LAB 1: Linear Array & 2D Matrix

[CO5]

Instructions for students:

- Complete the following methods based on Array & 2D Matrix.
- You may use Java / Python to complete the tasks.
- DO NOT CREATE a separate folder for each task.
- If you are using JAVA, then follow the **Java template.**
- If you are using PYTHON, then follow the **Python template**.

NOTE:

- **YOU CANNOT USE ANY OTHER DATA STRUCTURE OTHER THAN ARRAYS.**
- **YOUR CODE SHOULD WORK FOR ANY VALID INPUTS.**

[Make changes to the Sample Inputs and check whether your program works correctly]

The Lab Tasks should be completed during the lab class
YOU HAVE TO SUBMIT ONLY THE ASSIGNMENT TASK

Total Assignment Tasks: 4
Total Marks: 20

Lab Tasks [NO NEED TO SUBMIT]

1. Merge Sorted Arrays

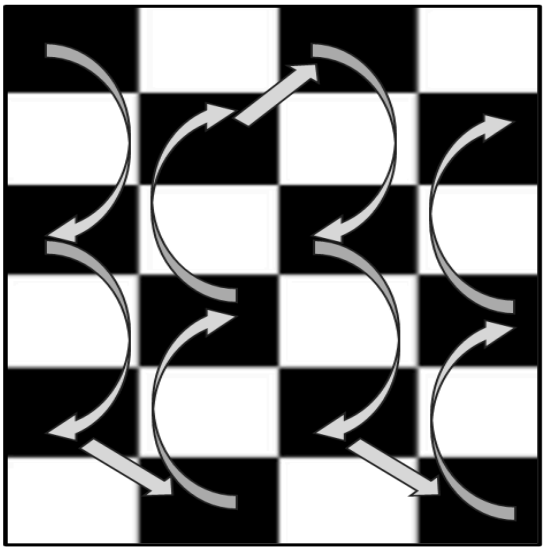
You are given two integer arrays arr1 and arr2, sorted in non-decreasing order (ascending order). Merge arr1 and arr2 into a single array sorted in non-decreasing order (ascending order) and return the new sorted array.

[Try to solve it in an optimized way]

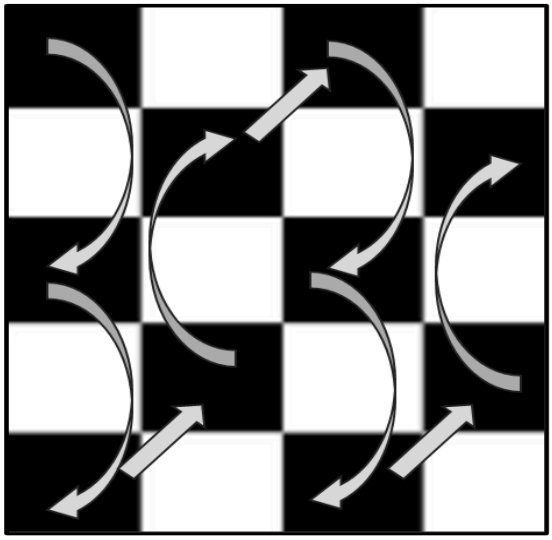
Sample Given Arrays	Sample Returned Array
arr1 = [1,2,3] arr2 = [2,5,6]	[1,2,2,3,5,6]
arr1 = [1,3,5,11] arr2 = [2,7,8]	[1,2,3,5,7,8,11]

2. Zigzag Walk:

As a child, you often played this game while walking on a tiled floor. You walked avoiding a certain color, for example white tiles (it almost felt like if you stepped on a white tile, you would die!). Now you are in a room of m x n dimension. The room has m*n black and white tiles. You step on the black tiles only. Your movement is like this:



OR



Now suppose you are given a 2D array which resembles the tiled floor. Each tile has a number. Can you write a method that will print
your walking sequence on the floor?

Hint 1: The first tile of the room is always black.
Hint 2: Look out for the number of rows in the matrix Notice the transition from column 0 to column 1 in the above figures

Sample Given Matrix	Sample Output
3 8 4 6 1 7 2 1 9 3 9 0 7 5 8 2 1 3 4 0 1 4 2 8 6	3 9 1 1 2 4 7 2 4 9 1 8 6
	3 9

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

```
1 2
4 7
4 9
1 8
```

3. Decryption Process:

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 the given matrix.

