = (2 * -1) + (5 * 3) + (4 * 2) = 21$$

Complete method `diagonalOp`.

```
/** Returns an integer, as described in part (a).
 * Precondition: matA and matB are 2D arrays that are both square,
 * have at least one row, and have the same dimensions.
 */
public static int diagonalOp(int[][] matA, int[][] matB)
```

(b) Write the method `expandMatrix`, which returns an expanded version of a given 2D array. To expand a 2D array, a new 2D array must be created and filled with values such that each element of the original 2D array occurs a total of four times in the new 2D array, arranged as shown in the example below.

For example, assume that `mat3` is a properly defined 2D array containing the values shown below.

		mat3		
		0	1	2
0		-1	2	3
1		5	4	6

After the call `int[][] mat4 = MatrixOp.expandMatrix(mat3)`, the array `mat4` would contain the values shown below.

		mat4					
		0	1	2	3	4	5
0		-1	-1	2	2	3	3
1		-1	-1	2	2	3	3
2		5	5	4	4	6	6
3		5	5	4	4	6	6

Complete method `expandMatrix`.

```
/** Returns a 2D array, as described in part (b).
 * Precondition: matA is a 2D array with at least one row and
 * at least one column.
 */
public static int[][] expandMatrix(int[][] matA)
```