Array

You do not need to write the Array class, 2D Array class, printArray() etc. functions unless specified in the problem.

Just design the function specified in the problem.

No need to write driver code.

This document is a compilation of previous semester's exam (midterm/final) problems and lab quiz problems. Try to practice relevant problems from online judges as well.

1.

I. Suppose you are given a multi-dimensional array with dimension 5x5x3. What is the multidimensional index for the linear index 65?

II. **Complete** the function **compress_matrix** that takes a 2D array as a parameter and return a new compressed 2D array. In the given array the number of rows and columns will always be even. **Compressing a matrix means grouping elements in 2x2 blocks and sums the elements within each block. Check the sample input output for further clarification.**

Hint: Generally the block consists of the (i,j), (i+1,j), (i,j+1) and (i+1, j+1) elements for 2x2 blocks.

You cannot use any built-in function except len() and range(). You can use the np variable to create an array.

Python Notation	**Java Notation**

import numpy as np
def compress_matrix (mat):
    # To Do
	public int[ ][ ] compress_matrix (int[] [] mat) {
	    // To Do
	}

Sample Input array	**All Box (No need to create these arrays)**	**Returned Array**	**Explanation**
[[1, 2, 3, 4], [5, 6, 7, 8], [1, 3, 5, 2], [-2, 0, 6,-3]]	[[1, 2], [[3, 4], [5, 6]] [[7, 8]] [[1, 3], [[5, 2], [-2, 0]] [[6, -3]]	[[14, 22], [2, 10]]	[[1+2+5+6, 3+4+7+8], [1+3+-2+0, 5+2+6+-3]]

2.

I. Suppose you are given a multi-dimensional array with dimensions 4x5x4. What is the linear index of the multidimensional index [2][1][0]?

II. Suppose you're working as a cryptographer for a secret intelligence agency. You are given an encrypted matrix as an input. You have to decrypt it after some processing of this matrix.

To decrypt this message efficiently, you have a task to construct a method called **sum_diff(matrix)** which takes an encrypted matrix as input and returns a decrypted linear array. The process of finding the decrypted linear array is given below:

You have to find out the column-wise summations for each column and store the difference of subsequent column-wise summations in a new linear array.

<table>
<tr><th>Sample Input</th><th>Sample output</th><th>Explanation</th></tr>
<tr>
<td>
<table>
<tr><td>1</td><td>3</td><td>1</td></tr>
<tr><td>6</td><td>4</td><td>2</td></tr>
<tr><td>5</td><td>1</td><td>7</td></tr>
<tr><td>9</td><td>3</td><td>3</td></tr>
<tr><td>8</td><td>5</td><td>4</td></tr>
</table>
</td>
<td>
<table>
<tr><td>-13</td><td>1</td></tr>
</table>
</td>
<td>
Sum of 0th column = 29
Sum of 1st column = 16
Sum of 2nd column = 17

Therefore, the size of the resulting array is 2 and the array is:
<table>
<tr><td>16-29 = -13</td><td>17-16 = 1</td></tr>
</table>
</td>
</tr>
</table>

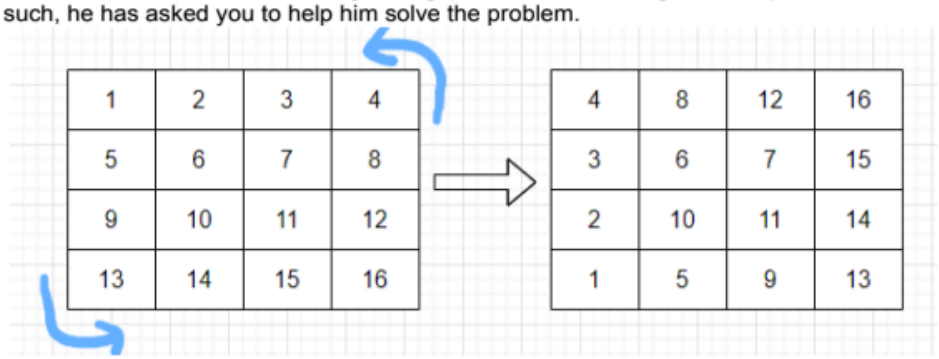
Python Notation	Java Notation

def sum_diff(matrix):
  # To Do
	public int[] sum_diff(matrix) {
	  // To Do
	}

3.