To decrypt this message efficiently, you have a task to construct a method called **decrypt_matrix(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.

Sample Given Matrix	Sample Output	Explanation
1 3 1 6 4 2 5 1 7 9 3 3 8 5 4	-13 1	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: 16-29 = -13 | 17-16 = 1

<u>Assignment Tasks</u> [NEED TO SUBMIT]

1. Row Rotation Policy of Your Classroom [5 marks]

You are no longer permitted to choose your own seat on the new campus of your university. You must abide by the new regulations, which state that if you sit in the first row for the first week, you must shift to the second row for Week2, the third row for the Week3, and so on. In your classroom, there are a total of 6 rows and 5 columns. Your friend “AA” wants to know from you that on the upcoming exam week in which row he will be in. Your task is to implement a function **`row_rotation(exam_week, seat_status)`** that takes the exam_week and a 2D array of current seat status as input and returns the row number in which your friend **“AA”** will be seated and print the seat status for that week.

Sample User Input & Given Matrix	**Output**
User Input: exam_week = 3 **Given Matrix:** **current_seat_status =** | A | B | C | D | E | | F | G | H | I | J | | K | L | M | N | O | | P | Q | R | S | T | | U | V | W | X | Y | **| Z | AA | BB | CC | DD |**	| U | V | W | X | Y | **| Z | AA | BB | CC | DD |** | A | B | C | D | E | | F | G | H | I | J | | K | L | M | N | O | | P | Q | R | S | T | Your friend AA will be on row 2
<u>Explanation</u>: According to the example the exam is 3 weeks away from the current seat status. So, during the exam day, the row containing **AA** will move **2 times** (basically the whole matrix will rotate downward vertically).	

2. Matrix Compression [5 marks]

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 row and column 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]]

3. Game Arena [5 marks]

Suppose you and your friends are in the world of '*Alice in Borderland*' where you decided to take part in a game and entered the *game arena*. As a team, you need to gain **at least 10 points** in order to keep surviving in the borderland. Otherwise, you will be out of the game and your team will be banished for good. Now, the arena has a 2D array like structure where players of a team are given certain positions with values that are **multiples of 50**. By staying in these positions, every player can gain points from the cells above, below, left and right (not diagonally) only if those cells **contain 2** [The cells containing 1s and 0s are to be avoided]. For each player, add from these cells containing 2s to your total points for the team to keep on surviving in borderland. **Be careful about corner cases.**

Your task is to write a method which tells us whether your team is out or has survived the game.

Sample Given Matrix 1	Sample Output 1
| 0 | 2 | 2 | 0 | | **50** | 1 | 2 | 0 | | 2 | 2 | 2 | 0 | | 1 | **100** | 2 | 0 |	Points Gained: 6. Your team is out.

Explanation:
Player with value 50 has 2 in the cell below him (1 cell). Player with value 100 has 2 in the cell above and in the right cell (2 cells). So in total, they got (1+2)*2 = 6 points which was not enough to survive the game.

Sample Given Matrix 2	Sample Output 2
| 0 | 2 | 2 | 0 | 2 | | 1 | **50** | 2 | 1 | **100** | | 2 | 2 | 2 | 0 | 2 | | 0 | **200** | 2 | 0 | 0 |	Points Gained: 14. Your team has survived the game.

Explanation:
Player with value 50 has 2 in the cell above, cell below and the right cell (3 cells). Player with value 100 has 2 in the cell above and the cell below (2 cells). Player with value 200 has 2 in the above cell and right cell (2 cells). So in total, they gained (3+2+2)*2 = 14 points and survived the game.

Note: For the cell with value 2 that is common between 2 players in position (2,1), both gained 2 points each, so it's not like if one player already added those 2 points, another player cannot. In the Given Matrix 2, Player with value 50 and player with value 200 are sharing a common 2 but both of them got points.

