Assignment#5: Arrays#2 [2D & 3D] (25 Sep 2024)

Problem#1: Simple Statistics (2D)

Write a Java program that performs basic statistical analysis on a two-dimensional array of integers. Your program will calculate and display the following statistics based on the values in the array:

- **Sum:** The total sum of all the values in the array.
- **Minimum Value:** The smallest number in the array.
- **Maximum Value:** The largest number in the array.
- **Average (Mean):** The average of all the numbers in the array.
- **Median:** The middle value when the numbers in the array are sorted in ascending order. If the number of elements is even, the median is the average of the two middle numbers.

The program will first accept two integers, m and n, representing the number of rows and columns of the array, followed by the elements of the m x n array.

Test Cases:

Input	Output
2 2 3 7 2 4	Sum: 16 Minimum value: 2 Maximum value: 7 Average value: 4.0 Median value: 3.5
3 3 5 5 5 3 4 5 6 3 3	Sum: 39 Minimum value: 3 Maximum value: 6 Average value: 4.333333333333333 Median value: 5.0

4 4 -1 -5 2 4 -7 3 -2 1 5 0 2 -6 3 -3 7 -4	Sum: -1 Minimum value: -7 Maximum value: 7 Average value: -0.0625 Median value: 0.5
1 5 8 8 8 8 8	Sum: 40 Minimum value: 8 Maximum value: 8 Average value: 8.0 Median value: 8.0
5 1 -10 5 3 0 -4	Sum: -6 Minimum value: -10 Maximum value: 5 Average value: -1.2 Median value: 0.0
5 4 12 5 -9 2 3 -6 5 5 0 0 -1 7 8 6 -3 2 1 -4 3 5	Sum: 41 Minimum value: -9 Maximum value: 12 Average value: 2.05 Median value: 2.5
1 1 10	Sum: 10 Minimum value: 10 Maximum value: 10 Average value: 10.0 Median value: 10.0
10 10 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50	Sum: 5050 Minimum value: 1 Maximum value: 100 Average value: 50.5 Median value: 50.5

51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100	

Problem#2: Add Two Matrices (2D)

Write a program that reads two 2 x 2 matrices and displays their sum. To be added, the two matrices must have the same dimensions and the same or compatible types of elements.

For example, for two 2 x 2 matrices **a** and **b**, **c** is

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$$

If matrix 1 is 1 2 3 4 5 6 7 8 9 and matrix 2 is 0 2 4 1 4.5 2.2 1.1 4.3 5.2, then the matrices are added as follows:

$$\begin{bmatrix} 1.0 & 2.0 \\ 4.0 & 5.0 \end{bmatrix} + \begin{bmatrix} 0.0 & 2.0 \\ 1.0 & 4.5 \end{bmatrix} = \begin{bmatrix} 1.0 & 4.0 \\ 5.0 & 9.5 \end{bmatrix}$$

Hint: Use double as datatype.

Test Cases:

Input	Output
1 2 4 5 0 2 1 4.5	1.0 4.0 5.0 9.5
1 -2 -3 4 -1 2 3 -4	0.0 0.0 0.0 0.0
0 0 0 0 0 0 0 0	0.0 0.0 0.0 0.0
1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8	6.6 8.8 11.0 13.200000000000001
1 2.5 3.5 4 5.5 6 7 8.5	6.5 8.5 10.5 12.5

Problem#3: Multiply Two Matrices (2D)

Write a program that reads two 3 x 3 matrices and display their product. To multiply matrix a by matrix b, the number of columns in a must be the same as the number of rows in b, and the two matrices must have elements of the same or compatible types. Let c be the result of the multiplication.

For example, for two 3 x 3 matrices **a** and **b**, **c** is

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

where $c_{ij} = a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + a_{13} \times b_{3j}$.