I. Suppose you are given a multi-dimensional array with dimensions 4x3x2x6. What is the corresponding linear index of the multidimensional index [2][1][0][4]?

II. Bob has already learnt about 2D matrices and its operations in the data structures course. However, he is currently facing problems in rotating a 2D square matrix. As such, he has asked you to help him solve the problem.



You should rotate the original input matrix's outer rows and columns in left, down, right, top manner without affecting the internal cells; **instead of creating a new matrix**.
Here, as you can see, if you rotate the given square matrix **anticlockwise**, 1st row will be 1st column, 1st column will be last row, last row will be last column and last column will be 1st row. The elements of the internal cells will be unchanged.

So given a square matrix of size n*n, your task is to complete the given method **rotate(matrix)** that takes a 2D square matrix as an input and returns the rotated matrix using **in-place** algorithm.
Hints of in-place: Rotating by one element at a time instead of an entire row/column should eliminate the need for any supporting data structures.

Sample Input	Sample Output
| 1 | 2 | 3 | | 5 | 6 | 7 | | 9 | 10 | 11 |	| 3 | 7 | 11 | | 2 | 6 | 10 | | 1 | 5 | 9 |

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

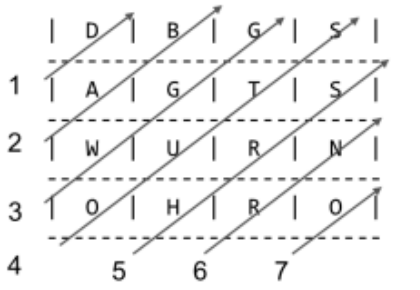
Python Notation	Java Notation

def rotate(matrix):
# To Do 	
	public int[][] rotate(int[][]matrix) {
	// To Do
	}

4.

I. Suppose you are given a multi-dimensional array with dimensions 3x4x2. What is the linear index of the multidimensional index [2][1][0]?

II. You have hidden a password into a square matrix for your friend. Only you know that the password can be achieved through traversing the array in a zigzag fashion. Now help your friend develop a function/method that will take the 2D array as input and print the password in the output console.
Complete the function zig_zag which will take a NxN 2D array and print all the values in a zigzag pattern. Each value will be separated by a space. Assume the function will always receive a NxN 2D array. Your code should work for any NxN array.

Input Array	Output	Zigzag traversal explanation
| D | B | G | S | | A | G | T | S | | W | U | R | N | | O | H | R | O |	D A B W G G O U T S H R S R N O	

Hint: You need to follow the arrowed line 1 to line 7 to find out the zigzag pattern. Use one loop to print the upper triangle (line 1 to 4) and another loop to print the lower triangle (line 5 to 7).

Python Notation	Java Notation

`def zig_zag(arr):`
`  # To Do`
	`public void zig_zag(int[][] arr) {`
	`  // To Do`
	`}`

5.

You are given an n x n 2D matrix representing an image. Rotate the image by **90 degrees (clockwise)**. You have to rotate the image in-place, which means you have to modify the input 2D matrix directly. You can assume that the input matrix will always be a square matrix.

Sample Input	Sample Output
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]	[[7, 4, 1], [8, 5, 2], [9, 6, 3]]
[[5, 1, 9,11], [2, 4, 8,10], [13, 3, 6, 7], [15,14,12,16]]	[[15,13, 2, 5], [14, 3, 4, 1], [12, 6, 8, 9], [16, 7,10,11]]

6.

Question: You are given an n x n 2D matrix representing an image. Rotate the image by **90 degrees (counter-clockwise)**. You have to rotate the image in-place, which means you have to modify the input 2D matrix directly. You can assume that the input matrix will always be a square matrix.