4. Rotate Secret [5 marks]

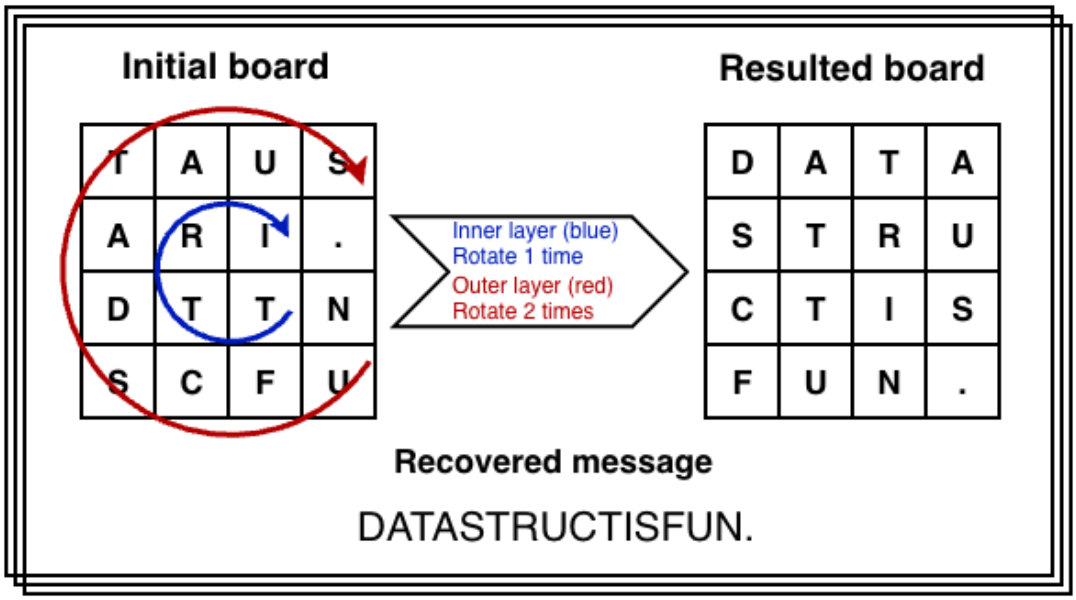
Your friend sent you a secret message scattered across a square board. The letters are scrambled, making the message unreadable. To recover it, you must rotate the concentric layers of the board clockwise, starting from the innermost layer and moving outward. You are given a square N × N matrix of characters, where both N (rows and columns) are even. Rotate each layer clockwise starting from the innermost layer, the innermost rotates once, and each outer layer rotates one time more than the one inside it. After performing all rotations, the hidden message will be revealed.

Write a method/function named **rotateSecret()**, which rotate the matrix clockwise :

- Innermost layer rotate 1 time
- Next outer layer rotate 2 times
- Next outer rotate 3 times, and so on.

After all successful rotations, print the recovered message.

Space Complexity of the solution must be O(1) [can't create new array]



Sample Given Matrix 1	After Processing	Output

board = {
{'T','A','U','S'},
{'A','R','I','.'},
{'D','T','T','N'},
{'S','C','F','U'}
}
	board = {
	{'D','A','T','A'},
	{'S','T','R','U'},
	{'C','T','I','S'},
	{'F','U','N','.'}
	};
		DATASTRUCTISFUN.

Explanation		
1 clockwise rotation on the innermost (2×2) layer, and 2 clockwise rotations on the outer (4×4) layer, each letter shifts to its correct position after these rotations. Then the message from the board is printed.		

Sample Given Matrix 2	After Processing	Output

board = {
    {'O','R','I','R','N','P'},
    {'G','S','A','A','L','R'},
    {'L','M','N','O','N','Y'},
    {'A','H','U','O','O','P'},
    {'T','F','C','T','H','S'},
    {'E','D','Y','O','C','K'}
}
	{
	    {'A','L','G','O','R','I'},
	    {'T','H','M','S','A','R'},
	    {'E','F','U','N','A','N'},
	    {'D','C','O','O','L','P'},
	    {'Y','T','H','O','N','R'},
	    {'O','C','K','S','P','Y'}
	}
		ALGORITHMSAREFUNANDCO
		OLPYTHONROCKSPY

Explanation		
1 clockwise rotation on the innermost (2×2) layer, and 2 clockwise rotations on the outer (4×4) layer, 3 clockwise rotations on the outer (6×6) layer, each letter shifts to its correct position after these rotations. Then the message from the board is printed.		