If matrix 1 is 1 2 3 4 5 6 7 8 9 and matrix 2 is 0 2 4 1 4.5 2.2 1.1 4.3 5.2, then the multiplication of the matrices is:

$$\begin{pmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 7.0 & 8.0 & 9.0 \end{pmatrix} * \begin{pmatrix} 0.0 & 2.0 & 4.0 \\ 1.0 & 4.5 & 2.2 \\ 1.1 & 4.3 & 5.2 \end{pmatrix} = \begin{pmatrix} 5.3 & 23.9 & 24 \\ 11.6 & 56.3 & 58.2 \\ 17.9 & 88.7 & 92.4 \end{pmatrix}$$

Each element of the product matrix ab can be calculated as follows:

- ab_{11} = (1.0 x 0.0) + (2.0 x 1.0) + (3.0 x 1.1) = 5.3
- ab_{12} = (1.0 x 2.0) + (2.0 x 4.5) + (3.0 x 4.3) = 11.6
- ab_{13} = (1.0 x 4.0) + (2.0 x 2.2) + (3.0 x 5.2) = 17.9

  ...
- ab_{33} = (7.0 x 4.0) + (8.0 x 2.2) + (9.0 x 5.2) = 92.4

Hint: Use double as datatype.

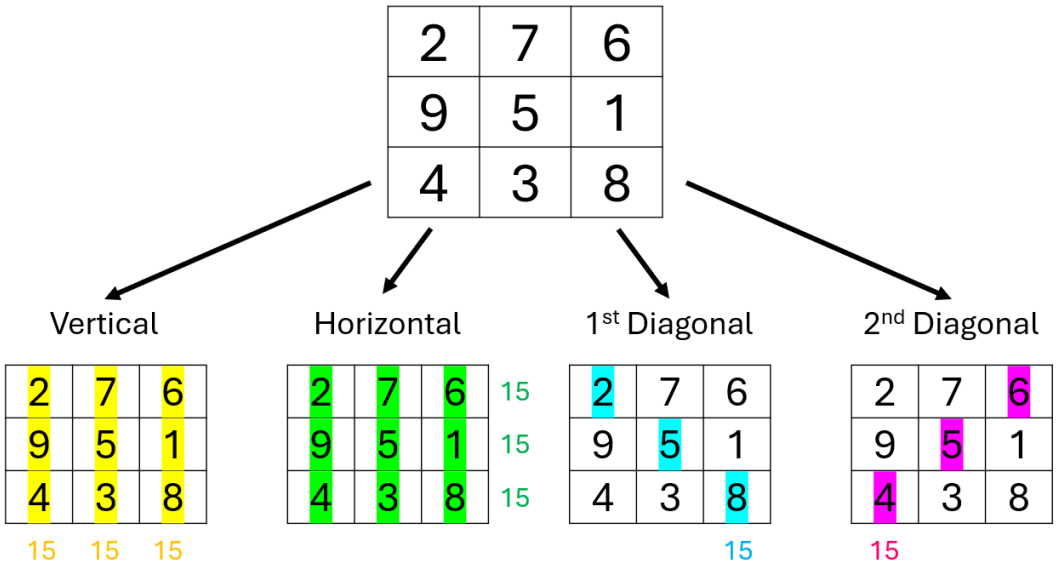
Test Cases:

Input	Output
1 2 3 4 5 6 7 8 9 0 2 4 1 4.5 2.2 1.1 4.3 5.2	5.300000000000001 23.9 24.0 11.600000000000001 56.3 58.2 17.9 88.69999999999999 92.4
1 -2 3 -4 5 -6 7 -8 9 9 -8 7 -6 5 -4 3 -2 1	30.0 -24.0 18.0 -84.0 69.0 -54.0 138.0 -114.0 90.0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8 9.9 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1	3.3 2.64 1.9800000000000002 9.24 7.590000000000001 5.94 15.180000000000001 12.540000000000001 9.9
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1	0.30000000000000004 0.24000000000000002 0.18000000000000002 0.8400000000000001 0.6900000000000001 0.54 1.38 1.14 0.9

Problem#4: Magic Squares in a Grid (2D)

A magic square is a grid of numbers where the sums of numbers in each row, each column, and both diagonals are all the same. For a given n x n grid, your task is to determine if the grid forms a magic square. The grid will consist of integers, and the size of the grid will be provided as input.