Sample Input	Sample Output
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]	[[3, 6, 9], [2, 5, 8], [1, 4, 7]]
[[5, 1, 9, 11], [2, 4, 8, 10], [13, 3, 6, 7], [15, 14, 12, 16]]	[[11, 10, 7, 16], [9, 8, 6, 12], [1, 4, 3, 14], [5, 2, 13, 15]]

7.

Question: You have joined as an intern under a renowned cyber specialist and have to assist her in locating a murderer. After staring at the cryptic messages sent by the murderer for hours, you have finally figured out the pattern: your main clue is on the primary diagonal, that is, you will start comparing the alphabets from the top left and bottom right till you reach the end point of the diagonal and the mismatched alphabets are your potential apartment names.

The murderer thinks that the law enforcement is incapable which is why this person sent a hint with the latest message. The hint states that upon decoding this message, you will be able to pinpoint the possible apartment numbers the murderer may be hiding in. The clock is ticking and you have 20 minutes to bring justice to the victims.

Exam Instructions: The 2D array will be a squared one.
The order of the apartment names do not matter since you have to search each one anyway.

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>
<table>
<tr><td>A</td><td>D</td><td>M</td><td>Q</td></tr>
<tr><td>E</td><td>S</td><td>Y</td><td>K</td></tr>
<tr><td>J</td><td>F</td><td>O</td><td>L</td></tr>
<tr><td>P</td><td>X</td><td>J</td><td>A</td></tr>
</table>
</td><td>Possible Apartment Names: S O</td></tr>
<tr><td>
<table>
<tr><td>A</td><td>D</td><td>M</td><td>Q</td><td>F</td></tr>
<tr><td>E</td><td>S</td><td>Y</td><td>K</td><td>W</td></tr>
<tr><td>J</td><td>F</td><td>O</td><td>L</td><td>T</td></tr>
<tr><td>P</td><td>X</td><td>J</td><td>S</td><td>Y</td></tr>
<tr><td>V</td><td>R</td><td>K</td><td>G</td><td>P</td></tr>
</table>
</td><td>Possible Apartment Names: A P O</td></tr>
</table>

```
//python
def decodeMessage(codeword):
        #To Do
        pass
```

```
//java
public static void decodeMessage(String[][] codeword){
    //To Do
}
```

8.

Question: You have joined as an intern under a renowned cyber specialist and have to assist her in locating a murderer. After staring at the cryptic messages sent by the murderer for hours, you have finally figured out the pattern: your main clue is on the secondary diagonal, that is, you will start comparing the alphabets from the top right and bottom left till you reach the end point of the diagonal and the mismatche alphabets are your potential apartment names.

The murderer thinks that the law enforcement is incapable which is why this person sent a hint with the latest message. The hint states that upon decoding this message, you will be able to pinpoint the possibl apartment numbers the murderer may be hiding in. The clock is ticking and you have 20 minutes to bring justice to the victims.

Exam Instructions: The 2D array will be a squared one.
The order of the apartment names do not matter since you have to search each one anyway.

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td>
<table>
<tr><td>A</td><td>D</td><td>M</td><td>Q</td></tr>
<tr><td>E</td><td>S</td><td>Y</td><td>K</td></tr>
<tr><td>J</td><td>Y</td><td>O</td><td>L</td></tr>
<tr><td>P</td><td>X</td><td>J</td><td>A</td></tr>
</table>
</td><td>Possible Apartment Names: Q P</td></tr>
<tr><td>
<table>
<tr><td>A</td><td>D</td><td>M</td><td>Q</td><td>F</td></tr>
<tr><td>E</td><td>S</td><td>Y</td><td>K</td><td>W</td></tr>
<tr><td>J</td><td>F</td><td>O</td><td>L</td><td>T</td></tr>
<tr><td>P</td><td>X</td><td>J</td><td>S</td><td>Y</td></tr>
<tr><td>F</td><td>R</td><td>K</td><td>G</td><td>P</td></tr>
</table>
</td><td>Possible Apartment Names: K O X</td></tr>
</table>

```
//python
def decodeMessage(codeword):
    #To Do
    pass
```

```
//java
public static void decodeMessage(String[][] codeword){
    //To Do
}
```

9.