For Example: n = 3 (It means 3 x 3 magic square.)



Test Cases:

Input	Output
3 2 7 6 9 5 1 4 3 8	true
2 1 2 3 4	false
4 16 2 3 13 5 11 10 8 9 7 6 12 4 14 15 1	true

4 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	false
3 -3 -1 7 6 2 -2 -5 8 -1	false
3 3 5 7 8 1 6 2 9 4	false

Problem#5: Set Matrix Zeroes (2D)

Given a matrix of size m x n, write a program that modifies the matrix such that if any element in the matrix is 0, the entire row and column containing that element are set to 0. The program should perform this modification in-place, meaning the original matrix should be updated without using any additional matrix storage.

The input to the program is the matrix itself, which may contain both positive and negative integers, and the output should be the modified matrix where all rows and columns containing a 0 have been set to 0 accordingly.

For example: 3 x 3 Matrix

Input:

1	1	1
1	0	1
1	1	1

Output:

1	0	1
0	0	0
1	0	1

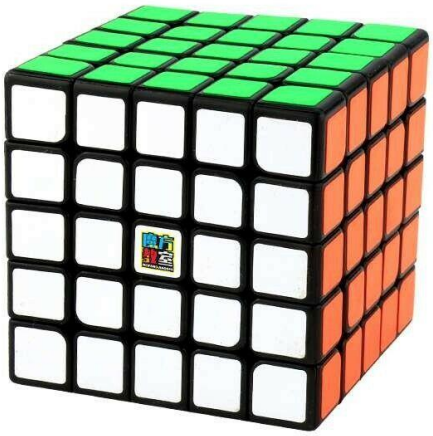
Test Cases:

Input	Output
3 3 1 1 1 1 0 1 1 1 1	1 0 1 0 0 0 1 0 1
3 4 0 1 2 0 3 4 5 2 1 3 1 5	0 0 0 0 0 4 5 0 0 3 1 0
4 4 1 2 3 4 5 0 7 8 9 10 11 12 13 14 15 16	1 0 3 4 0 0 0 0 9 0 11 12 13 0 15 16

3 3 0 2 3 4 5 6 7 8 9	0 0 0 0 5 6 0 8 9
4 3 1 2 3 4 0 6 7 8 9 0 11 12	0 0 3 0 0 0 0 0 9 0 0 0
3 3 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0
2 3 1 2 3 0 5 6	0 2 3 0 0 0

Problem#6: Bomber Man (3D)

You are creating a Bomber Man 3D with the field of size 5x5x5 (it is like Rubik 5x5x5 as picture below).



The rule is a player can put a bomb in any cell of the field. When a bomb explores, it will destroy all cells to the edge in 6 directions (i.e., up, down, left, right, front, back).

Your task is to write a program to count number of destroyed cells in the field when number of bombs and position of bombs are provided.

Input

The first line of the input is a number of bomb in the field.

The following lines are position of each bomb located in the field.

Output

The output is the total number of destroyed cells in the field.

Test Cases:

Input	Output
2 0 0 0 2 3 4	26
3 1 1 2 0 3 4 2 1 3	37
5 1 2 0 0 0 0 4 3 1 2 2 1 0 2 4	54
10 1 2 0 0 0 0 4 3 1 2 2 1 0 2 4 1 1 1 3 2 3 2 2 2 4 4 4 4 0 1	86
10 3 1 2 4 3 4 3 3 3 2 1 4 0 1 0	84

4 0 2 3 0 1 4 2 0 0 1 2 3 4 2	