Find The Queen's Age 👑

In a kingdom, the ages of noble members are recorded in an $N \times M$ matrix, where each cell represents a noble's age. The Queen's age is determined by identifying *the row with the smallest sum of ages* and *the column with the smallest sum of ages*, then *selecting the value at their intersection*. It is guaranteed that there is only one such minimum row and one such minimum column. Now, write a function, **findQueenAge(matrix)**, that takes the matrix as a parameter and returns the Queen's age.

Sample Given Matrix	Sample Output	Explanation
50 45 90 35 70 80 60 95 40 30 25 75	**30**	• Smallest row sum is found in the 3rd row. • Smallest column sum is found in the 2nd column. • Finally, Queen's Age is found at the intersection of the 3rd row and 2nd column.

[You are not allowed to use any built-in functions]

10.

Find The Strongest Energy Source ⚡

In a futuristic city, an $N \times N$ energy matrix records the power levels of energy sources. The strongest energy source is determined by finding *the column with the largest sum of power levels* and *the row with the largest sum of power levels*, then *selecting the value at their intersection*. There is guaranteed to be only one such maximum column and one such maximum row. Now, write a function, **findStrongSource(matrix),** that takes a matrix as a parameter and returns the strongest energy source.

Sample Given Matrix	Sample Output	Explanation
5 3 6 7 4 9 1 2 8	**9**	• Largest column sum is found in the 3rd column. • Largest row sum is found in the 2nd row. • Finally, Strong Source is found at the intersection of the 3rd column and 2nd row.

[You are not allowed to use any built-in functions]

11.

Tasks from Lab 1 - Lab 3 - 2D Matrix.docx

Tasks from Book - Practice Sheet 1 - Array.pdf

12.

Suppose you are a matrix transformation expert working on a mission for a tech company. You are given a 2D array (matrix) that contains important data, and your task is to transform the given matrix into a **symmetric lower triangular matrix** by **adding all elements above the main diagonal to their symmetric counterparts** below the diagonal, then setting the elements above the diagonal to zero.
❒ You need to implement the function **toSymLowerTri()**, which takes the matrix as a parameter and returns the transformed matrix.
❒ *Note: Your code should handle square matrices of any size. For non-square matrices, the function should return the matrix unchanged. You are not allowed to create any new matrices.*

Sample Input	Sample Output	Explanation
matrix= 8 2 1 3 5 4 6 9 7	8 0 0 5 5 0 7 13 7	The values above the diagonal are added to their symmetric counterparts: • 2 is added to its counterpart 3, resulting in 5. • 1 is added to its counterpart 6, resulting in 7. • 4 is added to its counterpart 9, resulting in 13. After the addition, the elements above the diagonal are set to zero.

13.

Suppose you are a matrix transformation expert working on a mission for a tech company. You are given a 2D array (matrix) that contains important data, and your task is to transform the given matrix into a **symmetric upper triangular matrix** by **adding all elements below the main diagonal to their symmetric counterparts** above the diagonal, then setting the elements below the diagonal to zero.
❒ You need to implement the function **toSymUpperTri()**, which takes the matrix as a parameter and returns the transformed matrix.
❒ *Note: Your code should handle square matrices of any size. For non-square matrices, the function should return the matrix unchanged. You are not allowed to create any new matrices.*

Sample Input	Sample Output	Explanation
matrix= 8 2 1 3 5 4 6 9 7	8 5 7 0 5 13 0 0 7	The values above the diagonal are added to their symmetric counterparts: • 3 is added to its counterpart 2, resulting in 5. • 6 is added to its counterpart 1, resulting in 7. • 9 is added to its counterpart 4, resulting in 13. After the addition, the elements below the diagonal are set to zero.

14.

A university records student **quiz scores** in a **2D matrix**, where each **row** represents a different **student** and each **column** represents a different **quiz**. The university wants to analyze the **difficulty trend** of the quizzes. To do this, they need you to **decrypt the performance data** using the following steps:

1. **Compute the average score for each quiz .**
2. **Find the difference between consecutive quiz averages** to determine if quizzes are getting harder or easier.
3. **Return the list of differences**, which will be used to adjust future quiz difficulties.

Sample Input	Sample Output
85 90 78 88 85 80 75 95 85 92 88 82	**[4.5 -8.25]** **Explanation:** Quiz 0 avg = (85 + 88 + 75 + 92) / 4 = 85.0 Quiz 1 avg = (90 + 85 + 95 + 88) / 4 = 89.5 Quiz 2 avg = (78 + 80 + 85 + 82) / 4 = 81.25 Quiz 1 - Quiz 0 = 89.5 - 85.0 = 4.5 Quiz 2 - Quiz 1 = 81.25 - 89.5 = -8.25

15.

A company records **monthly sales** of multiple products in a **2D matrix**, where each **row** represents a different **month** and each **column** represents a different **product**. Each value represents the **sales revenue** for that product in that month.

To analyze **sales trends**, decrypt the data as follows:

1. **Compute the total sales for each product .**
2. **Find the percentage contribution of each product to total sales.**
3. **Compute the difference between consecutive product percentages.**

Sample Input	Sample Output
100 200 300 150 250 350 120 180 290	**[13.40 15.98]** **Explanation** Product 0: 100 + 150 + 120 = 370 Product 1: 200 + 250 + 180 = 630 Product 2: 300 + 350 + 290 = 940 Total Sales = 370 + 630 + 940 = 1940 Product 0: (370 / 1940) * 100 = 19.07% Product 1: (630 / 1940) * 100 = 32.47% Product 2: (940 / 1940) * 100 = 48.45% Product 1 - Product 0 = 32.47 - 19.07 = 13.40 Product 2 - Product 1 = 48.45 - 32.47 = 15.98

16.

Suppose you are an FBI field agent working on a mission. The HQ has sent you a secret **message** in 2D array format containing information about the **pin** code of a highly protected facility. Now, you need to decode the **pin** from the **message** using the method "**decodeMessage()**," which takes the **message** as a **parameter** and returns a decoded **pin**.

Decoding procedure: You need to find out the **absolute difference** between the diagonal and counter-diagonal elements and create a new array containing the resultant values.

Notice: Consider the message being **(MxM)** 2D array, where **"M"** is **even**.

Sample Input	Sample Output
message= [[1, 2, 3, -4], [5, -6, 7, 8], [9, -10, 11, 12], [13, 14, 15, -16]]	[5, 13, 21, 29]

Explanation

1st Diagonal element = 1
1st Counter Diagonal element = - 4
Absolute difference= | 1 - (- 4) | = 5

2nd Diagonal element = - 6
2nd Counter Diagonal element = 7
Absolute difference= | -6 - 7 | = 13

3rd Diagonal element = 11
3rd Counter Diagonal element = - 10
Absolute difference= | 11 - (- 10) | = 21

4th Diagonal element = -16
4th Counter Diagonal element = 13
Absolute difference= | -16 - 13 | = 29

17.

Suppose you are an FBI field agent working on a mission. The HQ has sent you a secret **message** in 2D array format containing information about the **pin** code of a highly protected facility. Now, you need to decode the **pin** from the **message** using the method "**decodeMessage()**," which takes the **message** as a **parameter** and returns a decoded **pin**.

Decoding procedure: You need to find out the **absolute average** between the diagonal and counter-diagonal elements and create a new array containing the resultant values.

Notice: Consider the message being **(MxM)** 2D array, where **"M"** is **even**.

Sample Input	Sample Output
message= [[1, 2, 3, -4], [5, -6, 7, 8], [9, -10, 11, 12], [13, 14, 15, -16]]	[1.5, 0.5, 0.5, 1.5]

Explanation:

1st Diagonal element = 1
1st Counter Diagonal element = - 4
Absolute average= |1+(-4)|/2 = 1.5

2nd Diagonal element = - 6
2nd Counter Diagonal element = 7
Absolute average= |-6+7|/2 = 0.5

3rd Diagonal element = 11
3rd Counter Diagonal element = - 10
Absolute average= |11+(-10)|/2 = 0.5

4th Diagonal element = -16
4th Counter Diagonal element = 13
Absolute average= |-16+13|/2 = 1.5